ARM® Cortex® -M

32-bit Microcontroller

NuMicro® Family
M4TK Series
Technical Reference Manual

For additional information or questions, please contact: Nuvoton Technology Corporation.

www.nuvoton.com

TABLE OF CONTENTS

List of Figures

List of Tables

1 GENERAL DESCRIPTION

The M4TK series includes a total of 33 ARM® Cortex®-M4F based products, including the M4TK CAN Series which is pin compatible with M051 LQFP48 and M058S LQFP64. By planning a complete product line, Nuvoton hopes to fulfill the demand for the ARM® Cortex®-M4F core with products at all levels and realize its customer commitment: total support for long-term competitiveness enhancement, and to fulfill their current product development demand and future innovation imagination.

The M4TK series embedded with the ARM® Cortex®-M4F core supports DSC (digital signal controller) and FPU (float point unit) and features high performance computing capability running up to 72 MHz, built-in 256/128KB Flash ROM, 32/16 KB SRAM complying with IEC60730, built-in boot ROM and independent 4 KB in-system programming Flash ROM for developing more flexible online upgrade code that support external UART, SPI, I²C, CAN and USB. It also supports EBI to provide greater flexibility for external memory. The entire M4TK series is provided with outstanding specifications: four 32-bit timers, dual watchdogs, and integrates plenty of peripherals such as PDMA and RTCs, five UARTs that support 16-byte FIFO, three sets of SPI controllers that support quad mode, two I²C devices that support SMBus and PMBus, two sets of I²S, two LINs, CAN bus, ISO-7816-3 interface, full-speed USB OTG, full-speed USB devices, 16-channel 12-bit ADC with 1 MSPS conversion rate, built-in reference voltage (V_{REF}) for circuit generation, 12-bit DAC, two analog comparators and temperature detectors.

The M4TK series provides two special designs. One is high-resolution 144 MHz PWM with high-speed electromechanical control timer and resolution<7ns. In conjunction with a driver ADC, it delivers hardware brake protection and pulse capture functions to save MCU computing burden and effectively carry out advanced computing required by motor control, making it exceptionally outstanding in industrial automation and motor control performance. The other is VAI (Voltage Adjustment Interface) system to support voltage level adjustment with individual I/O (1.8V-5.5V) for saving additional cost on adjusting the interface voltage difference of peripheral components.

The M4TK series also provides the wide operating voltage (2.5V-5.5V) and 5V-tolerance input I/O to significantly enhance system stability, industrial operating temperature (-40°C - 105°C), 22.1184 MHz internal RC oscillator (HIRC variation < ±2%) and 32.768 kHz external crystal oscillator to trim HIRC (HIRC variation < ±0.25%) working at -40˚C- 105˚C to boost system immunity and adequately fulfill the high precision demand of communications. The M4TK series is specifically suitable for high-performance and high-precision applications, such as industrial control, system automation, security surveillance, autotronics and digital power control.

Product Line	USB	CAN	UART	I²C	I²S	SPI	PWM	ADC	DAC	RTC	EBI
M4TK	●	●	●	●	●	●	●	●	●	●	●

Table 1-1 Key Features Support Table

The NuMicro® M4TK series is suitable for a wide range of applications such as:

- Industrial Automation
- PLCs
- Inverters
- Home Automation
- Security Alarm System
- Power Metering
- Portable Data Collector
- Portable RFID Reader

- System Supervisors
- Smart Card Reader
- Printer
- Bar Code Scanner
- Motor Control
- Digital Power

2 FEATURES

2.1 NuMicro® M4TK Features

- Core
  - ARM® Cortex®-M4F core running up to 72 MHz
  - Supports DSP extension with hardware divider
  - Supports IEEE 754 compliant Floating-point Unit (FPU)
  - Supports Memory Protection Unit (MPU)
  - One 24-bit system timer
  - Supports Low Power Sleep mode by WFI and WFE instructions
  - Single-cycle 32-bit hardware multiplier
  - Supports programmable 16 level priorities of Nested Vectored Interrupt Controller (NVIC)
  - Supports programmable mask-able interrupts
- Built-in LDO for wide operating voltage ranged from 2.5V to 5.5V
- Flash Memory
  - Supports 128/256 KB application ROM (APROM)
  - Supports 4 KB Flash for loader (LDROM)
  - Supports Data Flash with configurable memory size
  - Supports In-System-Programming (ISP), In-Application-Programming (IAP) update embedded flash memory
  - Supports 2 KB page erase for all embedded flash
- Boot Loader
  - 16 KB embedded ROM
  - Supports Nuvoton native In-System-Programming (ISP) for UART0, SPI0, I^2C0, CAN[*1] and USB[*2]
  - Supports direct boot from Boot Loader by pin selection
- SRAM Memory
  - 32/16 KB embedded SRAM
  - 16/8 KB with hardware parity check
  - Supports byte-, half-word- and word-access
  - Supports exception (NMI) generated once a parity check error occurs
  - Supports PDMA mode
- PDMA (Peripheral DMA)
  - Supports 12/8 independent configurable channels for automatic data transfer between memories and peripherals
  - Supports Normal and Scatter-Gather Transfer modes
  - Supports two types of priorities modes: Fixed-priority and Round-robin modes
  - Supports byte-, half-word- and word-access
  - Auto increment of the source and destination address
  - Supports single and burst transfer type
- Clock Control
  - Built-in 22.1184 MHz internal high speed RC oscillator (HIRC) for system operation (variation < 2% at -40˚C ~ +105˚C)
  - Built-in 10 kHz internal low speed RC oscillator (LIRC) for Watchdog Timer and wake-up operation
  - Built-in 4~20 MHz external high speed crystal oscillator (HXT) for precise timing operation
  - Built-in 32.768 kHz external low speed crystal oscillator (LXT) for RTC function and low-power system operation

- Supports one PLL up to 144 MHz for high performance system operation, sourced from HIRC and HXT
- Supports clock failure detection for high/low speed external crystal oscillator
- Supports exception (NMI) generated once a clock failure detected
- Supports clock output

● GPIO

- Four I/O modes
- TTL/Schmitt trigger input selectable
- I/O pin configured as interrupt source with edge/level trigger setting
- Supports high driver and high sink current I/O (up to 20 mA at 5V)
- Supports software selectable slew rate control
- Supports 5V-tolerance function for following pins
  - ◆ PA.0 ~ PA.15, PC.0 ~ PC.7, PC.9 ~ PC.15, PD.5 ~ PD.8, PD.10 ~ PD.15, PE.0 ~ PE.14, PF.2, PF.5 ~ PF.6
  - ◆ PA.0 ~ PA.15, PB.14 ~ PB.14, PC.0 ~ PC.8, PC.10 ~ PC.13, PD.4 ~ PD.7, PD.12 ~ PD.15, PE.0 ~ PE.1, PE.3 ~ PE.5, PE.8 ~ PE.13, PF.2.
- Supports up to 85/55/42 GPIOs for LQFP100/64/48 respectively

● Timer

- Supports 4 sets of 32-bit timers with 24-bit up-timer and one 8-bit prescale counter
- Independent clock source for each timer
- Provides One-shot, Periodic, Toggle and Continuous Counting operation modes
- Supports event counting function to count the event from external pin
- Supports input capture function to capture or reset counter value

● Watchdog Timer

- Supports multiple clock sources from LIRC (default selection), HCLK/2048 and LXT
- 8 selectable time-out period from 1.6ms ~ 26.0sec (depending on clock source)
- Able to wake up from Power-down or Idle mode
- Interrupt or reset selectable on watchdog time-out

● Window Watchdog Timer

- Supports multiple clock sources from HCLK/2048 (default selection) and LIRC
- Window set by 6-bit counter with 11-bit prescale
- Able to wake up from Power-down or Idle mode
- Interrupt or reset selectable on time-out

● RTC

- Supports external power pin V_{BAT}
- Supports software compensation by setting frequency compensate register (FCR)
- Supports RTC counter (second, minute, hour) and calendar counter (day, month, year)
- Supports Alarm registers (second, minute, hour, day, month, year)
- Selectable 12-hour or 24-hour mode
- Automatic leap year recognition
- Supports periodic time tick interrupt with 8 period options 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second
- Supports wake-up function
- Supports 80 bytes spare registers
- Programmable spare register erase function
- Supports 32KHz Oscillator gain control
- Supports tamper detection function

● PWM

- Supports up to 12 independent PWM outputs with 16-bit resolution
- Supports maximum clock frequency up to 144MHz
- Supports 12-bit clock prescale
- Supports one-shot or auto-reload counter operation mode

- Supports up, down or up-down PWM counter type
- Supports synchronous function
- Supports dead time with maximum divided 12-bit prescale
- Supports brake function source from pin, comparator output and system safety events
- Supports PWM auto recovery function after brake condition removed
- Supports mask function and tri-state output for each PWM pin
- Supports PWM events interrupt
- Supports trigger EADC/DAC start conversion
- Supports up to 12 independent input capture channels with rising/falling capture and with counter reload option
- Supports capture counter with 16-bit resolution
- Supports capture interrupt
- Supports capture PDMA mode

● UART

- Supports up to four UARTs – UART0, UART1, UART2 and UART3
- Supports 16-byte FIFOs with programmable level trigger
- Supports auto flow control (CTS and RTS)
- Supports IrDA (SIR) function
- Supports RS-485 9-bit mode and direction control
- UART0 and UART1 support LIN function
- Programmable baud-rate generator up to 1/16 system clock
- Supports wake-up function
- Supports PDMA mode

● Smart Card Interface

- One set of ISO-7816-3 port
- Compliant to ISO-7816-3 T=0, T=1
- Separate receive / transmit 4 bytes entry FIFO for data payloads
- Programmable transmission clock frequency
- Programmable receiver buffer trigger level
- Programmable guard time selection (11 ETU ~ 266 ETU)
- A 24-bit and two 8 bit time-out counters for Answer to Request (ATR) and waiting times processing
- Supports auto inverse convention function
- Supports stop clock level and clock stop (clock keep) function
- Supports transmitter and receiver error retry and error limit function
- Supports hardware activation/deactivation sequence process
- Supports hardware warm reset sequence process
- Supports hardware auto deactivation sequence when detect the card is removal
- Supports UART function

● SPI

- Supports one set of SPI Quad controller – SPI0
- Supports Master or Slave mode operation
- Supports 2-bit Transfer mode
- Supports Dual and Quad I/O Transfer mode
- Configurable bit length of a transfer word from 8 to 32-bit
- Provides separate 8-level depth transmit and receive FIFO buffers
- Supports MSB first or LSB first transfer sequence
- Supports the byte reorder function
- Supports Byte or Word Suspend mode
- Supports PDMA mode
- Supports 3-wired, no slave select signal, bi-direction interface
- Master up to 32 MHz, and Slave up to 16 MHz (when chip works at V_{DD} = 5V)

● SPI/ I^2S

- Supports up to two sets of SPI controllers – SPI1 and SPI2
- Supports Master or Slave mode operation
- Configurable bit length of a transfer word from 8 to 32-bit
- Provides separate 4-level depth transmit and receive FIFO buffers
- Supports MSB first or LSB first transfer sequence
- Supports the byte reorder function
- Supports Byte or Word Suspend mode
- Supports 3-wire, no slave select signal, bi-direction interface
- Master mode up to 36 MHz and Slave mode up to 18 MHz (when chip works at V_{DD} = 5V)
- Supports up to two sets of I^2S by SPI controllers – SPI1 and SPI2
- Interface with external audio CODEC
- Supports Master and Slave mode
- Capable of handling 8-, 16-, 24- and 32-bit word sizes
- Supports mono and stereo audio data
- Supports PCM mode A, PCM mode B, I^2S and MSB justified data format
- Each provides two 4-word FIFO data buffers, one for transmitting and the other for receiving
- Generates interrupt requests when buffer levels cross a programmable boundary
- Each supports two PDMA requests, one for transmitting and the other for receiving

● I^2C

- Supports up to two sets of I^2C devices
- Supports Master/Slave mode
- Bidirectional data transfer between masters and slaves
- Multi-master bus (no central master)
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus
- Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer
- Programmable clocks allow versatile rate control
- Supports multiple address recognition (four slave address with mask option)
- Supports SMBus and PMBus
- Supports speed up to 1Mbps
- Supports multi-address Power-down wake-up function

● CAN 2.0

- Supports up to one set of CAN controller
- Supports CAN protocol version 2.0 part A and B
- Bit rates up to 1M bit/s
- Each supports 32 Message Objects
- Each Message Object has its own identifier mask
- Programmable FIFO mode (concatenation of Message Object)
- Supports interrupts
- Disabled Automatic Re-transmission mode for Time Triggered CAN applications
- Supports power-down wake-up function

● USB 2.0 FS Controller

- Supports one set of USB 2.0 FS OTG
- FS Host compatible with Open HCI 1.0 specification
- Compliant to USB specification version 2.0
- OTG compliant with USB OTG Supplement 1.3
- On-chip USB Transceiver
- Supports Control, Bulk In/Out, Interrupt and Isochronous transfers
- Auto suspend function when no bus signaling for 3 ms

- Provides 8 programmable endpoints
- Supports 512 Bytes internal SRAM as USB buffer
- Provides remote wake-up capability
- On-chip 5V to 3.3V LDO for USB PHY

● EBI

- Supports two dedicated external chip select pins for each memory block
- Supports accessible space up to 1 MB for each bank, actually external addressable space is dependent on package pin out
- Supports 8-/16-bit data width
- Supports byte write in 16-bit data width mode
- Supports PDMA mode
- Supports Address/Data multiplexed Mode
- Supports Timing parameters individual adjustment for each memory block

● EADC

- Analog input voltage range: 0~ V_{REF} (Max to AV_{DD})
- Supports single 12-bit SAR ADC conversion
- 12-bit resolution and 10-bit accuracy is guaranteed
- Up to 1MSPS conversion rate at 5.0V
- Up to 16 external single-ended analog input channels
- Up to 8 differential analog input pairs
- Supports single ADC interrupt
- Supports external V_{REF} pin
- Support internal reference voltages from Band-gap and Voltage divider
- An A/D conversion can be triggered by Software enable, External pin, Timer 0~3 overflow pulse trigger and PWM trigger
- Supports 3 internal channels for V_{BAT}, band-gap VBG input and Temperature sensor input
- Supports PDMA transfer

● DAC

- Supports a 12-bit voltage type DAC
- Rail to rail settle time 8us
- External reference voltage V_{REF}
- Max. output voltage AV_{DD} -0.2V at buffer mode
- Conversion started by software enable or PDMA trigger
- Supports PDMA mode

● Touch Key

- Supports up to 16 touch keys
- Supports programmable sensitivity adjustment for each channel
- Supports programmable scan speed for different applications
- Supports any key Wake-up for low-power applications
- Supports manual/one-time or periodic key-scan initiation
- Supports programmable interrupt options for automatic key-scan and interrupt generation

● Analog Comparator

- Up to two rail-to-rail analog comparators
- Supports a multiplexed I/O pin at positive node.
- Supports I/O pins, band-gap, Voltage divider and DAC output at negative node
- Supports programmable speed and power consumption
- Interrupts generated when compare results change (Interrupt event condition is programmable)
- Supports Power-down Wake-up
- Supports triggers for break events and cycle-by-cycle control for PWM

- Cyclic Redundancy Calculation Unit
  - Supports four common polynomials CRC-CCITT, CRC-8, CRC-16, and CRC-32
  - Programmable initial value
  - Supports programmable order reverse setting for input data and CRC checksum
  - Supports programmable 1's complement setting for input data and CRC checksum.
  - Supports 8-/16-/32-bit of data width
  - Interrupt generated once checksum error occurs
- Voltage Adjustable Interface
  - Supports user Configurable 1.8~5.5V I/O Interface with a dedicated power input (V_{DDIO})
  - Supports UART1, SPI0, SPI1, I^2C1 or I^2C0 interface
- Supports 96-bit Unique ID (UID)
- Supports 128-bit Unique Customer ID (UCID)
- One built-in temperature sensor with 1℃ resolution
- Brown-out detector
  - With 4 levels: 4.4 V/ 3.7 V/ 2.7 V/ 2.2 V
  - Supports Brown-out Interrupt and Reset option
- Low Voltage Reset
  - Threshold voltage levels: 2.0 V
- Operating Temperature: -40℃~105℃
- Packages
  - All Green package (RoHS)
  - LQFP 100-pin (14mm x 14mm)
  - LQFP 64-pin (10mm x 10mm)
  - LQFP 64-pin (7mm x 7mm)
  - LQFP 48-pin (7mm x 7mm)

Note:
*1: For optional part numbers which support CAN
*2: For optional part numbers which support USB

3 ABBREVIATIONS

Acronym	Description
ACMP	Analog Comparator Controller
ADC	Analog-to-Digital Converter
AES	Advanced Encryption Standard
APB	Advanced Peripheral Bus
AHB	Advanced High-Performance Bus
BOD	Brown-out Detection
CAN	Controller Area Network
DAP	Debug Access Port
DES	Data Encryption Standard
EBI	External Bus Interface
EPWM	Enhanced Pulse Width Modulation
FIFO	First In, First Out
FMC	Flash Memory Controller
FPU	Floating-point Unit
GPIO	General-Purpose Input/Output
HCLK	The Clock of Advanced High-Performance Bus
HIRC	22.1184 MHz Internal High Speed RC Oscillator
HXT	4~20 MHz External High Speed Crystal Oscillator
IAP	In Application Programming
ICP	In Circuit Programming
ISP	In System Programming
LDO	Low Dropout Regulator
LIN	Local Interconnect Network
LIRC	10 kHz internal low speed RC oscillator (LIRC)
MPU	Memory Protection Unit
NVIC	Nested Vectored Interrupt Controller
PCLK	The Clock of Advanced Peripheral Bus
PDMA	Peripheral Direct Memory Access
PLL	Phase-Locked Loop
PWM	Pulse Width Modulation
QEI	Quadrature Encoder Interface
SD	Secure Digital
SPI	Serial Peripheral Interface

SPS	Samples per Second
TDES	Triple Data Encryption Standard
TK	Touch Key
TMR	Timer Controller
UART	Universal Asynchronous Receiver/Transmitter
UCID	Unique Customer ID
USB	Universal Serial Bus
WDT	Watchdog Timer
WWDT	Window Watchdog Timer

Table 3-1 List of Abbreviations

4 PARTS INFORMATION LIST AND PIN CONFIGURATION

4.1 NuMicro® M4TK Selection Guide

4.1.1 NuMicro® M4TK Naming Rule

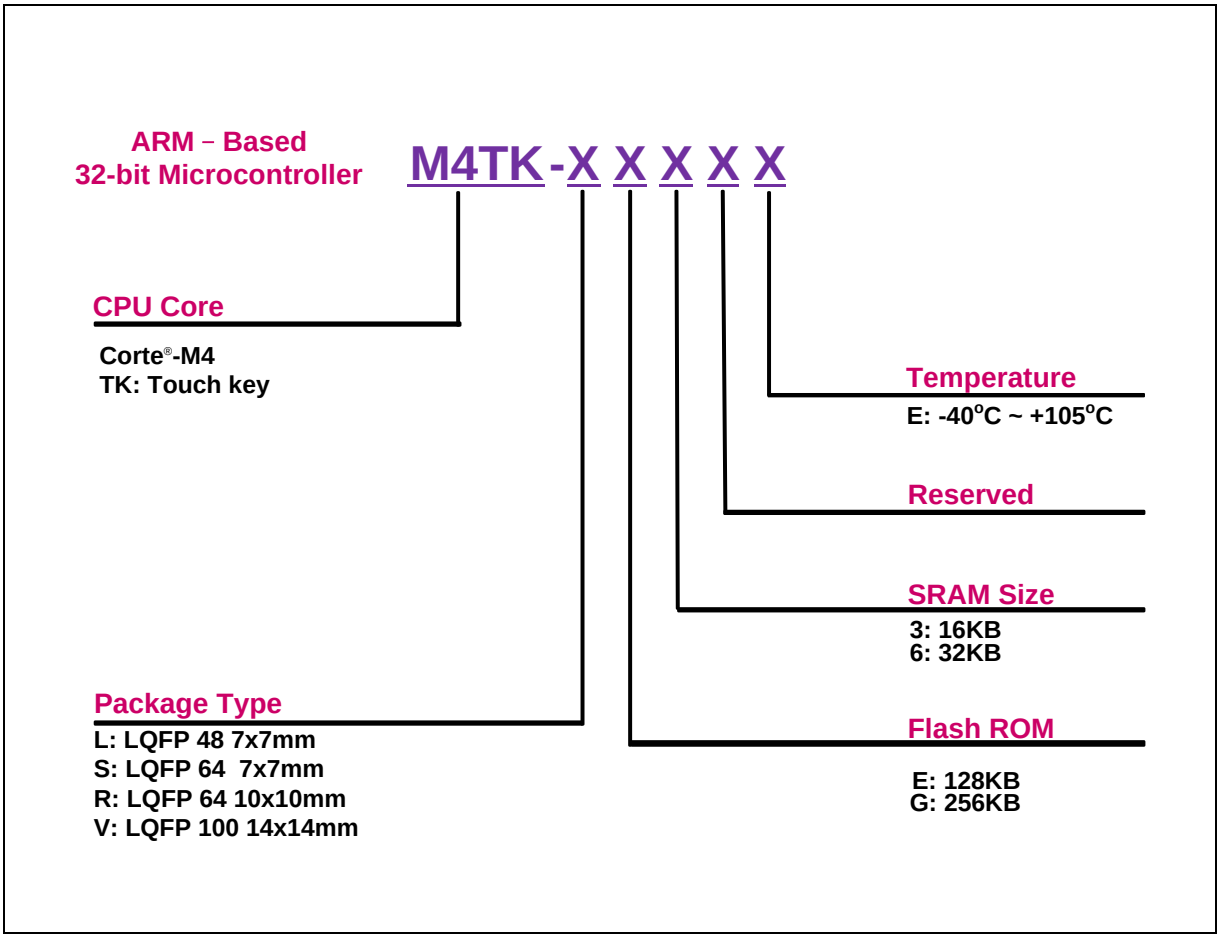


Figure 4.1-1 NuMicro® M4TK Selection Code

In this document, M4TKxG means the part numbers which include 256 KB flash, M4TKxE means the part numbers which include 128 KB flash.

4.1.2 NuMicro® M4TK CAN Series (CAN+USB) Selection Guide

Part Number	Flash (KB)	SRAM (KB)	ISP Loader ROM (KB)	I/O	Timer	Connectivity						I^2S	USB	PWM	Analog Comp.	DAC (12-Bit)	ADC (12-Bit)	Touch Key	RTC	EBI	ICP/ISPI/AP	Package
						UART*	SC* (ISO-7816)	SPI	I^2C	CAN	LIN											
M4TKLG6AE	256	32	4	34	4	3+1	1	3	2	√	2	2	OTG	10	2	1	8-ch	6	√	√	√	LQFP 48
M4TKLE6AE	128	32	4	34	4	3+1	1	3	2	√	2	2	OTG	10	2	1	8-ch	6	√	√	√	LQFP 48
M4TKRG6AE	256	32	4	48	4	4+1	1	3	2	√	2	2	OTG	12	2	1	12-ch	11	√	√	√	LQFP 64
M4TKRE6AE	128	32	4	48	4	4+1	1	3	2	√	2	2	OTG	12	2	1	12-ch	11	√	√	√	LQFP 64
M4TKVG6AE	256	32	4	80	4	4+1	1	3	2	√	2	2	OTG	12	2	1	16-ch	16	√	√	√	LQFP 100
M4TKVE6AE	128	32	4	80	4	4+1	1	3	2	√	2	2	OTG	12	2	1	16-ch	16	√	√	√	LQFP 100

*Marked in this table (4+1) means 4 UART + 1 SC UART

*SC (ISO-7816) supports full duplex UART mode

4.2 Pin Configuration

4.2.1 NuMicro® M4TK CAN Series (CAN+USB) LQFP48 Pin Diagram

Corresponding Part Number: M4TKLG6AE, M4TKLE6AE

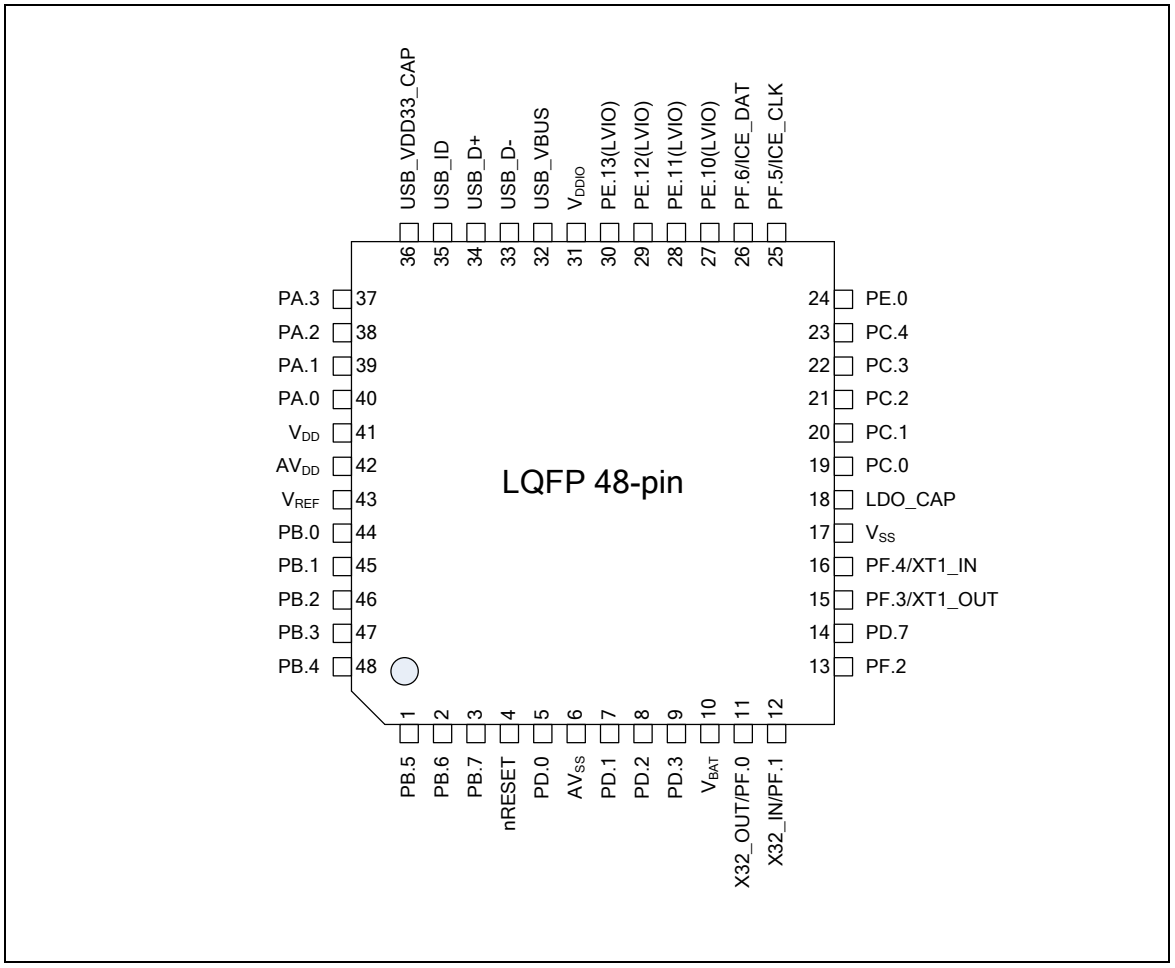


Figure 4.2-1 NuMicro® M4TK CAN Series (CAN+USB) LQFP 48-pin Diagram

4.2.2 NuMicro® M4TK CAN Series (CAN+USB) LQFP64 Pin Diagram

Corresponding Part Number: M4TKRG6AE, M4TKRE6AE

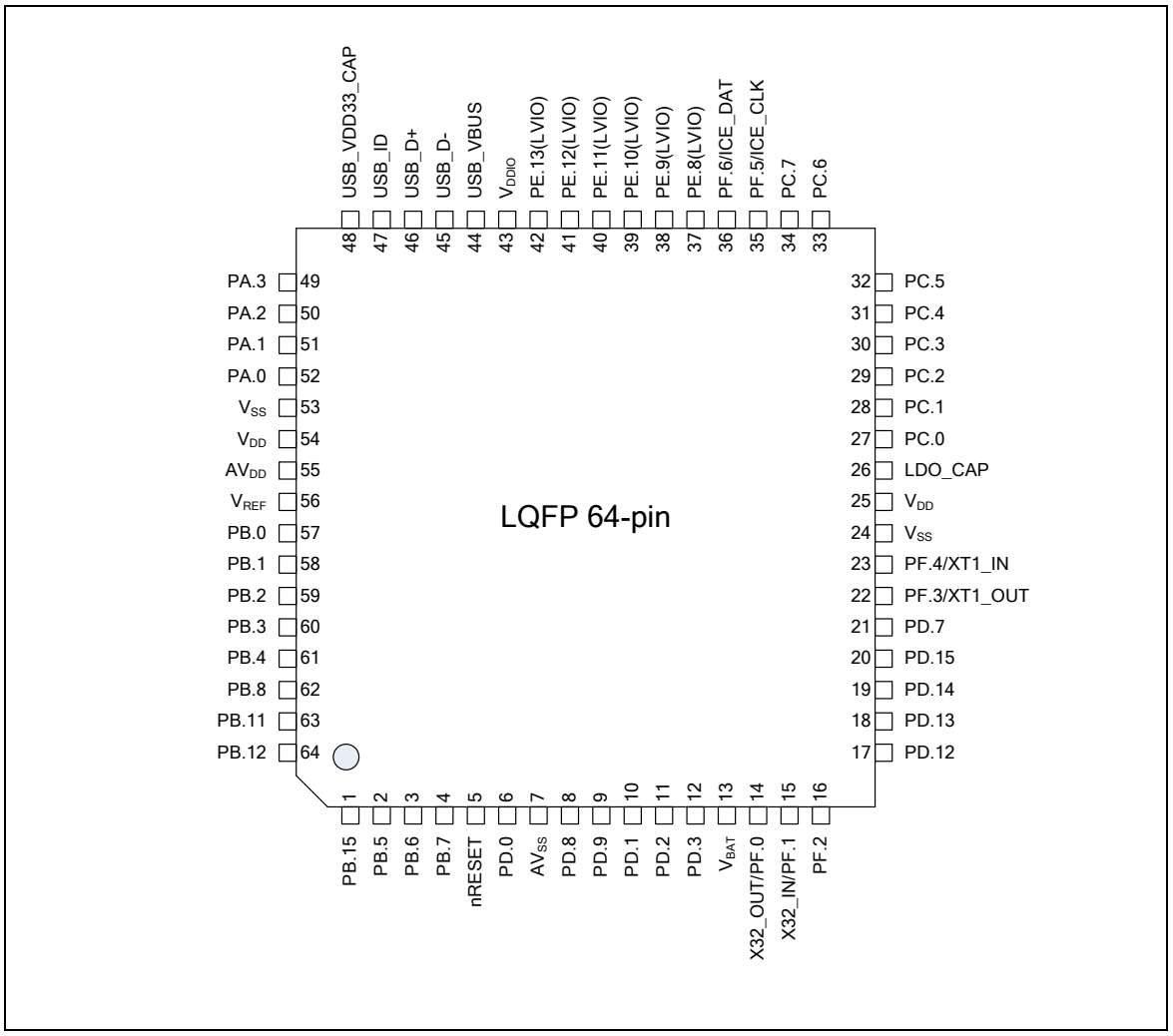


Figure 4.2-2 NuMicro® M4TK CAN Series (CAN+USB) LQFP 64-pin Diagram.

4.2.3 NuMicro® M4TK CAN Series (CAN+USB) LQFP100 Pin Diagram

Corresponding Part Number: M4TKVG6AE, M4TKVE6AE

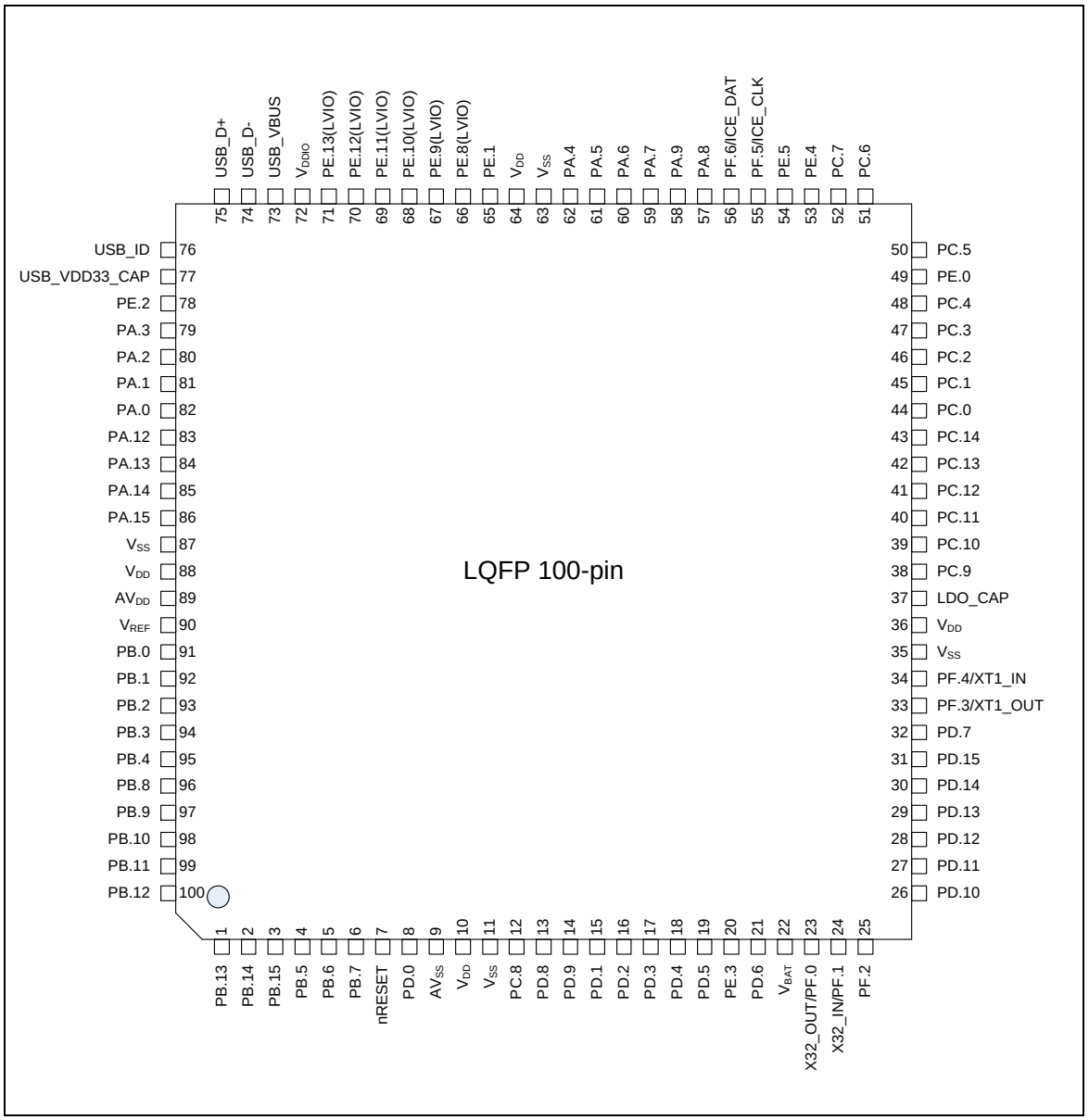


Figure 4.2-3 NuMicro® M4TK CAN Series (CAN+USB) LQFP 100-pin Diagram.

4.3 Pin Description

4.3.1 M4TK CAN Series(CAN+USB) LQFP48 Pin Description

MFP* = Multi-function pin. (Refer to section SYS_GPx_MFPL and SYS_GPx_MFPH)

PA.0 MFP0 means SYS_GPA_MFPL[3:0]=0x0.

PA.9 MFP5 means SYS_GPA_MFPH[7:4]=0x5.

Pin No.	Pin Name	Type	MFP*	Description
1	PB.5	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH13	A	MFP1	EADC analog input channel 13.
	SPI0_MOSI0	I/O	MFP2	SPI0 1st MOSI (Master Out, Slave In) pin.
	SPI1_MOSI	I/O	MFP3	SPI1 MOSI (Master Out, Slave In) pin.
	TK3	A	MFP4	Touch Key3.
	ACMP0_P2	A	MFP5	Comparator0 positive input pin.
	EBI_AD6	I/O	MFP7	EBI address/data bus bit 6.
2	PB.6	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH14	A	MFP1	EADC analog input channel 14.
	SPI0_MISO0	I/O	MFP2	SPI0 1st MISO (Master In, Slave Out) pin.
	SPI1_MISO	I/O	MFP3	SPI1 MISO (Master In, Slave Out) pin.
	TK4	A	MFP4	Touch Key4.
	ACMP0_P1	A	MFP5	Comparator0 positive input pin.
	EBI_AD5	I/O	MFP7	EBI address/data bus bit 5.
3	PB.7	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH15	A	MFP1	EADC analog input channel 15.
	SPI0_CLK	I/O	MFP2	SPI0 serial clock pin.
	SPI1_CLK	I/O	MFP3	SPI1 serial clock pin
	TK5	A	MFP4	Touch Key5.
	ACMP0_P0	A	MFP5	Comparator0 positive input pin.
	EBI_AD4	I/O	MFP7	EBI address/data bus bit 4.
4	nRESET	I	MFP0	External reset input: active LOW, with an internal pull-up. Set this pin low reset to initial state.
5	PD.0	I/O	MFP0	General purpose digital I/O pin.
	SPI1_I2SMCLK	O	MFP2	I2S1 master clock output pin.
	UART0_RXD	I	MFP3	Data receiver input pin for UART0.
	TK6	A	MFP4	Touch Key6.
	ACMP1_N	A	MFP5	Comparator1 negative input pin.
	INT3	I	MFP8	External interrupt3 input pin.
6	AV_{SS}	P	MFP0	Ground pin for analog circuit.

Pin No.	Pin Name	Type	MFP*	Description
7	PD.1	I/O	MFP0	General purpose digital I/O pin.
	PWM0_SYNC_IN	I	MFP2	PWM0 counter synchronous trigger input pin.
	UART0_TXD	O	MFP3	Data transmitter output pin for UART0.
	TK10	A	MFP4	Touch Key10.
	ACMP1_P2	A	MFP5	Comparator1 positive input pin.
	T0	I/O	MFP6	Timer0event counter input / toggle output
	EBI_nRD	O	MFP7	EBI read enable output pin.
8	PD.2	I/O	MFP0	General purpose digital I/O pin.
	STADC	I	MFP1	ADC external trigger input.
	T0_EXT	I	MFP3	Timer0 external capture input.
	TK11	A	MFP4	Touch Key11.
	ACMP1_P1	A	MFP5	Comparator1 positive input pin.
	PWM0_BRAKE0	I	MFP6	PWM0 break input 0
	EBI_nWR	O	MFP7	EBI write enable output pin.
	INT0	I	MFP8	External interrupt0 input pin.
9	PD.3	I/O	MFP0	General purpose digital I/O pin.
	T2	I/O	MFP1	Timer2 event counter input / toggle output
	T1_EXT	I	MFP3	Timer1 external capture input
	TK12	A	MFP4	Touch Key12.
	ACMP1_P0	A	MFP5	Comparator1 positive input pin.
	PWM0_BRAKE1	I	MFP6	PWM0 break input 1
	EBI_MCLK	O	MFP7	EBI external clock output pin
	INT1	I	MFP8	External interrupt1 input pin.
10	V_{BAT}		MFP0	Power supply by batteries for RTC and PF.0~PF.2.
11	PF.0	I/O	MFP0	General purpose digital I/O pin.
	X32_OUT	O	MFP1	External 32.768 kHZ (low speed) crystal output pin.
	INT5	I	MFP8	External interrupt5 input pin.
12	PF.1	I/O	MFP0	General purpose digital I/O pin.
	X32_IN	I	MFP1	External 32.768 kHZ (low speed) crystal input pin.
13	PF.2	I/O	MFP0	General purpose digital I/O pin.
	TAMPER	I/O	MFP1	TAMPER detector loop pin
14	PD.7	I/O	MFP0	General purpose digital I/O pin.
	PWM0_SYNC_IN	I	MFP3	PWM0 counter synchronous trigger input pin.
	T1	I/O	MFP4	Timer1 event counter input / toggle output

Pin No.	Pin Name	Type	MFP*	Description
	ACMP0_O	O	MFP5	Comparator0 output.
	PWM0_CH5	I/O	MFP6	PWM0 output/capture input.
	EBI_nRD	O	MFP7	EBI read enable output pin.
15	PF.3	I/O	MFP0	General purpose digital I/O pin.
	XT1_OUT	O	MFP1	External 4~20 MHz (high speed) crystal output pin.
	I2C1_SCL	I/O	MFP3	I2C1 clock pin.
16	PF.4	I/O	MFP0	General purpose digital I/O pin.
	XT1_IN	I	MFP1	External 4~20 MHz (high speed) crystal input pin.
	I2C1_SDA	I/O	MFP3	I2C1 data input/output pin.
17	V_{SS}	A	MFP0	Ground pin for digital circuit.
18	LDO_CAP	A	MFP0	LDO output pin. **Note:** This pin needs to be connected with a 1uF capacitor.
19	PC.0	I/O	MFP0	General purpose digital I/O pin.
	SPI2_CLK	I/O	MFP2	SPI2 serial clock pin.
	UART2_nCTS	I	MFP3	Clear to Send input pin for UART2.
	CAN0_TXD	I	MFP4	CAN bus transmitter input.
	PWM0_CH0	I/O	MFP6	PWM0 output/capture input.
	EBI_AD8	I/O	MFP7	EBI address/data bus bit 8.
	INT2	I	MFP8	External interrupt2 input pin.
20	PC.1	I/O	MFP0	General purpose digital I/O pin.
	CLKO	O	MFP1	Clock Out
	STDAC	I	MFP2	DAC external trigger input.
	UART2_nRTS	O	MFP3	Request to Send output pin for UART2.
	CAN0_RXD	I	MFP4	CAN bus receiver input.
	PWM0_CH1	I/O	MFP6	PWM0 output/capture input.
	EBI_AD9	I/O	MFP7	EBI address/data bus bit 9.
21	PC.2	I/O	MFP0	General purpose digital I/O pin.
	SPI2_SS	I	MFP2	SPI2 slave select pin.
	UART2_TXD	O	MFP3	Data transmitter output pin for UART2.
	ACMP1_O	O	MFP5	Comparator1 output .
	PWM0_CH2	I/O	MFP6	PWM0 output/capture input.
	EBI_AD10	I/O	MFP7	EBI address/data bus bit 10.
22	PC.3	I/O	MFP0	General purpose digital I/O pin.
	SPI2_MOSI	I/O	MFP2	SPI2 MOSI (Master Out, Slave In) pin.

Pin No.	Pin Name	Type	MFP*	Description
	UART2_RXD	I	MFP3	Data receiver input pin for UART2.
	USB_VBUS_ST	I	MFP4	USB external VBUS regulator status pin.
	PWM0_CH3	I/O	MFP6	PWM0 output/capture input.
	EBI_AD11	I/O	MFP7	EBI address/data bus bit 11.
23	PC.4	I/O	MFP0	General purpose digital I/O pin.
	SPI2_MISO	I/O	MFP2	SPI2 MISO (Master In, Slave Out) pin.
	I2C1_SCL	I/O	MFP3	I2C1 clock pin.
	USB_VBUS_EN	O	MFP4	USB external VBUS regulator enable pin.
	PWM0_CH4	I/O	MFP6	PWM0 output/capture input.
	EBI_AD12	I/O	MFP7	EBI address/data bus bit 12.
24	PE.0	I/O	MFP0	General purpose digital I/O pin.
	SPI2_CLK	I/O	MFP2	SPI2 serial clock pin.
	I2C1_SDA	I/O	MFP3	I2C1 data input/output pin.
	T2_EXT	I	MFP4	Timer2 external capture input
	SC0_CD	I	MFP5	SmartCard card detect pin.
	PWM0_CH0	I/O	MFP6	PWM0 output/capture input.
	EBI_nCS1	O	MFP7	EBI chip select 1 enable output pin.
	INT4	I	MFP8	External interrupt4 input pin.
25	PF.5	I/O	MFP0	General purpose digital I/O pin.
	ICE_CLK	I	MFP1	Serial wired debugger clock pin
26	PF.6	I/O	MFP0	General purpose digital I/O pin.
	ICE_DAT	I/O	MFP1	Serial wired debugger data pin
27	PE.10	I/O	MFP0	General purpose digital I/O pin.
	SPI1_MISO	I/O	MFP1	SPI1 MISO (Master In, Slave Out) pin.
	SPI0_MISO0	I/O	MFP2	SPI0 1st MISO (Master In, Slave Out) pin.
	UART1_nCTS	I	MFP3	Clear to Send input pin for UART1.
	I2C0_SMBAL	O	MFP4	I2C0 SMBus SMBALTER# pin
	SC0_DAT	I/O	MFP5	SmartCard data pin.
28	PE.11	I/O	MFP0	General purpose digital I/O pin.
	SPI1_MOSI	I/O	MFP1	SPI1 MOSI (Master Out, Slave In) pin.
	SPI0_MOSI0	I/O	MFP2	SPI0 1st MOSI (Master Out, Slave In) pin.
	UART1_nRTS	O	MFP3	Request to Send output pin for UART1.
	I2C0_SMBSUS	O	MFP4	I2C0 SMBus SMBSUS# pin (PMBus CONTROL pin)
	SC0_CLK	O	MFP5	SmartCard clock pin.

Pin No.	Pin Name	Type	MFP*	Description
29	PE.12	I/O	MFP0	General purpose digital I/O pin.
	SPI1_SS	I/O	MFP1	SPI1 slave select pin
	SPI0_SS	I/O	MFP2	SPI0 slave select pin.
	UART1_TXD	O	MFP3	Data transmitter output pin for UART1.
	I2C0_SCL	I/O	MFP4	I2C0 clock pin.
30	PE.13	I/O	MFP0	General purpose digital I/O pin.
	SPI1_CLK	I/O	MFP1	SPI1 serial clock pin
	SPI0_CLK	I/O	MFP2	SPI0 serial clock pin.
	UART1_RXD	I	MFP3	Data receiver input pin for UART1.
	I2C0_SDA	I/O	MFP4	I2C0 data input/output pin.
31	V_{DDIO}	A	MFP0	Power supply for PE.10~PE.13.
32	USB_VBUS	A	MFP0	Power supply from USB* host or HUB.
33	USB_D-	I	MFP0	USB differential signal D-.
34	USB_D+	I	MFP0	USB differential signal D+.
	USB_ID	I	MFP0	USB identification.
36	USB_VDD33_CAP	A	MFP0	Internal power regulator output 3.3V decoupling pin. **Note:** This pin needs to be connected with a 1uF capacitor.
37	PA.3	I/O	MFP0	General purpose digital I/O pin.
	USB_VBUS_ST	I	MFP1	USB external VBUS regulator status pin.
	UART0_RXD	I	MFP2	Data receiver input pin for UART0.
	UART0_nRTS	O	MFP3	Request to Send output pin for UART0.
	I2C0_SCL	I/O	MFP4	I2C0 clock pin.
	SC0_PWR	O	MFP5	SmartCard power pin.
	PWM1_CH2	I/O	MFP6	PWM1 output/capture input.
	EBI_AD3	I/O	MFP7	EBI address/data bus bit 3.
38	PA.2	I/O	MFP0	General purpose digital I/O pin.
	USB_VBUS_EN	O	MFP1	USB external VBUS regulator enable pin.
	UART0_TXD	O	MFP2	Data transmitter output pin for UART0.
	UART0_nCTS	I	MFP3	Clear to Send input pin for UART0.
	I2C0_SDA	I/O	MFP4	I2C0 data input/output pin.
	SC0_RST	O	MFP5	SmartCard reset pin.
	PWM1_CH3	I/O	MFP6	PWM1 output/capture input.
	EBI_AD2	I/O	MFP7	EBI address/data bus bit 2.
39	PA.1	I/O	MFP0	General purpose digital I/O pin.

Pin No.	Pin Name	Type	MFP*	Description
	UART1_nRTS	O	MFP1	Request to Send output pin for UART1.
	UART1_RXD	I	MFP3	Data receiver input pin for UART1.
	CAN0_TXD	I	MFP4	CAN bus transmitter input.
	SC0_DAT	I/O	MFP5	SmartCard data pin.
	PWM1_CH4	I/O	MFP6	PWM1 output/capture input.
	EBI_AD1	I/O	MFP7	EBI address/data bus bit 1.
40	PA.0	I/O	MFP0	General purpose digital I/O pin.
	UART1_nCTS	I	MFP1	Clear to Send input pin for UART1.
	UART1_TXD	O	MFP3	Data transmitter output pin for UART1.
	CAN0_RXD	I	MFP4	CAN bus receiver input.
	SC0_CLK	O	MFP5	SmartCard clock pin.
	PWM1_CH5	I/O	MFP6	PWM1 output/capture input.
	EBI_AD0	I/O	MFP7	EBI address/data bus bit 0.
	INT0	I	MFP8	External interrupt0 input pin.
41	V_{DD}	A	MFP0	Power supply for I/O ports and LDO source for internal PLL and digital function.
42	AV_{DD}	A	MFP0	Power supply for internal analog circuit.
43	V_{REF}	I	MFP0	Voltage reference input for ADC. **Note:** This pin needs to be connected with a 1uF capacitor.
44	PB.0	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH0	A	MFP1	EADC analog input.
	SPI0_MOSI1	I/O	MFP2	SPI0 2nd MOSI (Master Out, Slave In) pin.
	UART2_RXD	I	MFP3	Data receiver input pin for UART2.
	T2	I/O	MFP4	Timer2 event counter input / toggle output
	DAC	A	MFP5	DAC analog output
	EBI_nWRL	O	MFP7	EBI low byte write enable output pin.
	INT1	I	MFP8	External interrupt1 input pin.
45	PB.1	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH1	A	MFP1	EADC analog input channel 1.
	SPI0_MISO1	I/O	MFP2	SPI0 2nd MISO (Master In, Slave Out) pin.
	UART2_TXD	O	MFP3	Data transmitter output pin for UART2.
	T3	I/O	MFP4	Timer3 event counter input / toggle output
	SC0_RST	O	MFP5	SmartCard reset pin.
	PWM0_SYNC_OUT	O	MFP6	PWM0 counter synchronous trigger output pin.
	EBI_nWRH	O	MFP7	EBI high byte write enable output pin

Pin No.	Pin Name	Type	MFP*	Description
46	PB.2	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH2	A	MFP1	EADC analog input channel 2.
	SPI0_CLK	I/O	MFP2	SPI0 serial clock pin.
	SPI1_CLK	I/O	MFP3	SPI1 serial clock pin
	UART1_RXD	I	MFP4	Data receiver input pin for UART1.
	SC0_CD	I	MFP5	SmartCard card detect pin.
47	PB.3	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH3	A	MFP1	EADC analog input channel 3.
	SPI0_MISO0	I/O	MFP2	SPI0 1st MISO (Master In, Slave Out) pin.
	SPI1_MISO	I/O	MFP3	SPI1 MISO (Master In, Slave Out) pin.
	UART1_TXD	O	MFP4	Data transmitter output pin for UART1.
48	PB.4	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH4	A	MFP1	EADC analog input channel 4.
	SPI0_SS	I/O	MFP2	SPI0 slave select pin.
	SPI1_SS	I/O	MFP3	SPI1 slave select pin
	UART1_nCTS	I	MFP4	Clear to Send input pin for UART1.
	ACMP0_N	A	MFP5	Comparator0 negative input pin.
	EBI_AD7	I/O	MFP7	EBI address/data bus bit 7.

4.3.2 M4TK CAN Series(CAN+USB) LQFP64 Pin Description

MFP* = Multi-function pin. (Refer to section SYS_GPx_MFPL and SYS_GPx_MFPH)

PA.0 MFP0 means SYS_GPA_MFPL[3:0]=0x0.

PA.9 MFP5 means SYS_GPA_MFPH[7:4]=0x5.

Pin No.	Pin Name	Type	MFP*	Description
1	PB.15	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH12	A	MFP1	EADC analog input channel 12.
	TK2	A	MFP4	Touch Key2.
	ACMP0_P3	A	MFP5	Comparator0 positive input pin.
	EBI_nCS1	O	MFP7	EBI chip select 1 enable output pin.
2	PB.5	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH13	A	MFP1	EADC analog input channel 13.
	SPI0_MOSI0	I/O	MFP2	SPI0 1st MOSI (Master Out, Slave In) pin.
	SPI1_MOSI	I/O	MFP3	SPI1 MOSI (Master Out, Slave In) pin.
	TK3	A	MFP4	Touch Key3.
	ACMP0_P2	A	MFP5	Comparator0 positive input pin.
	EBI_AD6	I/O	MFP7	EBI address/data bus bit 6.
3	PB.6	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH14	A	MFP1	EADC analog input channel 14.
	SPI0_MISO0	I/O	MFP2	SPI0 1st MISO (Master In, Slave Out) pin.
	SPI1_MISO	I/O	MFP3	SPI1 MISO (Master In, Slave Out) pin.
	TK4	A	MFP4	Touch Key4.
	ACMP0_P1	A	MFP5	Comparator0 positive input pin.
	EBI_AD5	I/O	MFP7	EBI address/data bus bit 5.
4	PB.7	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH15	A	MFP1	EADC analog input channel 15.
	SPI0_CLK	I/O	MFP2	SPI0 serial clock pin.
	SPI1_CLK	I/O	MFP3	SPI1 serial clock pin
	TK5	A	MFP4	Touch Key5.
	ACMP0_P0	A	MFP5	Comparator0 positive input pin.
	EBI_AD4	I/O	MFP7	EBI address/data bus bit 4.
5	nRESET	I	MFP0	External reset input: active LOW, with an internal pull-up. Set this pin low reset to initial state.
6	PD.0	I/O	MFP0	General purpose digital I/O pin.
	SPI1_I2SMCLK	O	MFP2	I2S1 master clock output pin.
	UART0_RXD	I	MFP3	Data receiver input pin for UART0.

Pin No.	Pin Name	Type	MFP*	Description
	TK6	A	MFP4	Touch Key6.
	ACMP1_N	A	MFP5	Comparator1 negative input pin.
	INT3	I	MFP8	External interrupt3 input pin.
7	AV_{SS}	P	MFP0	Ground pin for analog circuit.
8	PD.8	I/O	MFP0	General purpose digital I/O pin.
	TK8	A	MFP4	Touch Key0.
	EBI_nCS0	O	MFP7	EBI chip select 0 enable output pin.
9	PD.9	I/O	MFP0	General purpose digital I/O pin.
	TK9	A	MFP4	Touch Key8.
	ACMP1_P3	A	MFP5	Comparator1 positive input pin.
	EBI_ALE	O	MFP7	EBI address latch enable output pin.
10	PD.1	I/O	MFP0	General purpose digital I/O pin.
	PWM0_SYNC_IN	I	MFP2	PWM0 counter synchronous trigger input pin.
	UART0_TXD	O	MFP3	Data transmitter output pin for UART0.
	TK10	A	MFP4	Touch Key10.
	ACMP1_P2	A	MFP5	Comparator1 positive input pin.
	T0	I/O	MFP6	Timer0event counter input / toggle output
	EBI_nRD	O	MFP7	EBI read enable output pin.
11	PD.2	I/O	MFP0	General purpose digital I/O pin.
	STADC	I	MFP1	ADC external trigger input.
	T0_EXT	I	MFP3	Timer0 external capture input.
	TK11	A	MFP4	Touch Key11.
	ACMP1_P1	A	MFP5	Comparator1 positive input pin.
	PWM0_BRAKE0	I	MFP6	PWM0 break input 0
	EBI_nWR	O	MFP7	EBI write enable output pin.
	INT0	I	MFP8	External interrupt0 input pin.
12	PD.3	I/O	MFP0	General purpose digital I/O pin.
	T2	I/O	MFP1	Timer2 event counter input / toggle output
	T1_EXT	I	MFP3	Timer1 external capture input
	TK12	A	MFP4	Touch Key12.
	ACMP1_P0	A	MFP5	Comparator1 positive input pin.
	PWM0_BRAKE1	I	MFP6	PWM0 break input 1
	EBI_MCLK	O	MFP7	EBI external clock output pin
	INT1	I	MFP8	External interrupt1 input pin.

Pin No.	Pin Name	Type	MFP*	Description
13	V_{BAT}		MFP0	Power supply by batteries for RTC and PF.0~PF.2.
14	PF.0	I/O	MFP0	General purpose digital I/O pin.
	X32_OUT	O	MFP1	External 32.768 kHZ (low speed) crystal output pin.
	INT5	I	MFP8	External interrupt5 input pin.
15	PF.1	I/O	MFP0	General purpose digital I/O pin.
	X32_IN	I	MFP1	External 32.768 kHZ (low speed) crystal input pin.
16	PF.2	I/O	MFP0	General purpose digital I/O pin.
	TAMPER	I/O	MFP1	TAMPER detector loop pin
17	PD.12	I/O	MFP0	General purpose digital I/O pin.
	SPI2_SS	I	MFP2	SPI2 slave select pin.
	UART3_TXD	O	MFP3	Data transmitter output pin for UART3.
	PWM1_CH0	I/O	MFP6	PWM1 output/capture input.
	EBI_ADR16	O	MFP7	EBI address bus bit 16.
18	PD.13	I/O	MFP0	General purpose digital I/O pin.
	SPI2_MOSI	I/O	MFP2	SPI2 MOSI (Master Out, Slave In) pin.
	UART3_RXD	I	MFP3	Data receiver input pin for UART3.
	PWM1_CH1	I/O	MFP6	PWM1 output/capture input.
	EBI_ADR17	O	MFP7	EBI address bus bit 17.
19	PD.14	I/O	MFP0	General purpose digital I/O pin.
	SPI2_MISO	I/O	MFP2	SPI2 MISO (Master In, Slave Out) pin.
	UART3_nCTS	I	MFP3	Clear to Send input pin for UART3.
	PWM1_CH2	I/O	MFP6	PWM1 output/capture input.
	EBI_ADR18	O	MFP7	EBI address bus bit 18.
20	PD.15	I/O	MFP0	General purpose digital I/O pin.
	SPI2_CLK	I/O	MFP2	SPI2 serial clock pin.
	UART3_nRTS	O	MFP3	Request to Send output pin for UART3.
	PWM1_CH3	I/O	MFP6	PWM1 output/capture input.
	EBI_ADR19	O	MFP7	EBI address bus bit 19.
21	PD.7	I/O	MFP0	General purpose digital I/O pin.
	PWM0_SYNC_IN	I	MFP3	PWM0 counter synchronous trigger input pin.
	T1	I/O	MFP4	Timer1 event counter input / toggle output
	ACMP0_O	O	MFP5	Comparator0 output.
	PWM0_CH5	I/O	MFP6	PWM0 output/capture input.
	EBI_nRD	O	MFP7	EBI read enable output pin.

Pin No.	Pin Name	Type	MFP*	Description
22	PF.3	I/O	MFP0	General purpose digital I/O pin.
	XT1_OUT	O	MFP1	External 4~20 MHz (high speed) crystal output pin.
	I2C1_SCL	I/O	MFP3	I2C1 clock pin.
23	PF.4	I/O	MFP0	General purpose digital I/O pin.
	XT1_IN	I	MFP1	External 4~20 MHz (high speed) crystal input pin.
	I2C1_SDA	I/O	MFP3	I2C1 data input/output pin.
24	V_{SS}	A	MFP0	Ground pin for digital circuit.
25	V_{DD}	A	MFP0	Power supply for I/O ports and LDO source for internal PLL and digital function.
26	LDO_CAP	A	MFP0	LDO output pin. **Note:** This pin needs to be connected with a 1uF capacitor.
27	PC.0	I/O	MFP0	General purpose digital I/O pin.
	SPI2_CLK	I/O	MFP2	SPI2 serial clock pin.
	UART2_nCTS	I	MFP3	Clear to Send input pin for UART2.
	CAN0_TXD	I	MFP4	CAN bus transmitter input.
	PWM0_CH0	I/O	MFP6	PWM0 output/capture input.
	EBI_AD8	I/O	MFP7	EBI address/data bus bit 8.
	INT2	I	MFP8	External interrupt2 input pin.
28	PC.1	I/O	MFP0	General purpose digital I/O pin.
	CLKO	O	MFP1	Clock Out
	STDAC	I	MFP2	DAC external trigger input.
	UART2_nRTS	O	MFP3	Request to Send output pin for UART2.
	CAN0_RXD	I	MFP4	CAN bus receiver input.
	PWM0_CH1	I/O	MFP6	PWM0 output/capture input.
	EBI_AD9	I/O	MFP7	EBI address/data bus bit 9.
29	PC.2	I/O	MFP0	General purpose digital I/O pin.
	SPI2_SS	I	MFP2	SPI2 slave select pin.
	UART2_TXD	O	MFP3	Data transmitter output pin for UART2.
	ACMP1_O	O	MFP5	Comparator1 output .
	PWM0_CH2	I/O	MFP6	PWM0 output/capture input.
	EBI_AD10	I/O	MFP7	EBI address/data bus bit 10.
30	PC.3	I/O	MFP0	General purpose digital I/O pin.
	SPI2_MOSI	I/O	MFP2	SPI2 MOSI (Master Out, Slave In) pin.
	UART2_RXD	I	MFP3	Data receiver input pin for UART2.
	USB_VBUS_ST	I	MFP4	USB external VBUS regulator status pin.

Pin No.	Pin Name	Type	MFP*	Description
	PWM0_CH3	I/O	MFP6	PWM0 output/capture input.
	EBI_AD11	I/O	MFP7	EBI address/data bus bit 11.
31	PC.4	I/O	MFP0	General purpose digital I/O pin.
	SPI2_MISO	I/O	MFP2	SPI2 MISO (Master In, Slave Out) pin.
	I2C1_SCL	I/O	MFP3	I2C1 clock pin.
	USB_VBUS_EN	O	MFP4	USB external VBUS regulator enable pin.
	PWM0_CH4	I/O	MFP6	PWM0 output/capture input.
	EBI_AD12	I/O	MFP7	EBI address/data bus bit 12.
32	PC.5	I/O	MFP0	General purpose digital I/O pin.
	SPI2_I2SMCLK	O	MFP2	I2S2 master clock output pin.
	PWM0_CH5	I/O	MFP6	PWM0 output/capture input.
	EBI_AD13	I/O	MFP7	EBI address/data bus bit 13.
33	PC.6	I/O	MFP0	General purpose digital I/O pin.
	I2C1_SMBAL	O	MFP3	I2C1 SMBus SMBALTER# pin
	ACMP1_O	O	MFP5	Comparator1 output .
	PWM1_CH0	I/O	MFP6	PWM1 output/capture input.
	EBI_AD14	I/O	MFP7	EBI address/data bus bit 14.
34	PC.7	I/O	MFP0	General purpose digital I/O pin.
	I2C1_SMBSUS	O	MFP3	I2C1 SMBus SMBSUS# pin (PMBus CONTROL pin)
	PWM1_CH1	I/O	MFP6	PWM1 output/capture input.
	EBI_AD15	I/O	MFP7	EBI address/data bus bit 15.
35	PF.5	I/O	MFP0	General purpose digital I/O pin.
	ICE_CLK	I	MFP1	Serial wired debugger clock pin
36	PF.6	I/O	MFP0	General purpose digital I/O pin.
	ICE_DAT	I/O	MFP1	Serial wired debugger data pin
37	PE.8	I/O	MFP0	General purpose digital I/O pin.
	UART1_TXD	O	MFP1	Data transmitter output pin for UART1.
	SPI0_MISO1	I/O	MFP2	SPI0 2nd MISO (Master In, Slave Out) pin.
	I2C1_SCL	I/O	MFP4	I2C1 clock pin.
	SC0_PWR	O	MFP5	SmartCard power pin.
38	PE.9	I/O	MFP0	General purpose digital I/O pin.
	UART1_RXD	I	MFP1	Data receiver input pin for UART1.
	SPI0_MOSI1	I/O	MFP2	SPI0 2nd MOSI (Master Out, Slave In) pin.
	I2C1_SDA	I/O	MFP4	I2C1 data input/output pin.

Pin No.	Pin Name	Type	MFP*	Description
	SC0_RST	O	MFP5	SmartCard reset pin.
39	PE.10	I/O	MFP0	General purpose digital I/O pin.
	SPI1_MISO	I/O	MFP1	SPI1 MISO (Master In, Slave Out) pin.
	SPI0_MISO0	I/O	MFP2	SPI0 1st MISO (Master In, Slave Out) pin.
	UART1_nCTS	I	MFP3	Clear to Send input pin for UART1.
	I2C0_SMBAL	O	MFP4	I2C0 SMBus SMBALTER# pin
	SC0_DAT	I/O	MFP5	SmartCard data pin.
40	PE.11	I/O	MFP0	General purpose digital I/O pin.
	SPI1_MOSI	I/O	MFP1	SPI1 MOSI (Master Out, Slave In) pin.
	SPI0_MOSI0	I/O	MFP2	SPI0 1st MOSI (Master Out, Slave In) pin.
	UART1_nRTS	O	MFP3	Request to Send output pin for UART1.
	I2C0_SMBSUS	O	MFP4	I2C0 SMBus SMBSUS# pin (PMBus CONTROL pin)
	SC0_CLK	O	MFP5	SmartCard clock pin.
41	PE.12	I/O	MFP0	General purpose digital I/O pin.
	SPI1_SS	I/O	MFP1	SPI1 slave select pin
	SPI0_SS	I/O	MFP2	SPI0 slave select pin.
	UART1_TXD	O	MFP3	Data transmitter output pin for UART1.
	I2C0_SCL	I/O	MFP4	I2C0 clock pin.
42	PE.13	I/O	MFP0	General purpose digital I/O pin.
	SPI1_CLK	I/O	MFP1	SPI1 serial clock pin
	SPI0_CLK	I/O	MFP2	SPI0 serial clock pin.
	UART1_RXD	I	MFP3	Data receiver input pin for UART1.
	I2C0_SDA	I/O	MFP4	I2C0 data input/output pin.
43	V_{DDIO}	A	MFP0	Power supply for PE.8~PE.13.
44	USB_VBUS	A	MFP0	Power supply from USB* host or HUB.
45	USB_D-	I	MFP0	USB differential signal D-.
46	USB_D+	I	MFP0	USB differential signal D+.
	USB_ID	I	MFP0	USB identification.
48	USB_VDD33_CAP	A	MFP0	Internal power regulator output 3.3V decoupling pin. **Note:** This pin needs to be connected with a 1uF capacitor.
49	PA.3	I/O	MFP0	General purpose digital I/O pin.
	USB_VBUS_ST	I	MFP1	USB external VBUS regulator status pin.
	UART0_RXD	I	MFP2	Data receiver input pin for UART0.
	UART0_nRTS	O	MFP3	Request to Send output pin for UART0.

Pin No.	Pin Name	Type	MFP*	Description
	I2C0_SCL	I/O	MFP4	I2C0 clock pin.
	SC0_PWR	O	MFP5	SmartCard power pin.
	PWM1_CH2	I/O	MFP6	PWM1 output/capture input.
	EBI_AD3	I/O	MFP7	EBI address/data bus bit 3.
50	PA.2	I/O	MFP0	General purpose digital I/O pin.
	USB_VBUS_EN	O	MFP1	USB external VBUS regulator enable pin.
	UART0_TXD	O	MFP2	Data transmitter output pin for UART0.
	UART0_nCTS	I	MFP3	Clear to Send input pin for UART0.
	I2C0_SDA	I/O	MFP4	I2C0 data input/output pin.
	SC0_RST	O	MFP5	SmartCard reset pin.
	PWM1_CH3	I/O	MFP6	PWM1 output/capture input.
	EBI_AD2	I/O	MFP7	EBI address/data bus bit 2.
51	PA.1	I/O	MFP0	General purpose digital I/O pin.
	UART1_nRTS	O	MFP1	Request to Send output pin for UART1.
	UART1_RXD	I	MFP3	Data receiver input pin for UART1.
	CAN0_TXD	I	MFP4	CAN bus transmitter input.
	SC0_DAT	I/O	MFP5	SmartCard data pin.
	PWM1_CH4	I/O	MFP6	PWM1 output/capture input.
	EBI_AD1	I/O	MFP7	EBI address/data bus bit 1.
52	PA.0	I/O	MFP0	General purpose digital I/O pin.
	UART1_nCTS	I	MFP1	Clear to Send input pin for UART1.
	UART1_TXD	O	MFP3	Data transmitter output pin for UART1.
	CAN0_RXD	I	MFP4	CAN bus receiver input.
	SC0_CLK	O	MFP5	SmartCard clock pin.
	PWM1_CH5	I/O	MFP6	PWM1 output/capture input.
	EBI_AD0	I/O	MFP7	EBI address/data bus bit 0.
	INT0	I	MFP8	External interrupt0 input pin.
53	V_{SS}	A	MFP0	Ground pin for digital circuit.
54	V_{DD}	A	MFP0	Power supply for I/O ports and LDO source for internal PLL and digital function.
55	AV_{DD}	A	MFP0	Power supply for internal analog circuit.
56	V_{REF}	I	MFP0	Voltage reference input for ADC. **Note:** This pin needs to be connected with a 1uF capacitor.
57	PB.0	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH0	A	MFP1	EADC analog input.

Pin No.	Pin Name	Type	MFP*	Description
	SPI0_MOSI1	I/O	MFP2	SPI0 2nd MOSI (Master Out, Slave In) pin.
	UART2_RXD	I	MFP3	Data receiver input pin for UART2.
	T2	I/O	MFP4	Timer2 event counter input / toggle output
	DAC	A	MFP5	DAC analog output
	EBI_nWRL	O	MFP7	EBI low byte write enable output pin.
	INT1	I	MFP8	External interrupt1 input pin.
58	PB.1	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH1	A	MFP1	EADC analog input channel 1.
	SPI0_MISO1	I/O	MFP2	SPI0 2nd MISO (Master In, Slave Out) pin.
	UART2_TXD	O	MFP3	Data transmitter output pin for UART2.
	T3	I/O	MFP4	Timer3 event counter input / toggle output
	SC0_RST	O	MFP5	SmartCard reset pin.
	PWM0_SYNC_OUT	O	MFP6	PWM0 counter synchronous trigger output pin.
	EBI_nWRH	O	MFP7	EBI high byte write enable output pin
59	PB.2	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH2	A	MFP1	EADC analog input channel 2.
	SPI0_CLK	I/O	MFP2	SPI0 serial clock pin.
	SPI1_CLK	I/O	MFP3	SPI1 serial clock pin
	UART1_RXD	I	MFP4	Data receiver input pin for UART1.
	SC0_CD	I	MFP5	SmartCard card detect pin.
60	PB.3	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH3	A	MFP1	EADC analog input channel 3.
	SPI0_MISO0	I/O	MFP2	SPI0 1st MISO (Master In, Slave Out) pin.
	SPI1_MISO	I/O	MFP3	SPI1 MISO (Master In, Slave Out) pin.
	UART1_TXD	O	MFP4	Data transmitter output pin for UART1.
61	PB.4	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH4	A	MFP1	EADC analog input channel 4.
	SPI0_SS	I/O	MFP2	SPI0 slave select pin.
	SPI1_SS	I/O	MFP3	SPI1 slave select pin
	UART1_nCTS	I	MFP4	Clear to Send input pin for UART1.
	ACMP0_N	A	MFP5	Comparator0 negative input pin.
	EBI_AD7	I/O	MFP7	EBI address/data bus bit 7.
62	PB.8	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH5	A	MFP1	EADC analog input channel 5.

Pin No.	Pin Name	Type	MFP*	Description
	UART1_nRTS	O	MFP4	Request to Send output pin for UART1.
	PWM0_CH2	I/O	MFP6	PWM0 output/capture input.
63	PB.11	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH8	A	MFP1	EADC analog input channel 8.
	TK0	A	MFP4	Touch Key0.
64	PB.12	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH9	A	MFP1	EADC analog input channel 9.
	TK1	A	MFP4	Touch Key1.

4.3.3 M4TK CAN Series(CAN+USB) LQFP100 Pin Description

MFP* = Multi-function pin. (Refer to section SYS_GPx_MFPL and SYS_GPx_MFPH)

PA.0 MFP0 means SYS_GPA_MFPL[3:0] = 0x0.

PA.9 MFP5 means SYS_GPA_MFPH[7:4] = 0x5.

Pin No.	Pin Name	Type	MFP*	Description
1	PB.13	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH10	A	MFP1	EADC analog input channel 10.
2	PB.14	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH11	A	MFP1	EADC analog input channel 11.
3	PB.15	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH12	A	MFP1	EADC analog input channel 12.
	TK2	A	MFP4	Touch Key2.
	ACMP0_P3	A	MFP5	Comparator0 positive input pin.
	EBI_nCS1	O	MFP7	EBI chip select 1 enable output pin.
4	PB.5	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH13	A	MFP1	EADC analog input channel 13.
	SPI0_MOSI0	I/O	MFP2	SPI0 1st MOSI (Master Out, Slave In) pin.
	SPI1_MOSI	I/O	MFP3	SPI1 MOSI (Master Out, Slave In) pin.
	TK3	A	MFP4	Touch Key3.
	ACMP0_P2	A	MFP5	Comparator0 positive input pin.
	EBI_AD6	I/O	MFP7	EBI address/data bus bit 6.
5	PB.6	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH14	A	MFP1	EADC analog input channel 14.
	SPI0_MISO0	I/O	MFP2	SPI0 1st MISO (Master In, Slave Out) pin.
	SPI1_MISO	I/O	MFP3	SPI1 MISO (Master In, Slave Out) pin.
	TK4	A	MFP4	Touch Key4.
	ACMP0_P1	A	MFP5	Comparator0 positive input pin.
	EBI_AD5	I/O	MFP7	EBI address/data bus bit 5.
6	PB.7	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH15	A	MFP1	EADC analog input channel 15.
	SPI0_CLK	I/O	MFP2	SPI0 serial clock pin.
	SPI1_CLK	I/O	MFP3	SPI1 serial clock pin
	TK5	A	MFP4	Touch Key5.
	ACMP0_P0	A	MFP5	Comparator0 positive input pin.
	EBI_AD4	I/O	MFP7	EBI address/data bus bit 4.
7	nRESET	I	MFP0	External reset input: active LOW, with an internal pull-up.

Pin No.	Pin Name	Type	MFP*	Description
				Set this pin low reset to initial state.
8	PD.0	I/O	MFP0	General purpose digital I/O pin.
	SPI1_I2SMCLK	O	MFP2	I2S1 master clock output pin.
	UART0_RXD	I	MFP3	Data receiver input pin for UART0.
	TK6	A	MFP4	Touch Key6.
	ACMP1_N	A	MFP5	Comparator1 negative input pin.
	INT3	I	MFP8	External interrupt3 input pin.
9	AV_{SS}	P	MFP0	Ground pin for analog circuit.
10	V_{DD}	A	MFP0	Power supply for I/O ports and LDO source for internal PLL and digital function.
11	V_{SS}	A	MFP0	Ground pin for digital circuit.
12	PC.8	I/O	MFP0	General purpose digital I/O pin.
	TK7	A	MFP4	Touch Key7.
13	PD.8	I/O	MFP0	General purpose digital I/O pin.
	TK8	A	MFP4	Touch Key0.
	EBI_nCS0	O	MFP7	EBI chip select 0 enable output pin.
14	PD.9	I/O	MFP0	General purpose digital I/O pin.
	TK9	A	MFP4	Touch Key8.
	ACMP1_P3	A	MFP5	Comparator1 positive input pin.
	EBI_ALE	O	MFP7	EBI address latch enable output pin.
15	PD.1	I/O	MFP0	General purpose digital I/O pin.
	PWM0_SYNC_IN	I	MFP2	PWM0 counter synchronous trigger input pin.
	UART0_TXD	O	MFP3	Data transmitter output pin for UART0.
	TK10	A	MFP4	Touch Key10.
	ACMP1_P2	A	MFP5	Comparator1 positive input pin.
	T0	I/O	MFP6	Timer0event counter input / toggle output
	EBI_nRD	O	MFP7	EBI read enable output pin.
16	PD.2	I/O	MFP0	General purpose digital I/O pin.
	STADC	I	MFP1	ADC external trigger input.
	T0_EXT	I	MFP3	Timer0 external capture input.
	TK11	A	MFP4	Touch Key11.
	ACMP1_P1	A	MFP5	Comparator1 positive input pin.
	PWM0_BRAKE0	I	MFP6	PWM0 break input 0
	EBI_nWR	O	MFP7	EBI write enable output pin.
	INT0	I	MFP8	External interrupt0 input pin.

Pin No.	Pin Name	Type	MFP*	Description
17	PD.3	I/O	MFP0	General purpose digital I/O pin.
	T2	I/O	MFP1	Timer2 event counter input / toggle output
	T1_EXT	I	MFP3	Timer1 external capture input
	TK12	A	MFP4	Touch Key12.
	ACMP1_P0	A	MFP5	Comparator1 positive input pin.
	PWM0_BRAKE1	I	MFP6	PWM0 break input 1
	EBI_MCLK	O	MFP7	EBI external clock output pin
	INT1	I	MFP8	External interrupt1 input pin.
18	PD.4	I/O	MFP0	General purpose digital I/O pin.
	SPI1_CLK	I/O	MFP2	SPI1 serial clock pin
	I2C0_SDA	I/O	MFP3	I2C0 data input/output pin.
	TK13	A	MFP4	Touch Key13.
	PWM0_BRAKE0	I	MFP5	PWM0 break input 0
	T0	I/O	MFP6	Timer0event counter input / toggle output
19	PD.5	I/O	MFP0	General purpose digital I/O pin.
	CLKO	O	MFP1	Clock Out
	SPI1_MISO	I/O	MFP2	SPI1 MISO (Master In, Slave Out) pin.
	I2C0_SCL	I/O	MFP3	I2C0 clock pin.
	TK14	A	MFP4	Touch Key14.
	PWM0_BRAKE1	I	MFP5	PWM0 break input 1
	T1	I/O	MFP6	Timer1 event counter input / toggle output
20	PE.3	I/O	MFP0	General purpose digital I/O pin.
	SPI1_MOSI	I/O	MFP2	SPI1 MOSI (Master Out, Slave In) pin.
	TK15	A	MFP4	Touch Key15.
	PWM0_CH3	I/O	MFP6	PWM0 output/capture input.
21	PD.6	I/O	MFP0	General purpose digital I/O pin.
	CLKO	O	MFP1	Clock Out
	SPI1_SS	I/O	MFP2	SPI1 slave select pin
	UART0_RXD	I	MFP3	Data receiver input pin for UART0.
	TK16	A	MFP4	Touch Key16.
	ACMP0_O	O	MFP5	Comparator0 output.
	PWM0_CH5	I/O	MFP6	PWM0 output/capture input.
	EBI_nWR	O	MFP7	EBI write enable output pin.
22	V_{BAT}	A	MFP0	Power supply by batteries for RTC and PF.0~PF.2.

Pin No.	Pin Name	Type	MFP*	Description
23	PF.0	I/O	MFP0	General purpose digital I/O pin.
	X32_OUT	O	MFP1	External 32.768 kHZ (low speed) crystal output pin.
	INT5	I	MFP8	External interrupt5 input pin.
24	PF.1	I/O	MFP0	General purpose digital I/O pin.
	X32_IN	I	MFP1	External 32.768 kHZ (low speed) crystal input pin.
25	PF.2	I/O	MFP0	General purpose digital I/O pin.
	TAMPER	I/O	MFP1	TAMPER detector loop pin
26	PD.10	I/O	MFP0	General purpose digital I/O pin.
	T2	I/O	MFP4	Timer2 event counter input / toggle output
27	PD.11	I/O	MFP0	General purpose digital I/O pin.
	T3	I/O	MFP4	Timer3 event counter input / toggle output.
28	PD.12	I/O	MFP0	General purpose digital I/O pin.
	SPI2_SS	I	MFP2	SPI2 slave select pin.
	UART3_TXD	O	MFP3	Data transmitter output pin for UART3.
	PWM1_CH0	I/O	MFP6	PWM1 output/capture input.
	EBI_ADR16	O	MFP7	EBI address bus bit 16.
29	PD.13	I/O	MFP0	General purpose digital I/O pin.
	SPI2_MOSI	I/O	MFP2	SPI2 MOSI (Master Out, Slave In) pin.
	UART3_RXD	I	MFP3	Data receiver input pin for UART3.
	PWM1_CH1	I/O	MFP6	PWM1 output/capture input.
	EBI_ADR17	O	MFP7	EBI address bus bit 17.
30	PD.14	I/O	MFP0	General purpose digital I/O pin.
	SPI2_MISO	I/O	MFP2	SPI2 MISO (Master In, Slave Out) pin.
	UART3_nCTS	I	MFP3	Clear to Send input pin for UART3.
	PWM1_CH2	I/O	MFP6	PWM1 output/capture input.
	EBI_ADR18	O	MFP7	EBI address bus bit 18.
31	PD.15	I/O	MFP0	General purpose digital I/O pin.
	SPI2_CLK	I/O	MFP2	SPI2 serial clock pin.
	UART3_nRTS	O	MFP3	Request to Send output pin for UART3.
	PWM1_CH3	I/O	MFP6	PWM1 output/capture input.
	EBI_ADR19	O	MFP7	EBI address bus bit 19.
32	PD.7	I/O	MFP0	General purpose digital I/O pin.
	PWM0_SYNC_IN	I	MFP3	PWM0 counter synchronous trigger input pin.
	T1	I/O	MFP4	Timer1 event counter input / toggle output

Pin No.	Pin Name	Type	MFP*	Description
	ACMP0_O	O	MFP5	Comparator0 output.
	PWM0_CH5	I/O	MFP6	PWM0 output/capture input.
	EBI_nRD	O	MFP7	EBI read enable output pin.
33	PF.3	I/O	MFP0	General purpose digital I/O pin.
	XT1_OUT	O	MFP1	External 4~20 MHz (high speed) crystal output pin.
	I2C1_SCL	I/O	MFP3	I2C1 clock pin.
34	PF.4	I/O	MFP0	General purpose digital I/O pin.
	XT1_IN	I	MFP1	External 4~20 MHz (high speed) crystal input pin.
	I2C1_SDA	I/O	MFP3	I2C1 data input/output pin.
35	V_{SS}	A	MFP0	Ground pin for digital circuit.
36	V_{DD}	A	MFP0	Power supply for I/O ports and LDO source for internal PLL and digital function.
37	LDO_CAP	A	MFP0	LDO output pin. **Note:** This pin needs to be connected with a 1uF capacitor.
38	PC.9	I/O	MFP0	General purpose digital I/O pin.
	SPI2_I2SMCLK	O	MFP2	I2S2 master clock output pin.
	PWM1_CH0	I/O	MFP6	PWM1 output/capture input.
39	PC.10	I/O	MFP0	General purpose digital I/O pin.
	SPI2_MOSI	I/O	MFP2	SPI2 MOSI (Master Out, Slave In) pin.
	PWM1_CH1	I/O	MFP6	PWM1 output/capture input.
40	PC.11	I/O	MFP0	General purpose digital I/O pin.
	SPI2_MISO	I/O	MFP2	SPI2 MISO (Master In, Slave Out) pin.
	PWM1_CH2	I/O	MFP6	PWM1 output/capture input.
41	PC.12	I/O	MFP0	General purpose digital I/O pin.
	SPI2_CLK	I/O	MFP2	SPI2 serial clock pin.
	PWM1_CH3	I/O	MFP6	PWM1 output/capture input.
42	PC.13	I/O	MFP0	General purpose digital I/O pin.
	SPI2_SS	I	MFP2	SPI2 slave select pin.
	PWM1_CH4	I/O	MFP6	PWM1 output/capture input.
43	PC.14	I/O	MFP0	General purpose digital I/O pin.
	PWM1_CH5	I/O	MFP6	PWM1 output/capture input.
44	PC.0	I/O	MFP0	General purpose digital I/O pin.
	SPI2_CLK	I/O	MFP2	SPI2 serial clock pin.
	UART2_nCTS	I	MFP3	Clear to Send input pin for UART2.
	CAN0_TXD	I	MFP4	CAN bus transmitter input.

Pin No.	Pin Name	Type	MFP*	Description
	PWM0_CH0	I/O	MFP6	PWM0 output/capture input.
	EBI_AD8	I/O	MFP7	EBI address/data bus bit 8.
	INT2	I	MFP8	External interrupt2 input pin.
45	PC.1	I/O	MFP0	General purpose digital I/O pin.
	CLKO	O	MFP1	Clock Out
	STDAC	I	MFP2	DAC external trigger input.
	UART2_nRTS	O	MFP3	Request to Send output pin for UART2.
	CAN0_RXD	I	MFP4	CAN bus receiver input.
	PWM0_CH1	I/O	MFP6	PWM0 output/capture input.
	EBI_AD9	I/O	MFP7	EBI address/data bus bit 9.
46	PC.2	I/O	MFP0	General purpose digital I/O pin.
	SPI2_SS	I	MFP2	SPI2 slave select pin.
	UART2_TXD	O	MFP3	Data transmitter output pin for UART2.
	ACMP1_O	O	MFP5	Comparator1 output .
	PWM0_CH2	I/O	MFP6	PWM0 output/capture input.
	EBI_AD10	I/O	MFP7	EBI address/data bus bit 10.
47	PC.3	I/O	MFP0	General purpose digital I/O pin.
	SPI2_MOSI	I/O	MFP2	SPI2 MOSI (Master Out, Slave In) pin.
	UART2_RXD	I	MFP3	Data receiver input pin for UART2.
	USB_VBUS_ST	I	MFP4	USB external VBUS regulator status pin.
	PWM0_CH3	I/O	MFP6	PWM0 output/capture input.
	EBI_AD11	I/O	MFP7	EBI address/data bus bit 11.
48	PC.4	I/O	MFP0	General purpose digital I/O pin.
	SPI2_MISO	I/O	MFP2	SPI2 MISO (Master In, Slave Out) pin.
	I2C1_SCL	I/O	MFP3	I2C1 clock pin.
	USB_VBUS_EN	O	MFP4	USB external VBUS regulator enable pin.
	PWM0_CH4	I/O	MFP6	PWM0 output/capture input.
	EBI_AD12	I/O	MFP7	EBI address/data bus bit 12.
49	PE.0	I/O	MFP0	General purpose digital I/O pin.
	SPI2_CLK	I/O	MFP2	SPI2 serial clock pin.
	I2C1_SDA	I/O	MFP3	I2C1 data input/output pin.
	T2_EXT	I	MFP4	Timer2 external capture input
	SC0_CD	I	MFP5	SmartCard card detect pin.
	PWM0_CH0	I/O	MFP6	PWM0 output/capture input.

Pin No.	Pin Name	Type	MFP*	Description
	EBI_nCS1	O	MFP7	EBI chip select 1 enable output pin.
	INT4	I	MFP8	External interrupt4 input pin.
50	PC.5	I/O	MFP0	General purpose digital I/O pin.
	SPI2_I2SMCLK	O	MFP2	I2S2 master clock output pin.
	PWM0_CH5	I/O	MFP6	PWM0 output/capture input.
	EBI_AD13	I/O	MFP7	EBI address/data bus bit 13.
51	PC.6	I/O	MFP0	General purpose digital I/O pin.
	I2C1_SMBAL	O	MFP3	I2C1 SMBus SMBALTER# pin
	ACMP1_O	O	MFP5	Comparator1 output .
	PWM1_CH0	I/O	MFP6	PWM1 output/capture input.
	EBI_AD14	I/O	MFP7	EBI address/data bus bit 14.
52	PC.7	I/O	MFP0	General purpose digital I/O pin.
	I2C1_SMBSUS	O	MFP3	I2C1 SMBus SMBSUS# pin (PMBus CONTROL pin)
	PWM1_CH1	I/O	MFP6	PWM1 output/capture input.
	EBI_AD15	I/O	MFP7	EBI address/data bus bit 15.
53	PE.4	I/O	MFP0	General purpose digital I/O pin.
	I2C1_SCL	I/O	MFP3	I2C1 clock pin.
	SC0_PWR	O	MFP5	SmartCard power pin.
	PWM1_BRAKE0	I	MFP6	PWM1 break input 0
	EBI_nCS0	O	MFP7	EBI chip select 0 enable output pin.
	INT0	I	MFP8	External interrupt0 input pin.
54	PE.5	I/O	MFP0	General purpose digital I/O pin.
	I2C1_SDA	I/O	MFP3	I2C1 data input/output pin.
	SC0_RST	O	MFP5	SmartCard reset pin.
	PWM1_BRAKE1	I	MFP6	PWM1 break input 1
	EBI_ALE	O	MFP7	EBI address latch enable output pin.
	INT1	I	MFP8	External interrupt1 input pin.
55	PF.5	I/O	MFP0	General purpose digital I/O pin.
	ICE_CLK	I	MFP1	Serial wired debugger clock pin
56	PF.6	I/O	MFP0	General purpose digital I/O pin.
	ICE_DAT	I/O	MFP1	Serial wired debugger data pin
57	PA.8	I/O	MFP0	General purpose digital I/O pin.
	UART3_TXD	O	MFP3	Data transmitter output pin for UART3.
58	PA.9	I/O	MFP0	General purpose digital I/O pin.

Pin No.	Pin Name	Type	MFP*	Description
	UART3_RXD	I	MFP3	Data receiver input pin for UART3.
59	PA.7	I/O	MFP0	General purpose digital I/O pin.
	SPI1_CLK	I/O	MFP2	SPI1 serial clock pin
	T0_EXT	I	MFP3	Timer0 external capture input.
	EBI_AD7	I/O	MFP7	EBI address/data bus bit 7.
60	PA.6	I/O	MFP0	General purpose digital I/O pin.
	SPI1_MISO	I/O	MFP2	SPI1 MISO (Master In, Slave Out) pin.
	T1_EXT	I	MFP3	Timer1 external capture input
	EBI_AD6	I/O	MFP7	EBI address/data bus bit 6.
61	PA.5	I/O	MFP0	General purpose digital I/O pin.
	SPI1_MOSI	I/O	MFP2	SPI1 MOSI (Master Out, Slave In) pin.
	T2_EXT	I	MFP3	Timer2 external capture input
	EBI_AD5	I/O	MFP7	EBI address/data bus bit 5.
62	PA.4	I/O	MFP0	General purpose digital I/O pin.
	SPI1_SS	I/O	MFP2	SPI1 slave select pin
	EBI_AD4	I/O	MFP7	EBI address/data bus bit 4.
63	V_{SS}	A	MFP0	Ground pin for digital circuit.
64	V_{DD}	A	MFP0	Power supply for I/O ports and LDO source for internal PLL and digital function.
65	PE.1	I/O	MFP0	General purpose digital I/O pin.
	T3_EXT	I	MFP3	Timer3 external capture input
	SC0_CD	I	MFP5	SmartCard card detect pin.
	PWM0_CH1	I/O	MFP6	PWM0 output/capture input.
66	PE.8	I/O	MFP0	General purpose digital I/O pin.
	UART1_TXD	O	MFP1	Data transmitter output pin for UART1.
	SPI0_MISO1	I/O	MFP2	SPI0 2nd MISO (Master In, Slave Out) pin.
	I2C1_SCL	I/O	MFP4	I2C1 clock pin.
	SC0_PWR	O	MFP5	SmartCard power pin.
67	PE.9	I/O	MFP0	General purpose digital I/O pin.
	UART1_RXD	I	MFP1	Data receiver input pin for UART1.
	SPI0_MOSI1	I/O	MFP2	SPI0 2nd MOSI (Master Out, Slave In) pin.
	I2C1_SDA	I/O	MFP4	I2C1 data input/output pin.
	SC0_RST	O	MFP5	SmartCard reset pin.
68	PE.10	I/O	MFP0	General purpose digital I/O pin.
	SPI1_MISO	I/O	MFP1	SPI1 MISO (Master In, Slave Out) pin.

Pin No.	Pin Name	Type	MFP*	Description
	SPI0_MISO0	I/O	MFP2	SPI0 1st MISO (Master In, Slave Out) pin.
	UART1_nCTS	I	MFP3	Clear to Send input pin for UART1.
	I2C0_SMBAL	O	MFP4	I2C0 SMBus SMBALTER# pin
	SC0_DAT	I/O	MFP5	SmartCard data pin.
69	PE.11	I/O	MFP0	General purpose digital I/O pin.
	SPI1_MOSI	I/O	MFP1	SPI1 MOSI (Master Out, Slave In) pin.
	SPI0_MOSI0	I/O	MFP2	SPI0 1st MOSI (Master Out, Slave In) pin.
	UART1_nRTS	O	MFP3	Request to Send output pin for UART1.
	I2C0_SMBSUS	O	MFP4	I2C0 SMBus SMBSUS# pin (PMBus CONTROL pin)
	SC0_CLK	O	MFP5	SmartCard clock pin.
70	PE.12	I/O	MFP0	General purpose digital I/O pin.
	SPI1_SS	I/O	MFP1	SPI1 slave select pin
	SPI0_SS	I/O	MFP2	SPI0 slave select pin.
	UART1_TXD	O	MFP3	Data transmitter output pin for UART1.
	I2C0_SCL	I/O	MFP4	I2C0 clock pin.
71	PE.13	I/O	MFP0	General purpose digital I/O pin.
	SPI1_CLK	I/O	MFP1	SPI1 serial clock pin
	SPI0_CLK	I/O	MFP2	SPI0 serial clock pin.
	UART1_RXD	I	MFP3	Data receiver input pin for UART1.
	I2C0_SDA	I/O	MFP4	I2C0 data input/output pin.
72	V_{DDIO}	A	MFP0	Power supply for PE.8~PE.13.
73	USB_VBUS	A	MFP0	Power supply from USB* host or HUB.
74	USB_D-	I	MFP0	USB differential signal D-.
75	USB_D+	I	MFP0	USB differential signal D+.
	USB_ID	I	MFP0	USB identification.
77	USB_VDD33_CAP	A	MFP0	Internal power regulator output 3.3V decoupling pin. **Note:** This pin needs to be connected with a 1uF capacitor.
78	PE.2	I/O	MFP0	General purpose digital I/O pin.
	PWM1_CH1	I/O	MFP6	PWM1 output/capture input.
79	PA.3	I/O	MFP0	General purpose digital I/O pin.
	USB_VBUS_ST	I	MFP1	USB external VBUS regulator status pin.
	UART0_RXD	I	MFP2	Data receiver input pin for UART0.
	UART0_nRTS	O	MFP3	Request to Send output pin for UART0.
	I2C0_SCL	I/O	MFP4	I2C0 clock pin.

Pin No.	Pin Name	Type	MFP*	Description
	SC0_PWR	O	MFP5	SmartCard power pin.
	PWM1_CH2	I/O	MFP6	PWM1 output/capture input.
	EBI_AD3	I/O	MFP7	EBI address/data bus bit 3.
80	PA.2	I/O	MFP0	General purpose digital I/O pin.
	USB_VBUS_EN	O	MFP1	USB external VBUS regulator enable pin.
	UART0_TXD	O	MFP2	Data transmitter output pin for UART0.
	UART0_nCTS	I	MFP3	Clear to Send input pin for UART0.
	I2C0_SDA	I/O	MFP4	I2C0 data input/output pin.
	SC0_RST	O	MFP5	SmartCard reset pin.
	PWM1_CH3	I/O	MFP6	PWM1 output/capture input.
	EBI_AD2	I/O	MFP7	EBI address/data bus bit 2.
81	PA.1	I/O	MFP0	General purpose digital I/O pin.
	UART1_nRTS	O	MFP1	Request to Send output pin for UART1.
	UART1_RXD	I	MFP3	Data receiver input pin for UART1.
	CAN0_TXD	I	MFP4	CAN bus transmitter input.
	SC0_DAT	I/O	MFP5	SmartCard data pin.
	PWM1_CH4	I/O	MFP6	PWM1 output/capture input.
	EBI_AD1	I/O	MFP7	EBI address/data bus bit 1.
82	PA.0	I/O	MFP0	General purpose digital I/O pin.
	UART1_nCTS	I	MFP1	Clear to Send input pin for UART1.
	UART1_TXD	O	MFP3	Data transmitter output pin for UART1.
	CAN0_RXD	I	MFP4	CAN bus receiver input.
	SC0_CLK	O	MFP5	SmartCard clock pin.
	PWM1_CH5	I/O	MFP6	PWM1 output/capture input.
	EBI_AD0	I/O	MFP7	EBI address/data bus bit 0.
	INT0	I	MFP8	External interrupt0 input pin.
83	PA.12	I/O	MFP0	General purpose digital I/O pin.
	SPI1_I2SMCLK	O	MFP2	I2S1 master clock output pin.
	CAN0_TXD	I	MFP4	CAN bus transmitter input.
84	PA.13	I/O	MFP0	General purpose digital I/O pin.
	CAN0_RXD	I	MFP4	CAN bus receiver input.
85	PA.14	I/O	MFP0	General purpose digital I/O pin.
	UART2_nCTS	I	MFP3	Clear to Send input pin for UART2.
	I2C0_SMBAL	O	MFP4	I2C0 SMBus SMBALTER# pin

Pin No.	Pin Name	Type	MFP*	Description
86	PA.15	I/O	MFP0	General purpose digital I/O pin.
	UART2_nRTS	O	MFP3	Request to Send output pin for UART2.
	I2C0_SMBSUS	O	MFP4	I2C0 SMBus SMBSUS# pin (PMBus CONTROL pin)
87	V_{SS}	A	MFP0	Ground pin for digital circuit.
88	V_{DD}	A	MFP0	Power supply for I/O ports and LDO source for internal PLL and digital function.
89	AV_{DD}	A	MFP0	Power supply for internal analog circuit.
90	V_{REF}	I	MFP0	Voltage reference input for ADC. **Note:** This pin needs to be connected with a 1uF capacitor.
91	PB.0	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH0	A	MFP1	EADC analog input.
	SPI0_MOSI1	I/O	MFP2	SPI0 2nd MOSI (Master Out, Slave In) pin.
	UART2_RXD	I	MFP3	Data receiver input pin for UART2.
	T2	I/O	MFP4	Timer2 event counter input / toggle output
	DAC	A	MFP5	DAC analog output
	EBI_nWRL	O	MFP7	EBI low byte write enable output pin.
	INT1	I	MFP8	External interrupt1 input pin.
92	PB.1	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH1	A	MFP1	EADC analog input channel 1.
	SPI0_MISO1	I/O	MFP2	SPI0 2nd MISO (Master In, Slave Out) pin.
	UART2_TXD	O	MFP3	Data transmitter output pin for UART2.
	T3	I/O	MFP4	Timer3 event counter input / toggle output
	SC0_RST	O	MFP5	SmartCard reset pin.
	PWM0_SYNC_OUT	O	MFP6	PWM0 counter synchronous trigger output pin.
	EBI_nWRH	O	MFP7	EBI high byte write enable output pin
93	PB.2	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH2	A	MFP1	EADC analog input channel 2.
	SPI0_CLK	I/O	MFP2	SPI0 serial clock pin.
	SPI1_CLK	I/O	MFP3	SPI1 serial clock pin
	UART1_RXD	I	MFP4	Data receiver input pin for UART1.
	SC0_CD	I	MFP5	SmartCard card detect pin.
94	PB.3	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH3	A	MFP1	EADC analog input channel 3.
	SPI0_MISO0	I/O	MFP2	SPI0 1st MISO (Master In, Slave Out) pin.
	SPI1_MISO	I/O	MFP3	SPI1 MISO (Master In, Slave Out) pin.

Pin No.	Pin Name	Type	MFP*	Description
	UART1_TXD	O	MFP4	Data transmitter output pin for UART1.
95	PB.4	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH4	A	MFP1	EADC analog input channel 4.
	SPI0_SS	I/O	MFP2	SPI0 slave select pin.
	SPI1_SS	I/O	MFP3	SPI1 slave select pin
	UART1_nCTS	I	MFP4	Clear to Send input pin for UART1.
	ACMP0_N	A	MFP5	Comparator0 negative input pin.
	EBI_AD7	I/O	MFP7	EBI address/data bus bit 7.
96	PB.8	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH5	A	MFP1	EADC analog input channel 5.
	UART1_nRTS	O	MFP4	Request to Send output pin for UART1.
	PWM0_CH2	I/O	MFP6	PWM0 output/capture input.
97	PB.9	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH6	A	MFP1	EADC analog input channel 6.
98	PB.10	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH7	A	MFP1	EADC analog input channel 7.
99	PB.11	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH8	A	MFP1	EADC analog input channel 8.
	TK0	A	MFP4	Touch Key0.
100	PB.12	I/O	MFP0	General purpose digital I/O pin.
	EADC_CH9	A	MFP1	EADC analog input channel 9.
	TK1	A	MFP4	Touch Key1.

4.3.4 GPIO Multi-function Pin Summary

MFP* = Multi-function pin. (Refer to section SYS_GPx_MFPL and SYS_GPx_MFPH)

PA.0 MFP0 means SYS_GPA_MFPL[3:0]=0x0.

PA.9 MFP5 means SYS_GPA_MFPH[7:4]=0x5.

Group	Pin Name	MFP*	Type	Description
ACMP0	ACMP0_N	MFP5	A	Comparator0 negative input pin.
	ACMP0_O	MFP5	O	Comparator0 output.
	ACMP0_O	MFP5	O	Comparator0 output.
	ACMP0_P0	MFP5	A	Comparator0 positive input pin.
	ACMP0_P1	MFP5	A	Comparator0 positive input pin.
	ACMP0_P2	MFP5	A	Comparator0 positive input pin.
	ACMP0_P3	MFP5	A	Comparator0 positive input pin.
ACMP1	ACMP1_N	MFP5	A	Comparator1 negative input pin.
	ACMP1_O	MFP5	O	Comparator1 output .
	ACMP1_O	MFP5	O	Comparator1 output .
	ACMP1_P0	MFP5	A	Comparator1 positive input pin.
	ACMP1_P1	MFP5	A	Comparator1 positive input pin.
	ACMP1_P2	MFP5	A	Comparator1 positive input pin.
	ACMP1_P3	MFP5	A	Comparator1 positive input pin.
EADC	EADC_CH0	MFP1	A	ADC0 analog input.
	EADC_CH1	MFP1	A	ADC1 analog input.
	EADC_CH2	MFP1	A	ADC2 analog input.
	EADC_CH3	MFP1	A	ADC3 analog input.
	EADC_CH4	MFP1	A	ADC4 analog input.
	EADC_CH5	MFP1	A	ADC5 analog input.
	EADC_CH6	MFP1	A	ADC6 analog input.
	EADC_CH7	MFP1	A	ADC7 analog input.
	EADC_CH8	MFP1	A	ADC8 analog input.
	EADC_CH9	MFP1	A	ADC9 analog input.
	EADC_CH10	MFP1	A	ADC10 analog input.
	EADC_CH11	MFP1	A	ADC11 analog input.
	EADC_CH12	MFP1	A	ADC12 analog input.
	EADC_CH13	MFP1	A	ADC13 analog input.
	EADC_CH14	MFP1	A	ADC14 analog input.
	EADC_CH15	MFP1	A	ADC15 analog input.
	STADC	MFP1	I	ADC external trigger input.

Group	Pin Name	MFP*	Type	Description
CAN0	CAN0_RXD	MFP4	I	CAN bus receiver input.
	CAN0_RXD	MFP4	I	CAN bus receiver input.
	CAN0_RXD	MFP4	I	CAN bus receiver input.
	CAN0_TXD	MFP4	I	CAN bus transmitter input.
	CAN0_TXD	MFP4	I	CAN bus transmitter input.
	CAN0_TXD	MFP4	I	CAN bus transmitter input.
CLKO	CLKO	MFP1	O	Clock Out
	CLKO	MFP1	O	Clock Out
	CLKO	MFP1	O	Clock Out
DAC	DAC	MFP5	A	DAC analog output
	STDAC	MFP2	I	DAC external trigger input.
EBI	EBI_AD0	MFP7	I/O	EBI address/data bus bit 0.
	EBI_AD1	MFP7	I/O	EBI address/data bus bit 1.
	EBI_AD2	MFP7	I/O	EBI address/data bus bit 2.
	EBI_AD3	MFP7	I/O	EBI address/data bus bit 3.
	EBI_AD4	MFP7	I/O	EBI address/data bus bit 4.
	EBI_AD4	MFP7	I/O	EBI address/data bus bit 4.
	EBI_AD5	MFP7	I/O	EBI address/data bus bit 5.
	EBI_AD5	MFP7	I/O	EBI address/data bus bit 5.
	EBI_AD6	MFP7	I/O	EBI address/data bus bit 6.
	EBI_AD6	MFP7	I/O	EBI address/data bus bit 6.
	EBI_AD7	MFP7	I/O	EBI address/data bus bit 7.
	EBI_AD7	MFP7	I/O	EBI address/data bus bit 7.
	EBI_AD8	MFP7	I/O	EBI address/data bus bit 8.
	EBI_AD9	MFP7	I/O	EBI address/data bus bit 9.
	EBI_AD10	MFP7	I/O	EBI address/data bus bit 10.
	EBI_AD11	MFP7	I/O	EBI address/data bus bit 11.
	EBI_AD12	MFP7	I/O	EBI address/data bus bit 12.
	EBI_AD13	MFP7	I/O	EBI address/data bus bit 13.
	EBI_AD14	MFP7	I/O	EBI address/data bus bit 14.
	EBI_AD15	MFP7	I/O	EBI address/data bus bit 15.
	EBI_ADR16	MFP7	O	EBI address bus bit 16.
	EBI_ADR17	MFP7	O	EBI address bus bit 17.
	EBI_ADR18	MFP7	O	EBI address bus bit 18.

Group	Pin Name	MFP*	Type	Description
	EBI_ADR19	MFP7	O	EBI address bus bit 19.
	EBI_ALE	MFP7	O	EBI address latch enable output pin.
	EBI_ALE	MFP7	O	EBI address latch enable output pin.
	EBI_MCLK	MFP7	O	EBI external clock output pin
	EBI_nCS0	MFP7	O	EBI chip select 0 enable output pin.
	EBI_nCS0	MFP7	O	EBI chip select 0 enable output pin.
	EBI_nCS1	MFP7	O	EBI chip select 1 enable output pin.
	EBI_nCS1	MFP7	O	EBI chip select 1 enable output pin.
	EBI_nRD	MFP7	O	EBI read enable output pin.
	EBI_nRD	MFP7	O	EBI read enable output pin.
	EBI_nWR	MFP7	O	EBI write enable output pin.
	EBI_nWR	MFP7	O	EBI write enable output pin.
	EBI_nWRH	MFP7	O	EBI high byte write enable output pin
	EBI_nWRL	MFP7	O	EBI low byte write enable output pin.
I2C0	I2C0_SCL	MFP3	I/O	I2C0 clock pin.
	I2C0_SCL	MFP4	I/O	I2C0 clock pin.
	I2C0_SCL	MFP4	I/O	I2C0 clock pin.
	I2C0_SDA	MFP3	I/O	I2C0 data input/output pin.
	I2C0_SDA	MFP4	I/O	I2C0 data input/output pin.
	I2C0_SDA	MFP4	I/O	I2C0 data input/output pin.
	I2C0_SMBAL	MFP4	O	I2C0 SMBus SMBALTER# pin
	I2C0_SMBAL	MFP4	O	I2C0 SMBus SMBALTER# pin
	I2C0_SMBSUS	MFP4	O	I2C0 SMBus SMBSUS# pin (PMBus CONTROL pin)
	I2C0_SMBSUS	MFP4	O	I2C0 SMBus SMBSUS# pin (PMBus CONTROL pin)
I2C1	I2C1_SCL	MFP3	I/O	I2C1 clock pin.
	I2C1_SCL	MFP3	I/O	I2C1 clock pin.
	I2C1_SCL	MFP3	I/O	I2C1 clock pin.
	I2C1_SCL	MFP4	I/O	I2C1 clock pin.
	I2C1_SDA	MFP3	I/O	I2C1 data input/output pin.
	I2C1_SDA	MFP3	I/O	I2C1 data input/output pin.
	I2C1_SDA	MFP3	I/O	I2C1 data input/output pin.
	I2C1_SDA	MFP4	I/O	I2C1 data input/output pin.
	I2C1_SMBAL	MFP3	O	I2C1 SMBus SMBALTER# pin
	I2C1_SMBSUS	MFP3	O	I2C1 SMBus SMBSUS# pin (PMBus CONTROL pin)

Group	Pin Name	MFP*	Type	Description
I2S1	SPI1_I2SMCLK	MFP2	O	I2S1 master clock output pin.
	SPI1_I2SMCLK	MFP2	O	I2S1 master clock output pin.
I2S2	SPI2_I2SMCLK	MFP2	O	I2S2 master clock output pin.
	SPI2_I2SMCLK	MFP2	O	I2S2 master clock output pin.
ICE	ICE_CLK	MFP1	I	Serial wired debugger clock pin
	ICE_DAT	MFP1	I/O	Serial wired debugger data pin
INT0	INT0	MFP8	I	External interrupt0 input pin.
	INT0	MFP8	I	External interrupt0 input pin.
	INT0	MFP8	I	External interrupt0 input pin.
INT1	INT1	MFP8	I	External interrupt1 input pin.
	INT1	MFP8	I	External interrupt1 input pin.
	INT1	MFP8	I	External interrupt1 input pin.
INT2	INT2	MFP8	I	External interrupt2 input pin.
INT3	INT3	MFP8	I	External interrupt3 input pin.
INT4	INT4	MFP8	I	External interrupt4 input pin.
INT5	INT5	MFP8	I	External interrupt5 input pin.
PWM0	PWM0_BRAKE0	MFP6	I	PWM0 break input 0
	PWM0_BRAKE0	MFP5	I	PWM0 break input 0
	PWM0_BRAKE1	MFP6	I	PWM0 break input 1
	PWM0_BRAKE1	MFP5	I	PWM0 break input 1
	PWM0_CH0	MFP6	I/O	PWM0 output/capture input.
	PWM0_CH0	MFP6	I/O	PWM0 output/capture input.
	PWM0_CH1	MFP6	I/O	PWM0 output/capture input.
	PWM0_CH1	MFP6	I/O	PWM0 output/capture input.
	PWM0_CH2	MFP6	I/O	PWM0 output/capture input.
	PWM0_CH2	MFP6	I/O	PWM0 output/capture input.
	PWM0_CH3	MFP6	I/O	PWM0 output/capture input.
	PWM0_CH3	MFP6	I/O	PWM0 output/capture input.
	PWM0_CH4	MFP6	I/O	PWM0 output/capture input.
	PWM0_CH5	MFP6	I/O	PWM0 output/capture input.
	PWM0_CH5	MFP6	I/O	PWM0 output/capture input.
	PWM0_CH5	MFP6	I/O	PWM0 output/capture input.
	PWM0_SYNC_IN	MFP2	I	PWM0 counter synchronous trigger input pin.
	PWM0_SYNC_IN	MFP3	I	PWM0 counter synchronous trigger input pin.

Group	Pin Name	MFP*	Type	Description
	PWM0_SYNC_OUT	MFP6	O	PWM0 counter synchronous trigger output pin.
PWM1	PWM1_BRAKE0	MFP6	I	PWM1 break input 0
	PWM1_BRAKE1	MFP6	I	PWM1 break input 1
	PWM1_CH0	MFP6	I/O	PWM1 output/capture input.
	PWM1_CH0	MFP6	I/O	PWM1 output/capture input.
	PWM1_CH0	MFP6	I/O	PWM1 output/capture input.
	PWM1_CH0	MFP6	I/O	PWM1 output/capture input.
	PWM1_CH1	MFP6	I/O	PWM1 output/capture input.
	PWM1_CH1	MFP6	I/O	PWM1 output/capture input.
	PWM1_CH1	MFP6	I/O	PWM1 output/capture input.
	PWM1_CH1	MFP6	I/O	PWM1 output/capture input.
	PWM1_CH2	MFP6	I/O	PWM1 output/capture input.
	PWM1_CH2	MFP6	I/O	PWM1 output/capture input.
	PWM1_CH2	MFP6	I/O	PWM1 output/capture input.
	PWM1_CH3	MFP6	I/O	PWM1 output/capture input.
	PWM1_CH3	MFP6	I/O	PWM1 output/capture input.
	PWM1_CH3	MFP6	I/O	PWM1 output/capture input.
	PWM1_CH4	MFP6	I/O	PWM1 output/capture input.
	PWM1_CH4	MFP6	I/O	PWM1 output/capture input.
	PWM1_CH5	MFP6	I/O	PWM1 output/capture input.
	PWM1_CH5	MFP6	I/O	PWM1 output/capture input.
SC0	SC0_CD	MFP5	I	SmartCard card detect pin.
	SC0_CD	MFP5	I	SmartCard card detect pin.
	SC0_CD	MFP5	I	SmartCard card detect pin.
	SC0_CLK	MFP5	O	SmartCard clock pin.
	SC0_CLK	MFP5	O	SmartCard clock pin.
	SC0_DAT	MFP5	I/O	SmartCard data pin.
	SC0_DAT	MFP5	I/O	SmartCard data pin.
	SC0_PWR	MFP5	O	SmartCard power pin.
	SC0_PWR	MFP5	O	SmartCard power pin.
	SC0_PWR	MFP5	O	SmartCard power pin.
	SC0_RST	MFP5	O	SmartCard reset pin.
	SC0_RST	MFP5	O	SmartCard reset pin.
	SC0_RST	MFP5	O	SmartCard reset pin.

Group	Pin Name	MFP*	Type	Description
	SC0_RST	MFP5	O	SmartCard reset pin.
SPI0	SPI0_CLK	MFP2	I/O	SPI0 serial clock pin.
	SPI0_CLK	MFP2	I/O	SPI0 serial clock pin.
	SPI0_CLK	MFP2	I/O	SPI0 serial clock pin.
	SPI0_MISO0	MFP2	I/O	SPI0 1st MISO (Master In, Slave Out) pin.
	SPI0_MISO0	MFP2	I/O	SPI0 1st MISO (Master In, Slave Out) pin.
	SPI0_MISO0	MFP2	I/O	SPI0 1st MISO (Master In, Slave Out) pin.
	SPI0_MISO1	MFP2	I/O	SPI0 2nd MISO (Master In, Slave Out) pin.
	SPI0_MISO1	MFP2	I/O	SPI0 2nd MISO (Master In, Slave Out) pin.
	SPI0_MOSI0	MFP2	I/O	SPI0 1st MOSI (Master Out, Slave In) pin.
	SPI0_MOSI0	MFP2	I/O	SPI0 1st MOSI (Master Out, Slave In) pin.
	SPI0_MOSI1	MFP2	I/O	SPI0 2nd MOSI (Master Out, Slave In) pin.
	SPI0_MOSI1	MFP2	I/O	SPI0 2nd MOSI (Master Out, Slave In) pin.
	SPI0_SS	MFP2	I/O	SPI0 slave select pin.
	SPI0_SS	MFP2	I/O	SPI0 slave select pin.
SPI1	SPI1_CLK	MFP3	I/O	SPI1 serial clock pin
	SPI1_CLK	MFP2	I/O	SPI1 serial clock pin
	SPI1_CLK	MFP2	I/O	SPI1 serial clock pin
	SPI1_CLK	MFP1	I/O	SPI1 serial clock pin
	SPI1_CLK	MFP3	I/O	SPI1 serial clock pin
	SPI1_MISO	MFP3	I/O	SPI1 MISO (Master In, Slave Out) pin.
	SPI1_MISO	MFP2	I/O	SPI1 MISO (Master In, Slave Out) pin.
	SPI1_MISO	MFP2	I/O	SPI1 MISO (Master In, Slave Out) pin.
	SPI1_MISO	MFP1	I/O	SPI1 MISO (Master In, Slave Out) pin.
	SPI1_MISO	MFP3	I/O	SPI1 MISO (Master In, Slave Out) pin.
	SPI1_MOSI	MFP3	I/O	SPI1 MOSI (Master Out, Slave In) pin.
	SPI1_MOSI	MFP2	I/O	SPI1 MOSI (Master Out, Slave In) pin.
	SPI1_MOSI	MFP2	I/O	SPI1 MOSI (Master Out, Slave In) pin.
	SPI1_MOSI	MFP1	I/O	SPI1 MOSI (Master Out, Slave In) pin.
	SPI1_SS	MFP2	I/O	SPI1 slave select pin
	SPI1_SS	MFP2	I/O	SPI1 slave select pin
	SPI1_SS	MFP1	I/O	SPI1 slave select pin
	SPI1_SS	MFP3	I/O	SPI1 slave select pin
SPI2	SPI2_CLK	MFP2	I/O	SPI2 serial clock pin.

Group	Pin Name	MFP*	Type	Description
	SPI2_CLK	MFP2	I/O	SPI2 serial clock pin.
	SPI2_CLK	MFP2	I/O	SPI2 serial clock pin.
	SPI2_CLK	MFP2	I/O	SPI2 serial clock pin.
	SPI2_MISO	MFP2	I/O	SPI2 MISO (Master In, Slave Out) pin.
	SPI2_MISO	MFP2	I/O	SPI2 MISO (Master In, Slave Out) pin.
	SPI2_MISO	MFP2	I/O	SPI2 MISO (Master In, Slave Out) pin.
	SPI2_MOSI	MFP2	I/O	SPI2 MOSI (Master Out, Slave In) pin.
	SPI2_MOSI	MFP2	I/O	SPI2 MOSI (Master Out, Slave In) pin.
	SPI2_MOSI	MFP2	I/O	SPI2 MOSI (Master Out, Slave In) pin.
	SPI2_SS	MFP2	I	SPI2 slave select pin.
	SPI2_SS	MFP2	I	SPI2 slave select pin.
	SPI2_SS	MFP2	I	SPI2 slave select pin.
TAMPER	TAMPER	MFP1	I/O	TAMPER detector loop pin
TK	TK0	MFP4	A	Touch Key0
	TK1	MFP4	A	Touch Key1
	TK2	MFP4	A	Touch Key2
	TK3	MFP4	A	Touch Key3
	TK4	MFP4	A	Touch Key4
	TK5	MFP4	A	Touch Key5
	TK6	MFP4	A	Touch Key6
	TK7	MFP4	A	Touch Key7
	TK8	MFP4	A	Touch Key0
	TK9	MFP4	A	Touch Key8
	TK10	MFP4	A	Touch Key10
	TK11	MFP4	A	Touch Key11
	TK12	MFP4	A	Touch Key12
	TK13	MFP4	A	Touch Key13
	TK14	MFP4	A	Touch Key14
	TK15	MFP4	A	Touch Key15
	TK16	MFP4	A	Touch Key16
TMR0	T0	MFP6	I/O	Timer0event counter input / toggle output
	T0	MFP6	I/O	Timer0event counter input / toggle output
	T0_EXT	MFP3	I	Timer0 external capture input
	T0_EXT	MFP3	I	Timer0 external capture input

Group	Pin Name	MFP*	Type	Description
TMR1	T1	MFP6	I/O	Timer1 event counter input / toggle output
	T1	MFP4	I/O	Timer1 event counter input / toggle output
	T1_EXT	MFP3	I	Timer1 external capture input
	T1_EXT	MFP3	I	Timer1 external capture input
TMR2	T2	MFP1	I/O	Timer2 event counter input / toggle output
	T2	MFP4	I/O	Timer2 event counter input / toggle output
	T2	MFP4	I/O	Timer2 event counter input / toggle output
	T2_EXT	MFP4	I	Timer2 external capture input
	T2_EXT	MFP3	I	Timer2 external capture input
TMR3	T3	MFP4	I/O	Timer3 event counter input / toggle output
	T3	MFP4	I/O	Timer3 event counter input / toggle output
	T3_EXT	MFP3	I	Timer3 external capture input
	T3_EXT	MFP3	I	Timer3 external capture input
UART0	UART0_RXD	MFP3	I	Data receiver input pin for UART0.
	UART0_RXD	MFP3	I	Data receiver input pin for UART0.
	UART0_RXD	MFP2	I	Data receiver input pin for UART0.
	UART0_TXD	MFP3	O	Data transmitter output pin for UART0.
	UART0_TXD	MFP2	O	Data transmitter output pin for UART0.
	UART0_nCTS	MFP3	I	Clear to Send input pin for UART0.
	UART0_nRTS	MFP3	O	Request to Send output pin for UART0.
UART1	UART1_RXD	MFP1	I	Data receiver input pin for UART1.
	UART1_RXD	MFP3	I	Data receiver input pin for UART1.
	UART1_RXD	MFP3	I	Data receiver input pin for UART1.
	UART1_RXD	MFP4	I	Data receiver input pin for UART1.
	UART1_TXD	MFP1	O	Data transmitter output pin for UART1.
	UART1_TXD	MFP3	O	Data transmitter output pin for UART1.
	UART1_TXD	MFP3	O	Data transmitter output pin for UART1.
	UART1_TXD	MFP4	O	Data transmitter output pin for UART1.
	UART1_nCTS	MFP3	I	Clear to Send input pin for UART1.
	UART1_nCTS	MFP1	I	Clear to Send input pin for UART1.
	UART1_nCTS	MFP4	I	Clear to Send input pin for UART1.
	UART1_nRTS	MFP3	O	Request to Send output pin for UART1.
	UART1_nRTS	MFP1	O	Request to Send output pin for UART1.
	UART1_nRTS	MFP4	O	Request to Send output pin for UART1.

Group	Pin Name	MFP*	Type	Description
UART2	UART2_RXD	MFP3	I	Data receiver input pin for UART2.
	UART2_RXD	MFP3	I	Data receiver input pin for UART2.
	UART2_TXD	MFP3	O	Data transmitter output pin for UART2.
	UART2_TXD	MFP3	O	Data transmitter output pin for UART2.
	UART2_nCTS	MFP3	I	Clear to Send input pin for UART2.
	UART2_nCTS	MFP3	I	Clear to Send input pin for UART2.
	UART2_nRTS	MFP3	O	Request to Send output pin for UART2.
	UART2_nRTS	MFP3	O	Request to Send output pin for UART2.
UART3	UART3_RXD	MFP3	I	Data receiver input pin for UART3.
	UART3_RXD	MFP3	I	Data receiver input pin for UART3.
	UART3_TXD	MFP3	O	Data transmitter output pin for UART3.
	UART3_TXD	MFP3	O	Data transmitter output pin for UART3.
	UART3_nCTS	MFP3	I	Clear to Send input pin for UART3.
	UART3_nCTS	MFP3	I	Clear to Send input pin for UART3.
	UART3_nRTS	MFP3	O	Request to Send output pin for UART3.
	UART3_nRTS	MFP3	O	Request to Send output pin for UART3.
USB	USB_VBUS_EN	MFP4	O	USB external VBUS regulator enable pin.
	USB_VBUS_EN	MFP1	O	USB external VBUS regulator enable pin.
	USB_VBUS_ST	MFP4	I	USB external VBUS regulator status pin.
	USB_VBUS_ST	MFP1	I	USB external VBUS regulator status pin.
LXT	X32_IN	MFP1	I	External 32.768 kHZ (low speed) crystal input pin.
	X32_OUT	MFP1	O	External 32.768 kHZ (low speed) crystal output pin.
HXT	XT1_IN	MFP1	I	External 4~20 MHz (high speed) crystal input pin.
	XT1_OUT	MFP1	O	External 4~20 MHz (high speed) crystal output pin.

Table 4-1 M4TK GPIO Multi-function Table

5 BLOCK DIAGRAM

5.1 NuMicro® M4TK Block Diagram

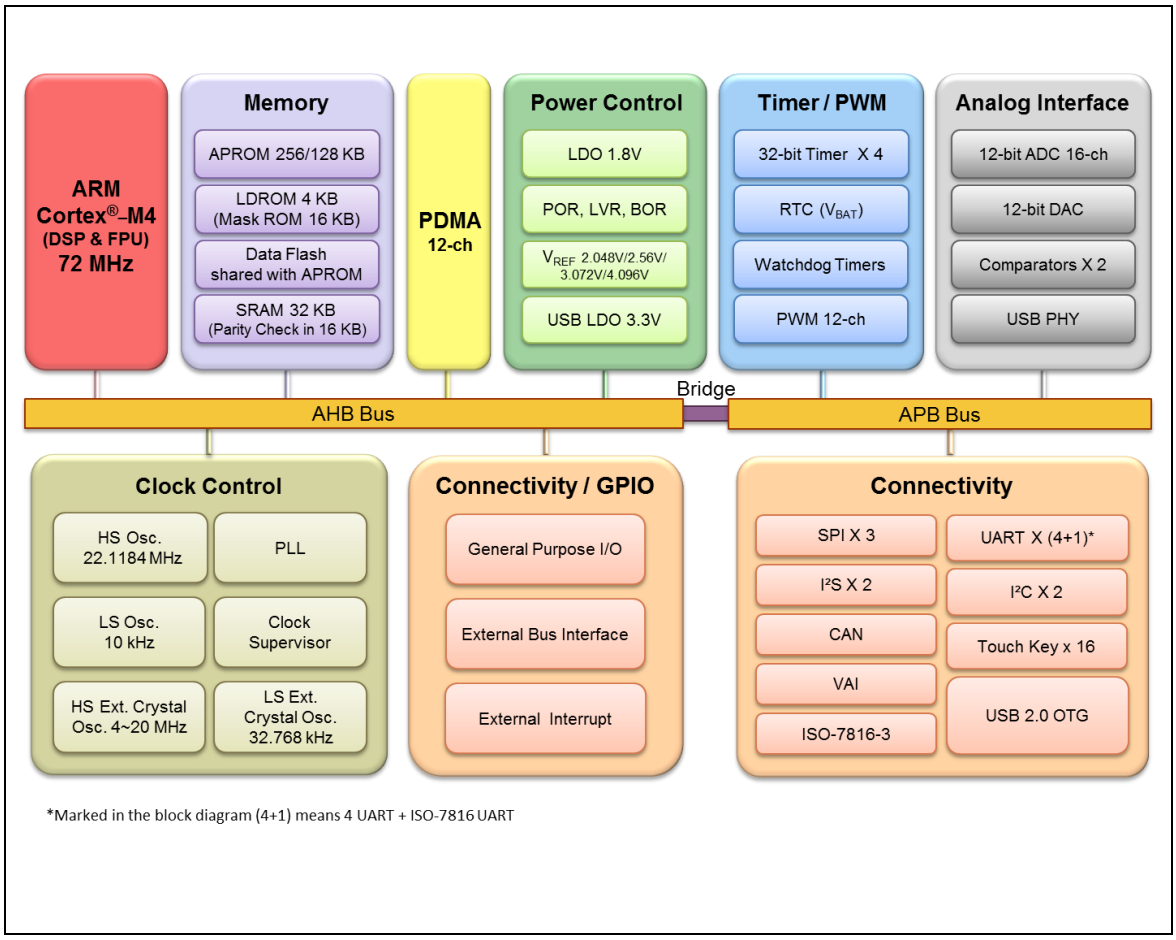


Figure 5.1-1 NuMicro® M4TK Block Diagram

6 FUNCTIONAL DESCRIPTION

6.1 ARM® Cortex®-M4 Core

The Cortex®-M4 processor, a configurable, multistage, 32-bit RISC processor, has three AMBA AHB-Lite interfaces for best parallel performance and includes an NVIC component. The processor with optional hardware debug functionality can execute Thumb code and is compatible with other Cortex-M profile processors. The profile supports two modes -Thread mode and Handler mode. Handler mode is entered as a result of an exception. An exception return can only be issued in Handler mode. Thread mode is entered on Reset, and can be entered as a result of an exception return. The Cortex®-M4F is a processor with the same capability as the Cortex®-M4 processor and includes floating point arithmetic functionality. The NuMicro® M4TK family is embedded with Cortex®-M4F processor. Throughout this document, the name Cortex®-M4 refers to both Cortex®-M4 and Cortex®-M4F processors. The Figure 6.1-1 shows the functional controller of the processor.

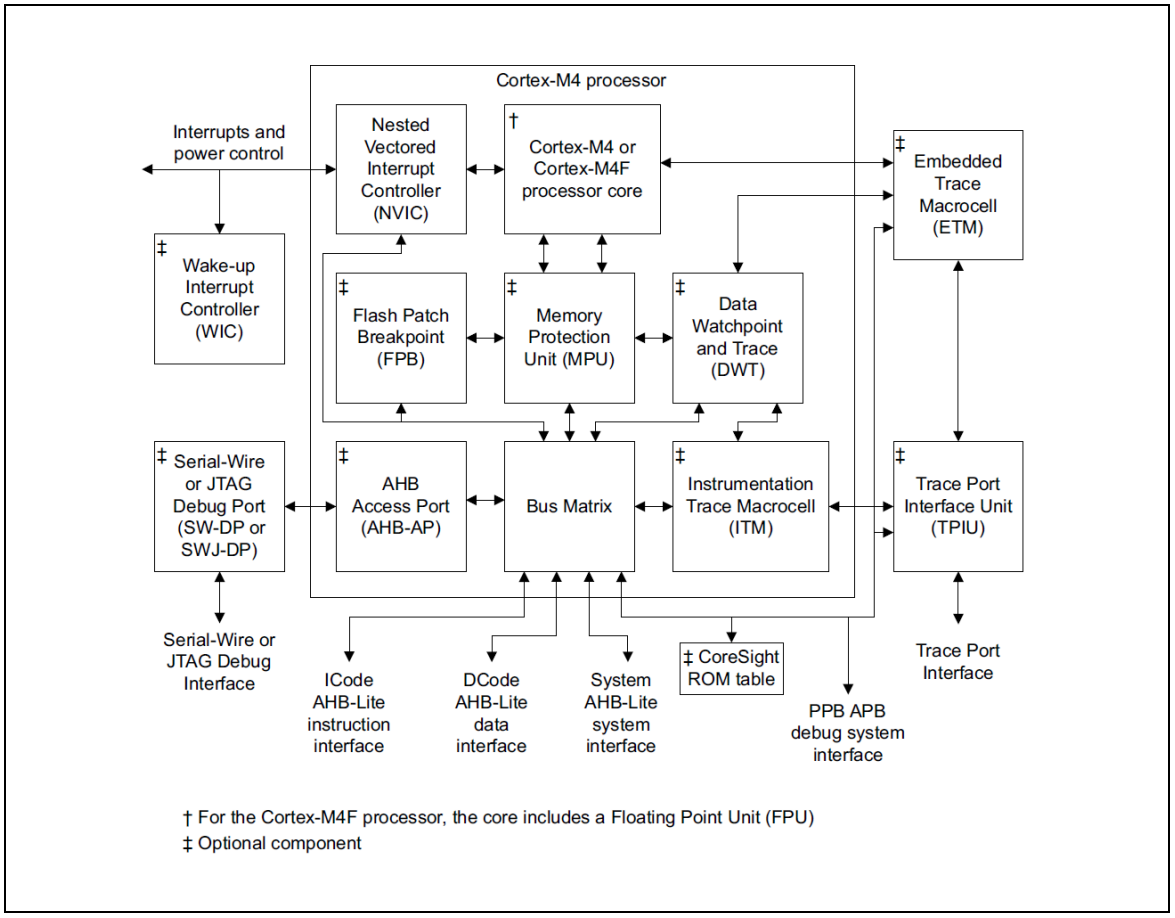


Figure 6.1-1 Cortex®-M4 Block Diagram

Cortex®-M4 processor features:

- A low gate count processor core, with low latency interrupt processing that has:
  - A subset of the Thumb instruction set, defined in the *ARMv7-M Architecture Reference Manual*
  - Banked Stack Pointer (SP)

- Hardware integer divide instructions, SDIV and UDIV
- Handler and Thread modes
- Thumb and Debug states
- Support for interruptible-continued instructions LDM, STM, PUSH, and POP for low interrupt latency
- Automatic processor state saving and restoration for low latency *Interrupt Service Routine (ISR)* entry and exit
- Support for ARMv6 big-endian byte-invariant or little-endian accesses
- Support for ARMv6 unaligned accesses

- Floating Point Unit (FPU) in the Cortex®-M4F processor providing:
  - 32-bit instructions for single-precision (C float) data-processing operations
  - Combined Multiply and Accumulate instructions for increased precision (Fused MAC)
  - Hardware support for conversion, addition, subtraction, multiplication with optional accumulate, division, and square-root
  - Hardware support for denormals and all IEEE rounding modes
  - 32 dedicated 32-bit single precision registers, also addressable as 16 double-word registers
  - Decoupled three stage pipeline
- Nested Vectored Interrupt Controller (NVIC) closely integrated with the processor core to achieve low latency interrupt processing. Features include:
  - External interrupts. Configurable from 1 to 240 (the NuMicro® M4TK family configured with 64 interrupts)
  - Bits of priority, configurable from 3 to 8
  - Dynamic reprioritization of interrupts
  - Priority grouping which enables selection of preempting interrupt levels and nonpreempting interrupt levels
  - Support for tril-chaining and late arrival of interrupts, which enables back-to-back interrupt processing without the overhead of state saving and restoration between interrupts.
  - Processor state automatically saved on interrupt entry, and restored on interrupt exit with on instruction overhead
  - Support for Wake-up Interrupt Controller (WIC) with Ultra-low Power Sleep mode
- Memory Protection Unit (MPU). An optional MPU for memory protection, including:
  - Eight memory regions
  - Sub Region Disable (SRD), enabling efficient use of memory regions
  - The ability to enable a background region that implements the default memory map attributes
- Low-cost debug solution that features:
  - Debug access to all memory and registers in the system, including access to memory mapped devices, access to internal core registers when the core is halted, and access to debug control registers even while SYSRESETn is

asserted.

- Serial Wire Debug Port(SW-DP) or Serial Wire JTAG Debug Port (SWJ-DP) debug access
- Optional Flash Patch and Breakpoint (FPB) unit for implementing breakpoints and code patches
- Optional Data Watchpoint and Trace (DWT) unit for implementing watchpoints, data tracing, and system profiling
- Optional Instrumentation Trace Macrocell (ITM) for support of printf() style debugging
- Optional Trace Port Interface Unit (TPIU) for bridging to a Trace Port Analyzer (TPA), including Single Wire Output (SWO) mode
- Optional Embedded Trace Macrocell (ETM) for instruction trace.

- Bus interfaces:
  - Three Advanced High-performance Bus-Lite (AHB-Lite) interfaces: ICode, Dcode, and System bus interfaces
  - Private Peripheral Bus (PPB) based on Advanced Peripheral Bus (APB) interface
  - Bit-band support that includes atomic bit-band write and read operations.
  - Memory access alignment
  - Write buffer for buffering of write data
  - Exclusive access transfers for multiprocessor systems

6.2 System Manager

6.2.1 Overview

The system manager provides the functions of system control, power modes, wake-up sources, reset sources, system memory map, product ID and multi-function pin control. The following sections describe the functions for

- System Reset
- Power Modes and Wake-up Sources
- System Power Distribution
- SRAM Memory Organization
- System Control Register for Part Number ID, Chip Reset and Multi-function Pin Control
- System Timer (SysTick)
- Nested Vectored Interrupt Controller (NVIC)
- System Control register

6.2.2 System Reset

The system reset can be issued by one of the events listed below. These reset event flags can be read from SYS_RSTSTS register to determine the reset source. Hardware reset can reset chip through peripheral reset signals. Software reset can trigger reset through control registers.

- Hardware Reset Sources
  - Power-on Reset (POR)
  - Low level on the nRESET pin
  - Watchdog Time-out Reset and Window Watchdog Reset (WDT/WWDT Reset)
  - Low Voltage Reset (LVR)
  - Brown-out Detector Reset (BOD Reset)
- Software Reset Sources
  - CHIP Reset will reset whole chip by writing 1 to CHIPRST (SYS_IPRST0[0])
  - MCU Reset to reboot but keeping the booting setting from APROM or LDROM by writing 1 to SYSRESETREQ (AIRCR[2])
  - CPU Reset for Cortex®-M4 core Only by writing 1 to CPURST (SYS_IPRST0[1])

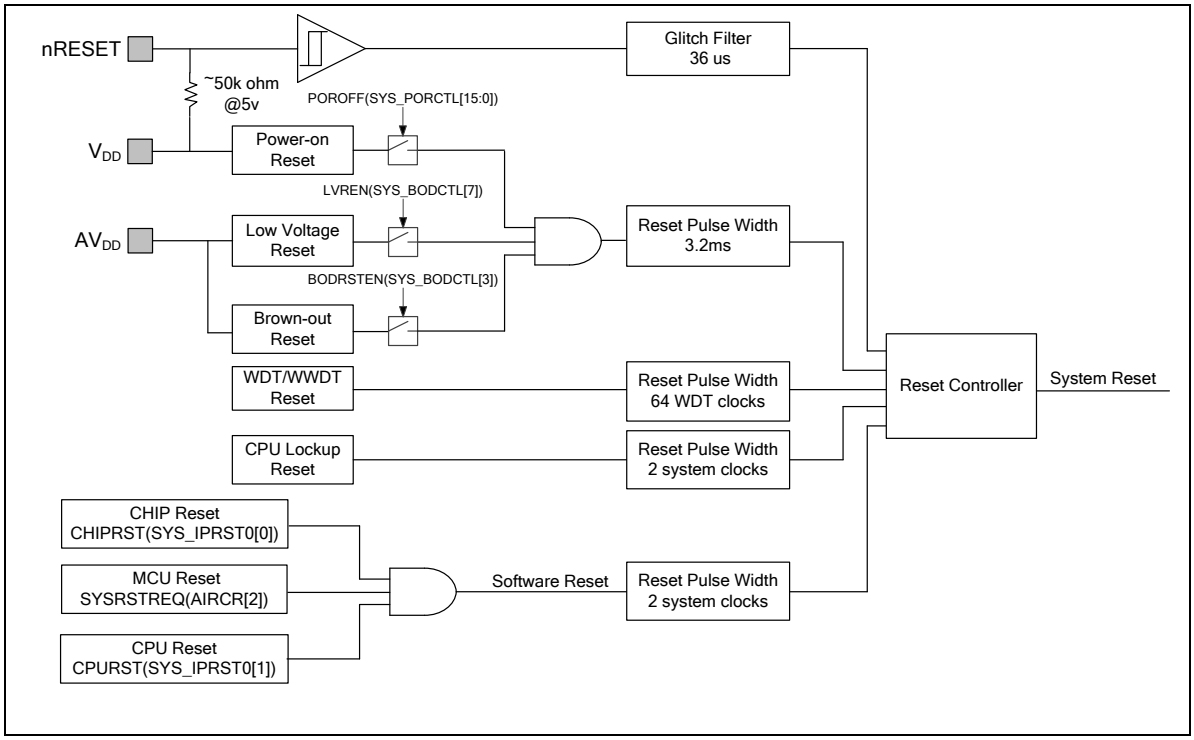


Figure 6.2-1 System Reset Sources

There are a total of 9 reset sources in the NuMicro® family. In general, CPU reset is used to reset Cortex-M4 only; the other reset sources will reset Cortex-M4 and all peripherals. However, there are small differences between each reset source and they are listed in Table 6-1.

Reset Sources Register	POR	NRESET	WDT	LVR	BOD	Lockup	CHIP	MCU	CPU
SYS_RSTSTS	0x001	Bit 1 = 1	Bit 2 = 1	Bit 3 = 1	Bit 4 = 1	Bit 8 = 1	Bit 0 = 1	Bit 5 = 1	Bit 7 = 1
CHIPRST (SYS_IPRST0[0])	0x0	-	-	-	-	-	-	-	-
BODEN (SYS_BODCTL[0])	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	-	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	-
BODVL (SYS_BODCTL[2:1])									
BODRSTEN (SYS_BODCTL[3])									
HXTEN (CLK_PWRCTL[0])	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	
LXTEN (CLK_PWRCTL[1])	0x0	-	-	-	-	-	-	-	-
WDTCKEN (CLK_APBCLK0[0])	0x1	-	0x1	-	-	-	0x1	-	-
HCLKSEL (CLK_CLKSEL0[2:0])	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	-
WDTSEL (CLK_CLKSEL1[1:0])	0x3	0x3	-	-	-	-	-	-	-
HXTSTB (CLK_STATUS[0])	0x0	-	-	-	-	-	-	-	-
LXTSTB (CLK_STATUS[1])	0x0	-	-	-	-	-	-	-	-
PLLSTB (CLK_STATUS[2])	0x0	-	-	-	-	-	-	-	-
HIRCSTB (CLK_STATUS[4])	0x0	-	-	-	-	-	-	-	-
CLKSFAIL (CLK_STATUS[7])	0x0	0x0	-	-	-	-	-	-	-
RSTEN (WDT_CTL[1])	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	-	Reload from CONFIG0	-	-
WDTEN (WDT_CTL[7])									
WDT_CTL except bit 1 and bit 7.	0x0700	0x0700	0x0700	0x0700	0x0700	-	0x0700	-	-

WDT_ALTCTL	0x0000	0x0000	0x0000	0x0000	0x0000	-	0x0000	-	-
WWDT_RLDCNT	0x0000	0x0000	0x0000	0x0000	0x0000	-	0x0000	-	-
WWDT_CTL	0x3F0800	0x3F0800	0x3F0800	0x3F0800	0x3F0800	-	0x3F0800	-	-
WWDT_STATUS	0x0000	0x0000	0x0000	0x0000	0x0000	-	0x0000	-	-
WWDT_CNT	0x3F	0x3F	0x3F	0x3F	0x3F	-	0x3F	-	-
BS (FMC_ISPCTL[1])	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	-	Reload from CONFIG0	-	-
BL (FMC_ISPCTL[16])									
FMC_DFBA	Reload from CONFIG1	Reload from CONFIG1	Reload from CONFIG1	Reload from CONFIG1	Reload from CONFIG1	-	Reload from CONFIG1	-	-
CBS (FMC_ISPSTS[2:1])	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	Reload from CONFIG0	-	Reload from CONFIG0	-	-
MBS ● **(FMC_ISPSTS[3])**									
PGFF (FMC_ISPSTS[5])	0x0	-	0x0	-	-	-	0x0	-	-
VECMAP ● **(FMC_ISPSTS[23:9])**	Reload base on CONFIG0	Reload base on CONFIG0	Reload base on CONFIG0	Reload base on CONFIG0	Reload base on CONFIG0	-	Reload base on CONFIG0	-	-
Other Peripheral Registers	Reset Value								-
FMC Registers	Reset Value								
Note: '-' means that the value of register keeps original setting.									

Table 6-1 Reset Value of Registers

6.2.2.1 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 V_{DD} 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 V_{DD} 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 6.2-2 shows the nRESET reset waveform.

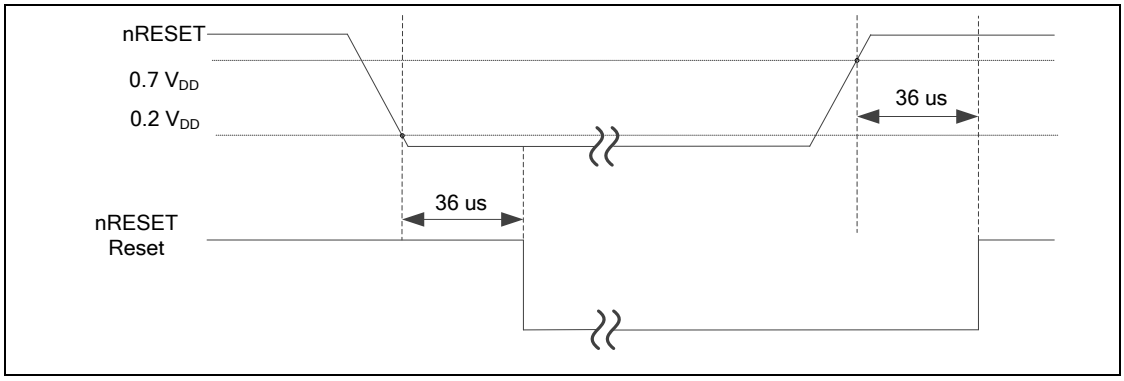


Figure 6.2-2 nRESET Reset Waveform

6.2.2.2 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 PORF(SYS_RSTSTS[0]) will be set to 1 to indicate there is a POR reset event. The PORF(SYS_RSTSTS[0]) bit can be cleared by writing 1 to it. Figure 6.2-3 shows the power-on reset waveform.

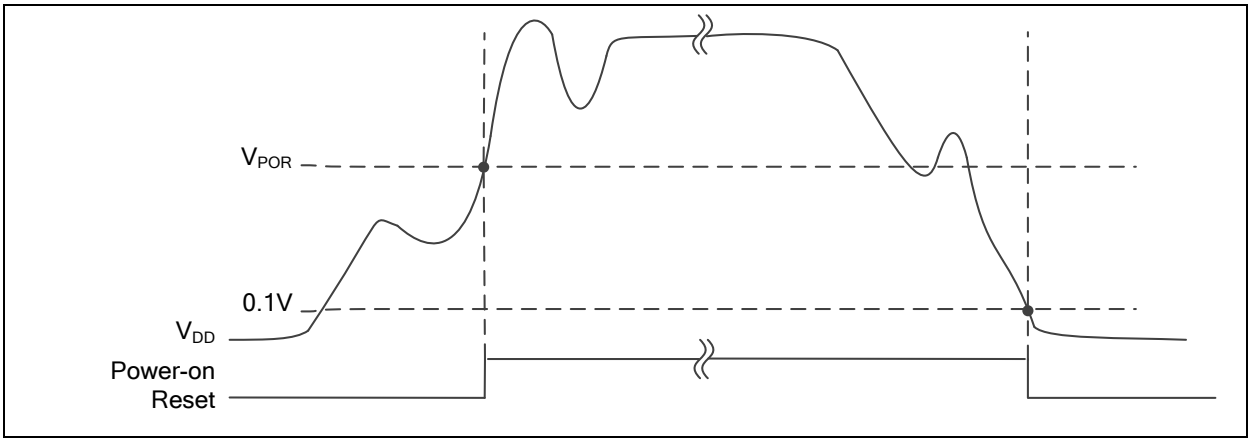


Figure 6.2-3 Power-on Reset (POR) Waveform

6.2.2.3 Low Voltage Reset (LVR)

If the Low Voltage Reset function is enabled by setting the Low Voltage Reset Enable Bit LVREN (SYS_BODCTL[7]) to 1, after 200us delay, LVR detection circuit will be stable and the LVR function will be active. Then LVR function will detect AV_{DD} during system operation. When the AV_{DD} voltage is lower than V_{LVR} and the state keeps longer than De-glitch time set by LVRDGSEL (SYS_BODCTL[14:12]), chip will be reset. The LVR reset will control the chip in reset state until the AV_{DD} voltage rises above V_{LVR} and the state keeps longer than De-glitch time set by LVRDGSEL (SYS_BODCTL[14:12]). The PINRF(SYS_RSTSTS[1]) will be set to 1 if the previous reset source is nRESET reset. The default setting of Low Voltage Reset is enabled without De-glitch function. Figure 6.2-4 shows the Low Voltage Reset waveform.

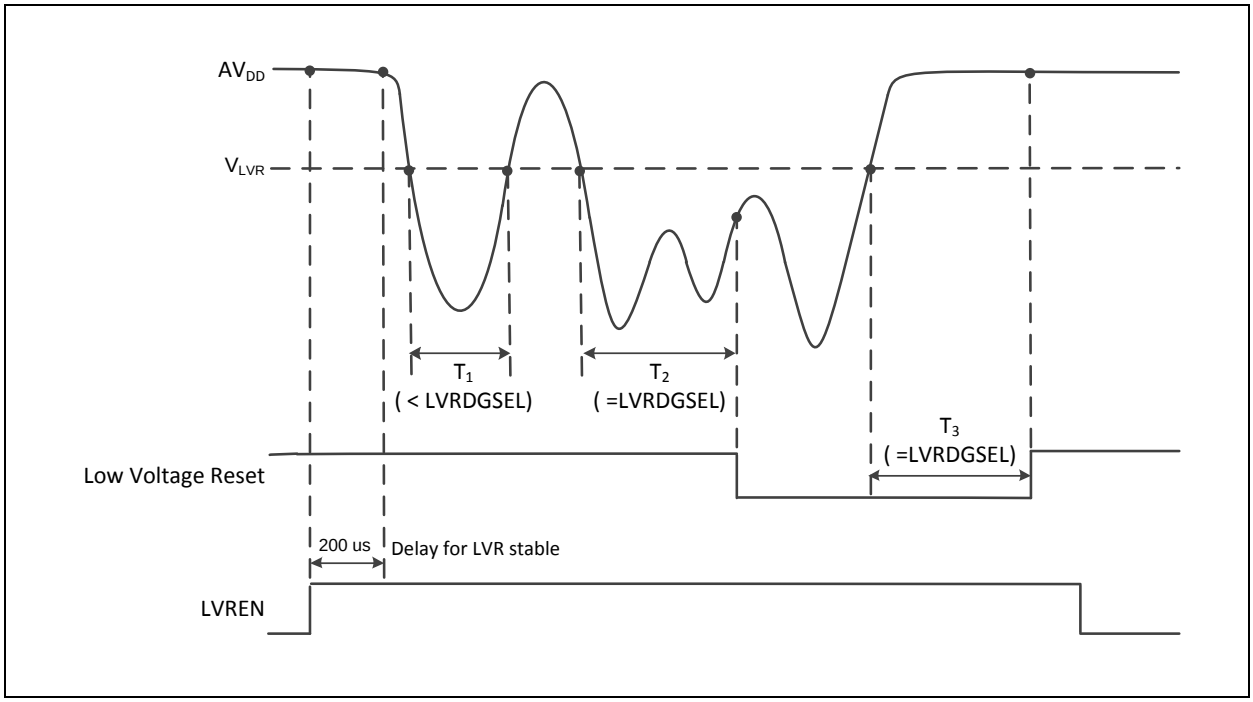


Figure 6.2-4 Low Voltage Reset (LVR) Waveform

6.2.2.4 Brown-out Detector Reset (BOD Reset)

If the Brown-out Detector (BOD) function is enabled by setting the Brown-out Detector Enable Bit BODEN (SYS_BODCTL[0]), Brown-Out Detector function will detect AV_{DD} during system operation. When the AV_{DD} voltage is lower than V_{BOD} and the state keeps longer than De-glitch time set by BODDGSEL (SYS_BODCTL[10:8]), chip will be reset. The BOD reset will control the chip in reset state until the AV_{DD} voltage rises above V_{BOD} and the state keeps longer than De-glitch time set by BODDGSEL (SYS_BODCTL[10:8]). The default value of BODEN, BODVL and BODRSTEN is set by flash controller user configuration register CBODEN (CONFIG0 [23]), CBOV (CONFIG0 [22:21]) and CBORST(CONFIG0[20]) respectively. User can determine the initial BOD setting by setting the CONFIG0 register. Figure 6.2-5 shows the Brown-Out Detector waveform.

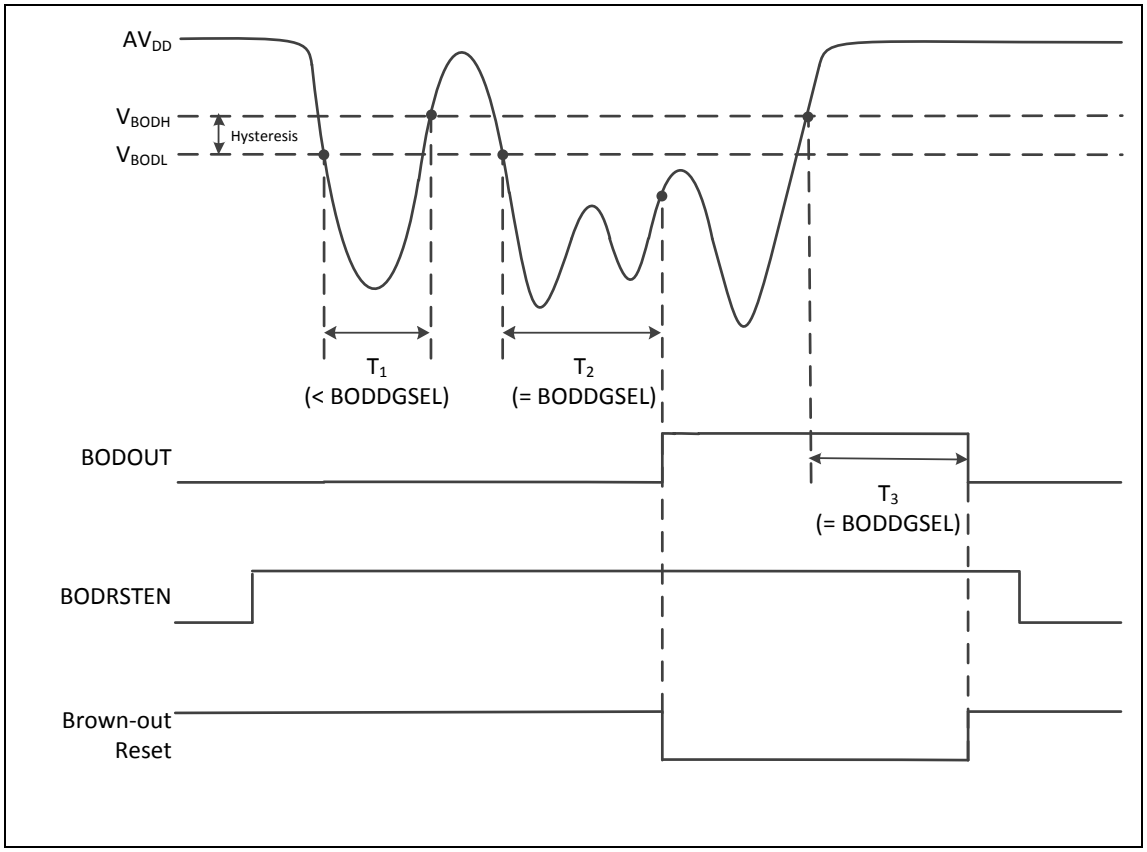


Figure 6.2-5 Brown-out Detector (BOD) Waveform

6.2.2.5 Watchdog Timer Reset (WDT)

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 watchdog 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 watchdog time-out. User may decide to enable system reset during watchdog 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 watchdog time-out to indicate the previous reset is a watchdog reset and handle the failure of MCU after watchdog time-out reset by checking WDTRF(SYS_RSTSTS[2]).

6.2.2.6 CPU Reset, CHIP Reset and MCU Reset

The CPU Reset means only Cortex®-M4 core is reset and all other peripherals remain the same status after CPU reset. User can set the CPURST(SYS_IPRST0[1]) to 1 to assert the CPU Reset signal.

The CHIP Reset is same with Power-On Reset. The CPU and all peripherals are reset and BS(FMC_ISPCTL[1]) bit is automatically reloaded from CONFIG setting. User can set the CHIPRST(SYS_IPRST0[1]) to 1 to assert the CHIP Reset signal.

The MCU Reset is similar with CHIP Reset. The difference is that BS(FMC_ISPCTL[1]) will not be reloaded from CONFIG setting and keep its original software setting for booting from APROM or LDROM. User can set the SYSRESETREQ(AIRCR[2]) to 1 to assert the MCU Reset.

6.2.3 Power Modes and Wake-up Sources

There are several wake-up sources in Idle mode and Power-down mode. Table 6-2 lists the available clocks for each power mode.

Power Mode	Normal Mode	Idle Mode	Power-Down Mode
Definition	CPU is in active state	CPU is in sleep state	CPU is in sleep state and all clocks stop except LXT and LIRC. SRAM content retended.
Entry Condition	Chip is in normal mode after system reset released	CPU executes WFI instruction.	CPU sets sleep mode enable and power down enable and executes WFI instruction.
Wake-up Sources	N/A	All interrupts	RTC, WDT, I²C, Timer, UART, BOD, GPIO, USBH, USBD, OTG, CAN, ACMP and TK
Available Clocks	All	All except CPU clock	LXT and LIRC
After Wake-up	N/A	CPU back to normal mode	CPU back to normal mode

Table 6-2 Power Mode Difference Table

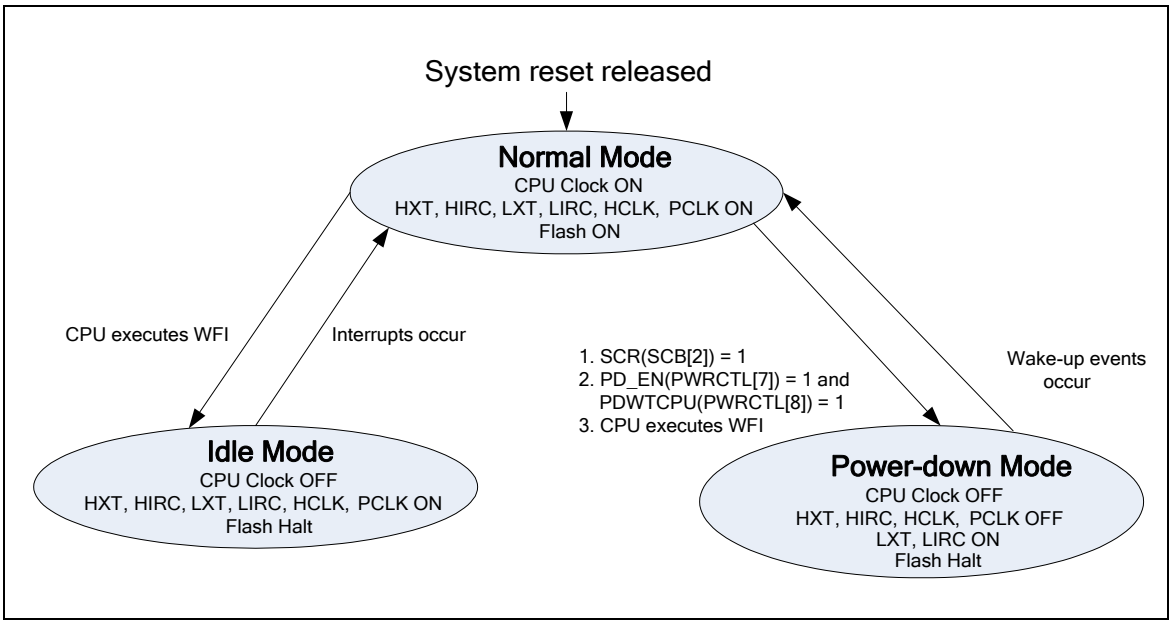


Figure 6.2-6 Power Mode State Machine

1. LXT (32768 Hz XTL) ON or OFF depends on SW setting in run mode.
2. LIRC (10 kHz OSC) ON or OFF depends on S/W setting in run mode.
3. If TIMER clock source is selected as LIRC/LXT and LIRC/LXT is on.
4. If WDT clock source is selected as LIRC and LIRC is on.
5. If RTC clock source is selected as LXT and LXT is on.

	Normal Mode	Idle Mode	Power-Down Mode
HXT (4~20 MHz XTL)	ON	ON	Halt
HIRC (12/16 MHz OSC)	ON	ON	Halt
LXT (32768 Hz XTL)	ON	ON	ON/OFF[1]
LIRC (10 kHz OSC)	ON	ON	ON/OFF[2]
PLL	ON	ON	Halt
LDO	ON	ON	ON
CPU	ON	Halt	Halt
HCLK/PCLK	ON	ON	Halt
SRAM retention	ON	ON	ON
FLASH	ON	ON	Halt
EBI	ON	ON	Halt
GPIO	ON	ON	Halt
PDMA	ON	ON	Halt
TIMER	ON	ON	ON/OFF[3]
PWM	ON	ON	Halt
WDT	ON	ON	ON/OFF[4]
WWDT	ON	ON	Halt
RTC	ON	ON	ON/OFF[5]
UART	ON	ON	Halt
SC	ON	ON	Halt
I^2C	ON	ON	Halt
SPI	ON	ON	Halt
USBH	ON	ON	Halt
USBG	ON	ON	Halt
OTG	ON	ON	Halt
CAN	ON	ON	Halt
TK	ON	ON	Halt
EADC	ON	ON	Halt
DAC	ON	ON	Halt

ACMP	ON	ON	Halt

Table 6-3 Clocks in Power Modes

Wake-up sources in Power-down mode:

RTC, WDT, I²C, Timer, UART, BOD, GPIO, USBH, USBD, OTG, CAN, ACMP and TK

After chip enters power down, the following wake-up sources can wake chip up to normal mode. Table 6-4 lists the condition about how to enter Power-down mode again for each peripheral.

*User needs to wait this condition before setting PD_EN(PWRCTL[6]) and execute WFI to enter Power-down mode.

Wake-Up Source	Wake-Up Condition	System Can Enter Power-Down Mode Again Condition*
BOD	Brown-Out Detector Interrupt	After software writes 1 to clear SYS_BODCTL[BODIF].
GPIO	GPIO Interrupt	After software write 1 to clear the INTSRC[n] bit.
TIMER	Timer Interrupt	After software writes 1 to clear TWKF (TIMERx_INTSTS[1]) and TIF (TIMERx_INTSTS[0]).
WDT	WDT Interrupt	After software writes 1 to clear WKF (WDT_CTL[5]) (Write Protect).
RTC	Alarm Interrupt	After software writes 1 to clear ALMIF (RTC_INTSTS[0]).
	Time Tick Interrupt	After software writes 1 to clear TICKIF (RTC_INTSTS[1]).
	Snoop Detection Interrupt	After software writes 1 to clear SNPDIF (RTC_INTSTS[2]).
UART	RX Data wake-up	After software writes 1 to clear DATWKIF (UARTx_INTSTS[17]).
	nCTS wake-up	After software writes 1 to clear CTSWKIF (UARTx_INTSTS[16]).
I²C	Falling edge in the I2C_SDA or I2C_CLK	After software writes 1 to clear WKIF(I2C_WKSTS[0]).
USBH	Remote Wake-up	After software writes 1 to clear CSC (HcRhPortStatus1[16]).
USBD	Remote Wake-up	After software writes 1 to clear BUSIF (USBD_INTSTS[0]).
OTG	OTG ID Pin Wake-Up	After software writes 1 to clear OTG_INTSTS[IDCHGIF].
CAN	Falling edge in the CAN_RX	After software writes 0 to clear WAKUP_STS (CAN_WU_STATUS [0]).
TK	Wake-up by Key Touch/Release or Any Key Touch	After software writes 1 to clear TK_STATUS[TKIFx].
ACMP	Comparator Power-Down Wake-Up Interrupt	After software writes 1 to clear ACMP_STATUS[WKIFx].

Table 6-4 Condition of Entering Power-down Mode Again

6.2.4 System Power Distribution

In this chip, power distribution is divided into five segments:

- Analog power from AV_{DD} and AV_{SS} provides the power for analog components operation. The V_{REF} should be connected with an external 1uF capacitor that should be located close to the V_{REF} pin to avoid power noise for analog applications.
- Digital power from V_{DD} and V_{SS} supplies the power to the internal regulator which provides a fixed 1.8 V power for digital operation and I/O pins.
- USB transceiver power from V_{DD} offers the power for operating the USB transceiver.
- RTC power from V_{BAT} provides the power for PF.0~PF.2, RTC and 80 bytes backup registers.
- A dedicated power from V_{DDIO} supplies the power for PE.8~PE.13.

The outputs of internal voltage regulators, LDO_CAP and USB_VDD33_CAP, require an external capacitor which should be located close to the corresponding pin. Analog power (AV_{DD}) should be the same voltage level of the digital power (V_{DD}). The Figure 6.2-7 shows the power distribution of the NuMicro® M4TK.

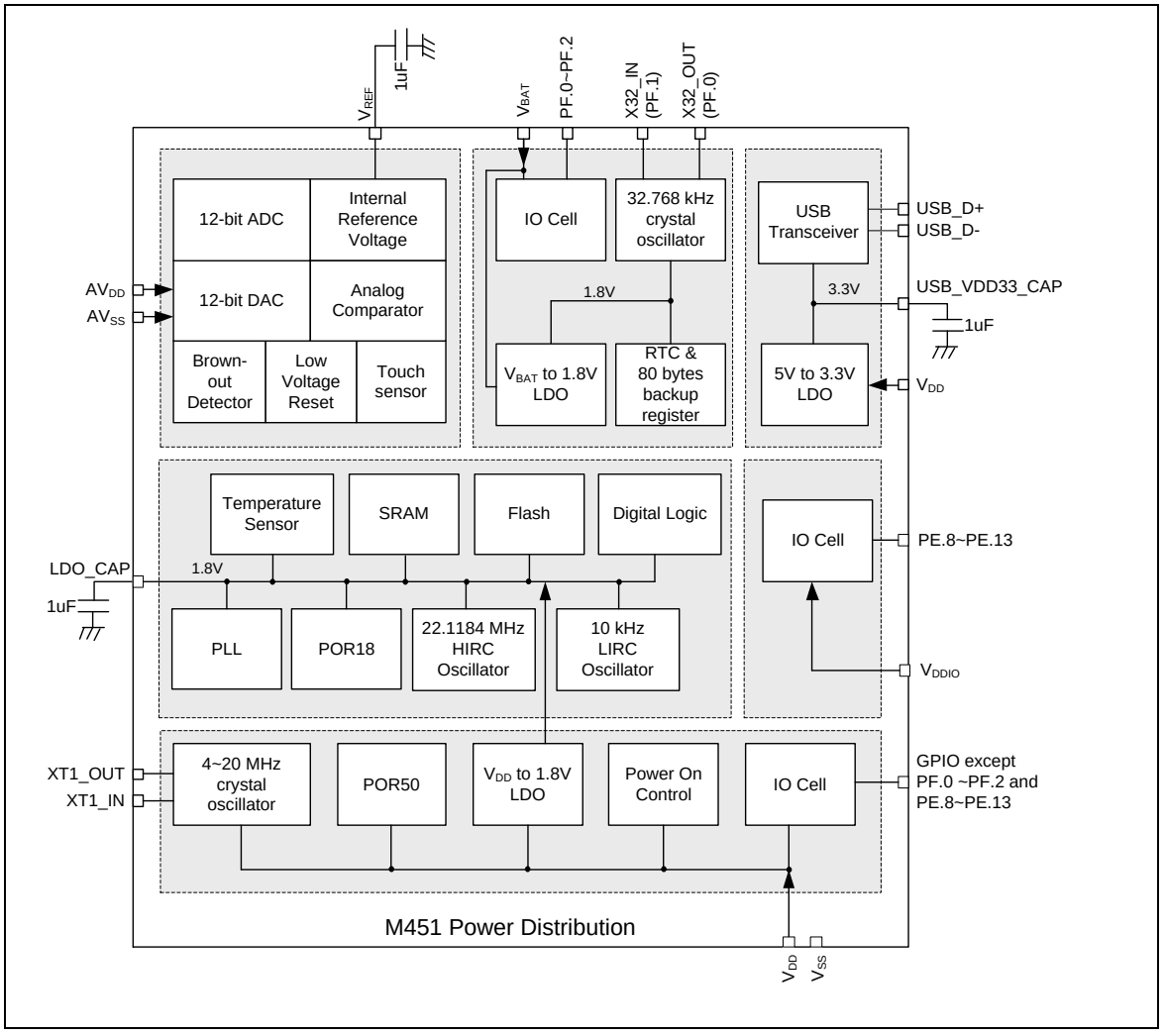


Figure 6.2-7 NuMicro® M4TK Power Distribution Diagram

6.2.5 System Memory Map

The NuMicro® M4TK series provides 4G-byte addressing space. The memory locations assigned to each on-chip controllers are shown in theTable 6-5. The detailed register definition, memory space, and programming will be described in the following sections for each on-chip peripheral. The NuMicro® M4TK series only supports little-endian data format.

Address Space	Token	Controllers
Flash and SRAM Memory Space		
0x0000_0000 – 0x0001_FFFF	FLASH_BA	FLASH Memory Space (128KB) **(M4TKxE Only)**
0x0000_0000 – 0x0003_FFFF	FLASH_BA	FLASH Memory Space (256KB) **(M4TKxG Only)**
0x0004_0000 – 0x0005_FFFF	Reserved	Reserved
0x0006_0000 – 0x0007_FFFF	Reserved	Reserved
0x2000_0000 – 0x2000_3FFF	SRAM0_BA	SRAM Memory Space
0x2000_4000 – 0x2000_7FFF	SRAM1_BA	SRAM Memory Space
0x2000_8000 – 0x2000_BFFF	Reserved	Reserved
0x2000_C000 – 0x2000_FFFF	Reserved	Reserved
0x6000_0000 – 0x6FFF_FFFF	EXTMEM_BA	External Memory Space for EBI Interface (256 MB)
Peripheral Controllers Space (0x4000_0000 – 0x400F_FFFF)		
0x4000_0000 – 0x4000_01FF	SYS_BA	System Control Registers
0x4000_0200 – 0x4000_02FF	CLK_BA	Clock Control Registers
0x4000_0300 – 0x4000_03FF	NMI_BA	NMI Control Registers
0x4000_4000 – 0x4000_4FFF	GPIO_BA	GPIO Control Registers
0x4000_8000 – 0x4000_8FFF	PDMA_BA	Peripheral DMA Control Registers
0x4000_9000 – 0x4000_9FFF	USBH_BA	USB Host Control Registers
0x4000_B000 – 0x4000_BFFF	Reserved	Reserved
0x4000_C000 – 0x4000_CFFF	FMC_BA	Flash Memory Control Registers
0x4000_D000 – 0x4000_DFFF	Reserved	Reserved
0x4001_0000 – 0x4001_0FFF	EBI_BA	External Bus Interface Control Registers
0x4001_9000 – 0x4001_9FFF	Reserved	Reserved
0x4003_0000 – 0x4003_0FFF	Reserved	Reserved
0x4003_1000 – 0x4003_1FFF	CRC_BA	CRC Generator Registers
0x5000_8000 – 0x5000_FFFF	Reserved	Reserved
APB Controllers Space (0x4000_0000 ~ 0x400F_FFFF)		
0x4004_0000 – 0x4004_0FFF	WDT_BA	Watchdog Timer Control Registers
0x4004_1000 – 0x4004_1FFF	RTC_BA	Real Time Clock (RTC) Control Register
0x4004_3000 – 0x4004_3FFF	EADC_BA	Enhanced Analog-Digital-Converter (EADC) Control Registers
0x4004_4000 – 0x4004_4FFF	Reserved	Reserved
0x4004_5000 – 0x4004_5FFF	ACMP01_BA	Analog Comparator 0/ 1 Control Registers

0x4004_6000 – 0x4004_6FFF	Reserved	Reserved
0x4004_7000 – 0x4004_7FFF	DAC_BA	DAC Control Registers
0x4004_8000 – 0x4004_8FFF	Reserved	Reserved
0x4004_9000 – 0x4004_9FFF	Reserved	Reserved
0x4004_D000 – 0x4004_DFFF	OTG_BA	USB OTG Control Register
0x4005_0000 – 0x4005_0FFF	TMR01_BA	Timer0/Timer1 Control Registers
0x4005_1000 – 0x4005_1FFF	TMR23_BA	Timer2/Timer3 Control Registers
0x4005_8000 – 0x4005_8FFF	PWM0_BA	PWM0 Control Registers
0x4005_9000 – 0x4005_9FFF	PWM1_BA	PWM1 Control Registers
0x4005_C000 – 0x4005_CFFF	Reserved	Reserved
0x4005_D000 – 0x4005_DFFF	Reserved	Reserved
0x4006_0000 – 0x4006_0FFF	SPI0_BA	SPI0 Control Registers
0x4006_1000 – 0x4006_1FFF	SPI1_BA	SPI1 Control Registers
0x4006_2000 – 0x4006_2FFF	SPI2_BA	SPI2 Control Registers
0x4006_3000 – 0x4006_3FFF	Reserved	Reserved
0x4007_0000 – 0x4007_0FFF	UART0_BA	UART0 Control Registers
0x4007_1000 – 0x4007_1FFF	UART1_BA	UART1 Control Registers
0x4007_2000 – 0x4007_2FFF	UART2_BA	UART2 Control Registers
0x4007_3000 – 0x4007_3FFF	UART3_BA	UART3 Control Registers
0x4007_4000 – 0x4007_4FFF	Reserved	Reserved
0x4007_5000 – 0x4007_5FFF	Reserved	Reserved
0x4008_0000 – 0x4008_0FFF	I2C0_BA	I^2C0 Control Registers
0x4008_1000 – 0x4008_1FFF	I2C1_BA	I^2C1 Control Registers
0x4008_2000 – 0x4008_2FFF	Reserved	Reserved
0x4008_3000 – 0x4008_3FFF	Reserved	Reserved
0x4008_4000 – 0x4008_4FFF	Reserved	Reserved
0x4009_0000 – 0x4009_0FFF	SC0_BA	Smartcard Host 0 Control Registers
0x4009_1000 – 0x4009_1FFF	Reserved	Reserved
0x4009_2000 – 0x4009_2FFF	Reserved	Reserved
0x4009_3000 – 0x4009_3FFF	Reserved	Reserved
0x4009_4000 – 0x4009_4FFF	Reserved	Reserved
0x4009_5000 – 0x4009_5FFF	Reserved	Reserved
0x400A_0000 – 0x400A_0FFF	CAN0_BA	CAN0 Bus Control Registers
0x400A_1000 – 0x400A_1FFF	Reserved	Reserved
0x400B_0000 – 0x400B_0FFF	Reserved	Reserved

0x400B_1000 – 0x400B_1FFF	Reserved	Reserved
0x400B_0000 – 0x400B_0FFF	Reserved	Reserved
0x400B_1000 – 0x400B_1FFF	Reserved	Reserved
0x400C_0000 – 0x400C_0FFF	USBD_BA	USB Device Control Register
0x400E_0000 – 0x400E_0FFF	Reserved	Reserved
0x400E_2000 – 0x400E_2FFF	TK_BA	Touch Key Control Registers
0x5008_0000 – 0x5008_0FFF	Reserved	Reserved
System Controllers Space (0xE000_E000 ~ 0xE000_EFFF)		
0xE000_E010 – 0xE000_E0FF	SCS_BA	System Timer Control Registers
0xE000_E100 – 0xE000_ECFF	SCS_BA	External Interrupt Controller Control Registers
0xE000_ED00 – 0xE000_ED8F	SCS_BA	System Control Registers

Table 6-5 Address Space Assignments for On-Chip Controllers

6.2.6 SRAM Memory Organization

The M4TK supports embedded SRAM with total 32 KB size and the SRAM organization is separated to two banks: SRAM bank0 and SRAM bank1. Each of these two banks has 16 KB address space and can be accessed simultaneously. The SRAM bank0 supports parity error check to make sure chip operating more stable.

- Supports total 32 KB SRAM
- Supports byte / half word / word write
- Supports fixed 16 KB SRAM bank for independent access
- Supports parity error check function for SRAM bank0
- Supports oversize response error
- Supports remap address to 0x1000_0000

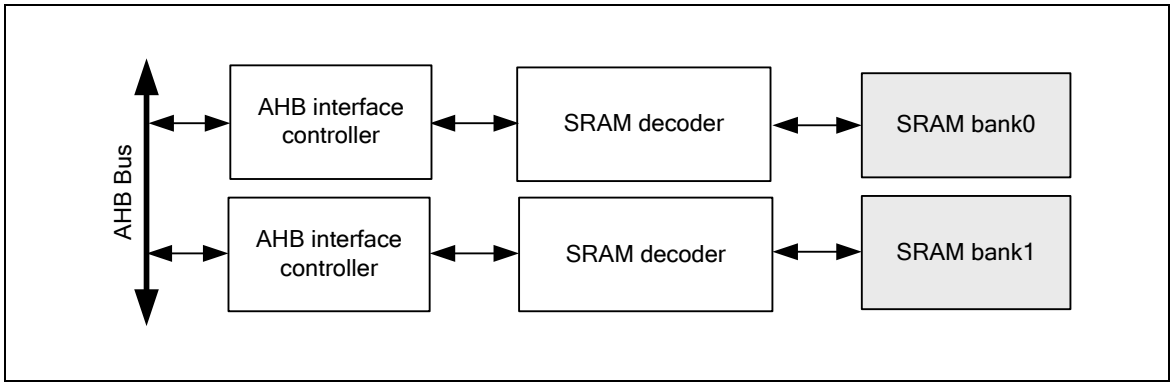


Figure 6.2-8 SRAM Block Diagram

Figure 6.2-9 shows the SRAM organization of M4TK. There are two SRAM banks in M4TK and each bank is addressed to 16 KB. The bank0 address space is from 0x2000_0000 to 0x2000_3FFF. The bank1 address space is from 0x2000_4000 to 0x2000_7FFF. The address between 0x2000_8000 to 0x3FFF_FFFF is illegal memory space and chip will enter hardfault if CPU accesses these illegal memory addresses.

The address of each bank is remapping from 0x2000_0000 to 0x1000_0000. CPU can read SRAM bank0 through 0x2000_0000 to 0x2000_3FFF or 0x1000_0000 to 0x1000_3FFF, and read SRAM bank1 through 0x2000_4000 to 0x2000_7FFF or 0x1000_4000 to 0x1000_7FFF.

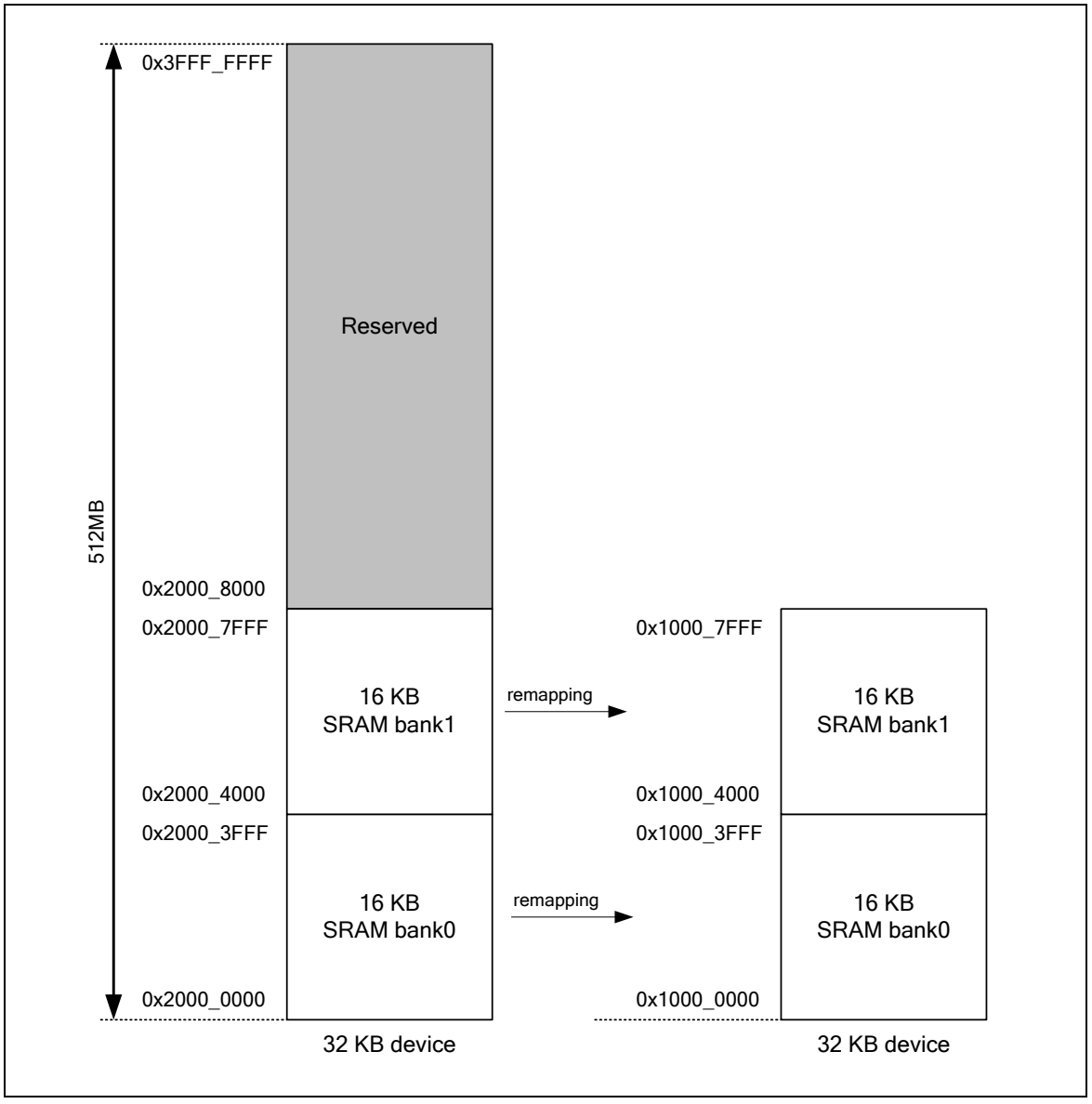


Figure 6.2-9 SRAM Memory Organization

SRAM bank0 has byte parity error check function. When CPU is accessing SRAM bank0, the parity error checking mechanism is dynamic operating. As parity error occurred, the PERRIF (SYS_SRAM_STATUS[0]) will be asserted to 1 and the SYS_SRAM_ERRADDR register will recode the address with parity error. Chip will enter interrupt when SRAM parity error occurred if PERRIEN (SYS_SRAM_INTCTL[0]) is set to 1. When SRAM parity error occurred, chip will stop detecting SRAM parity error until user writes 1 to clear the PERRIF(SYS_SRAM_STATUS[0]) bit.

6.2.7 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 = 0x4000_0000**				
SYS_PDID	SYS_BA+0x00	R	Part Device Identification Number Register	0xXXXX_XXXX [1]
SYS_RSTSTS	SYS_BA+0x04	R/W	System Reset Status Register	0x0000_0043
SYS_IPRST0	SYS_BA+0x08	R/W	Peripheral Reset Control Register 0	0x0000_0000
SYS_IPRST1	SYS_BA+0x0C	R/W	Peripheral Reset Control Register 1	0x0000_0000
SYS_IPRST2	SYS_BA+0x10	R/W	Peripheral Reset Control Register 2	0x0000_0000
SYS_BODCTL	SYS_BA+0x18	R/W	Brown-Out Detector Control Register	0x0000_038X
SYS_IVSCTL	SYS_BA+0x1C	R/W	Internal Voltage Source Control Register	0x0000_0000
SYS_PORCTL	SYS_BA+0x24	R/W	Power-On-Reset Controller Register	0x0000_XXXX
SYS_VREFCTL	SYS_BA+0x28	R/W	V_{REF} Control Register	0x0000_0000
SYS_USBPHY	SYS_BA+0x2C	R/W	USB PHY Control Register	0x0000_0003
SYS_GPA_MFPL	SYS_BA+0x30	R/W	GPIOA Low Byte Multiple Function Control Register	0x0000_0000
SYS_GPA_MFPH	SYS_BA+0x34	R/W	GPIOA High Byte Multiple Function Control Register	0x0000_0000
SYS_GPB_MFPL	SYS_BA+0x38	R/W	GPIOB Low Byte Multiple Function Control Register	0x0000_0000
SYS_GPB_MFPH	SYS_BA+0x3C	R/W	GPIOB High Byte Multiple Function Control Register	0x0000_0000
SYS_GPC_MFPL	SYS_BA+0x40	R/W	GPIOC Low Byte Multiple Function Control Register	0x0000_0000
SYS_GPC_MFPH	SYS_BA+0x44	R/W	GPIOC High Byte Multiple Function Control Register	0x0000_0000
SYS_GPD_MFPL	SYS_BA+0x48	R/W	GPIOD Low Byte Multiple Function Control Register	0x0000_0000
SYS_GPD_MFPH	SYS_BA+0x4C	R/W	GPIOD High Byte Multiple Function Control Register	0x0000_0000
SYS_GPE_MFPL	SYS_BA+0x50	R/W	GPIOE Low Byte Multiple Function Control Register	0x0000_0000
SYS_GPE_MFPH	SYS_BA+0x54	R/W	GPIOE High Byte Multiple Function Control Register	0x0000_0000
SYS_GPF_MFPL	SYS_BA+0x58	R/W	GPIOF Low Byte Multiple Function Control Register	0x0000_0000
SYS_SRAM_INTCTL	SYS_BA+0xC0	R/W	System SRAM Interrupt Enable Control Register	0x0000_0000
SYS_SRAM_STATUS	SYS_BA+0xC4	R	System SRAM Parity Error Status Register	0x0000_0000
SYS_SRAM_ERRADDR	SYS_BA+0xC8	R	System SRAM Parity Check Error Address Register	0x0000_0000
SYS_SRAM_BISTCTL	SYS_BA+0xD0	R/W	System SRAM BIST Test Control Register	0x0000_0000

SYS_SRAM_BISTSTS	SYS_BA+0xD4	R	System SRAM BIST Test Status Register	0x00xx_00xx
SYS_IRCTCTL	SYS_BA+0xF0	R/W	HIRC Trim Control Register	0x0000_0000
SYS_IRCTIEN	SYS_BA+0xF4	R/W	HIRC Trim Interrupt Enable Register	0x0000_0000
SYS_IRCTISTS	SYS_BA+0xF8	R/W	HIRC Trim Interrupt Status Register	0x0000_0000
SYS_REGLCTL	SYS_BA+0x100	R/W	Register Lock Control Register	0x0000_0000
SYS_AHBMCTL	SYS_BA+0x400	R/W	AHB Bus Matrix Priority Control Register	0x0000_0001

6.2.8 Register Description

Part Device Identification Number Register (SYS_PDID)

Register	Offset	R/W	Description	Reset Value
SYS_PDID	SYS_BA+0x00	R	Part Device Identification Number Register	0xXXXX_XXXX [1]

[1] Every part number has a unique default reset value.

31	30	29	28	27	26	25	24
PDID							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PDID							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PDID							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PDID							

Bits	Description	
[31:0]	**PDID**	**Part Device Identification Number (Read Only)** This register reflects device part number code. Software can read this register to identify which device is used.

System Reset Status Register (SYS_RSTSTS)

This register provides specific information for software to identify this chip's reset source from last operation.

Register	Offset	R/W	Description	Reset Value
SYS_RSTSTS	SYS_BA+0x04	R/W	System Reset Status Register	0x0000_0043

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CPURF	**Reserved**	**SYSRF**	**BODRF**	**LVRF**	**WDTRF**	**PINRF**	**PORF**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**CPURF**	**CPU Reset Flag** The CPU reset flag is set by hardware if software writes CPURST (SYS_IPRST0[1]) 1 to reset Cortex®-M4 Core and Flash Memory Controller (FMC). 0 = No reset from CPU. 1 = The Cortex®-M4 Core and FMC are reset by software setting CPURST to 1. **Note:** Write 1 to clear this bit to 0.
[6]	**Reserved**	Reserved.
[5]	**SYSRF**	**System Reset Flag** The system reset flag is set by the "Reset Signal" from the Cortex®-M4 Core to indicate the previous reset source. 0 = No reset from Cortex®-M4. 1 = The Cortex®-M4 had issued the reset signal to reset the system by writing 1 to the bit SYSRESETREQ(AIRCR[2], Application Interrupt and Reset Control Register, address = 0xE000ED0C) in system control registers of Cortex®-M4 core. **Note:** Write 1 to clear this bit to 0.
[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:** Write 1 to clear this bit to 0.

Bits	Description	
[3]	**LVRF**	**LVR Reset Flag** The LVR reset flag is set by the “Reset Signal” from the Low Voltage Reset Controller to indicate the previous reset source. 0 = No reset from LVR. 1 = LVR controller had issued the reset signal to reset the system. **Note:** Write 1 to clear this bit to 0.
[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. **Note1:** Write 1 to clear this bit to 0. **Note2:** Watchdog Timer register RSTF(WDT_CTL[2]) bit is set if the system has been reset by WDT time-out reset. Window Watchdog Timer register WWDTRF(WWDT_STATUS[1]) bit is set if the system has been reset by WWDT time-out reset.
[1]	**PINRF**	**NRESET Pin Reset Flag** The nRESET pin reset flag is set by the “Reset Signal” from the nRESET Pin to indicate the previous reset source. 0 = No reset from nRESET pin. 1 = Pin nRESET had issued the reset signal to reset the system. **Note:** Write 1 to clear this bit to 0.
[0]	**PORF**	**POR Reset Flag** The POR reset flag is set by the “Reset Signal” from the Power-on Reset (POR) Controller or bit CHIPRST (SYS_IPRST0[0]) to indicate the previous reset source. 0 = No reset from POR or CHIPRST. 1 = Power-on Reset (POR) or CHIPRST had issued the reset signal to reset the system. **Note:** Write 1 to clear this bit to 0.

Peripheral Reset Control Register 0 (SYS_IPRST0)

Register	Offset	R/W	Description	Reset Value
SYS_IPRST0	SYS_BA+0x08	R/W	Peripheral Reset Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
CRCRST	Reserved		USBHRST	EBIRST	PDMARST	CPURST	CHIPRST

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**CRCRST**	**CRC Calculation Controller Reset (Write Protect)** Set this bit to 1 will generate a reset signal to the CRC calculation controller. User needs to set this bit to 0 to release from the reset state. 0 = CRC calculation controller normal operation. 1 = CRC calculation controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[6:5]	**Reserved**	Reserved.
[4]	**USBHRST**	**USBH Controller Reset (Write Protect)** Set this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state. 0 = USBH controller normal operation. 1 = USBH controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[3]	**EBIRST**	**EBI Controller Reset (Write Protect)** Set this bit to 1 will generate a reset signal to the EBI. User needs to set this bit to 0 to release from the reset state. 0 = EBI controller normal operation. 1 = EBI controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2]	**PDMARST**	**PDMA Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the PDMA. User needs to set this bit to 0 to release from reset state. 0 = PDMA controller normal operation. 1 = PDMA controller reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[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 after the 2 clock cycles. 0 = Processor core normal operation. 1 = Processor core one-shot reset. **Note:** This bit is write protected. Refer to the SYS_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 after the 2 clock cycles. The CHIPRST is same as the POR reset, all the chip controllers is reset and the chip setting from flash are also reload. About the difference between CHIPRST and SYSRESETREQ(AIRCR[2]), please refer to section 6.2.2 0 = Chip normal operation. 1 = Chip one-shot reset. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

Peripheral Reset Control Register 1 (SYS_IPRST1)

Setting these bits 1 will generate asynchronous reset signals to the corresponding module controller. Users need to set these bits to 0 to release corresponding module controller from reset state.

Register	Offset	R/W	Description	Reset Value
SYS_IPRST1	SYS_BA+0x0C	R/W	Peripheral Reset Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved			EADCRST	USBDRST	OTGRST	Reserved	CAN0RST
23	22	21	20	19	18	17	16
Reserved				UART3RST	UART2RST	UART1RST	UART0RST
15	14	13	12	11	10	9	8
Reserved	SPI2RST	SPI1RST	SPI0RST	Reserved		I2C1RST	I2C0RST
7	6	5	4	3	2	1	0
ACMP01RST	Reserved	TMR3RST	TMR2RST	TMR1RST	TMR0RST	GPIORST	Reserved

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28]	**EADCRST**	**EADC Controller Reset** 0 = EADC controller normal operation. 1 = EADC controller reset.
[27]	**USBDRST**	**USB Device Controller Reset** 0 = USB device controller normal operation. 1 = USB device controller reset.
[26]	**OTGRST**	**OTG Controller Reset** 0 = OTG controller normal operation. 1 = OTG controller reset.
[25]	**Reserved**	Reserved.
[24]	**CAN0RST**	**CAN0 Controller Reset** 0 = CAN0 controller normal operation. 1 = CAN0 controller reset.
[23:20]	**Reserved**	Reserved.
[19]	**UART3RST**	**UART3 Controller Reset** 0 = UART3 controller normal operation. 1 = UART3 controller reset.
[18]	**UART2RST**	**UART2 Controller Reset** 0 = UART2 controller normal operation. 1 = UART2 controller reset.
[17]	**UART1RST**	**UART1 Controller Reset** 0 = UART1 controller normal operation.

		1 = UART1 controller reset.
[16]	**UART0RST**	**UART0 Controller Reset** 0 = UART0 controller normal operation. 1 = UART0 controller reset.
[15]	**Reserved**	Reserved.
[14]	**SPI2RST**	**SPI2 Controller Reset** 0 = SPI2 controller normal operation. 1 = SPI2 controller reset.
[13]	**SPI1RST**	**SPI1 Controller Reset** 0 = SPI1 controller normal operation. 1 = SPI1 controller reset.
[12]	**SPI0RST**	**SPI0 Controller Reset** 0 = SPI0 controller normal operation. 1 = SPI0 controller reset.
[11:10]	**Reserved**	Reserved.
[9]	**I2C1RST**	**I2C1 Controller Reset** 0 = I2C1 controller normal operation. 1 = I2C1 controller reset.
[8]	**I2C0RST**	**I2C0 Controller Reset** 0 = I2C0 controller normal operation. 1 = I2C0 controller reset.
[7]	**ACMP01RST**	**Analog Comparator 0/1 Controller Reset** 0 = Analog Comparator 0/1 controller normal operation. 1 = Analog Comparator 0/1 controller reset.
[6]	**Reserved**	Reserved.
[5]	**TMR3RST**	**Timer3 Controller Reset** 0 = Timer3 controller normal operation. 1 = Timer3 controller reset.
[4]	**TMR2RST**	**Timer2 Controller Reset** 0 = Timer2 controller normal operation. 1 = Timer2 controller reset.
[3]	**TMR1RST**	**Timer1 Controller Reset** 0 = Timer1 controller normal operation. 1 = Timer1 controller reset.
[2]	**TMR0RST**	**Timer0 Controller Reset** 0 = Timer0 controller normal operation. 1 = Timer0 controller reset.
[1]	**GPIORST**	**GPIO Controller Reset** 0 = GPIO controller normal operation. 1 = GPIO controller reset.
[0]	**Reserved**	Reserved.

Peripheral Reset Control Register 2 (SYS_IPRST2)

Setting these bits to 1 will generate asynchronous reset signals to the corresponding module controller. Users need to set these bits to 0 to release corresponding module controller from reset state.

Register	Offset	R/W	Description	Reset Value
SYS_IPRST2	SYS_BA+0x10	R/W	Peripheral Reset Control Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						TKRST	Reserved
23	22	21	20	19	18	17	16
Reserved						PWM1RST	PWM0RST
15	14	13	12	11	10	9	8
Reserved			DACRST	Reserved			
7	6	5	4	3	2	1	0
Reserved							SC0RST

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**TKRST**	**Touch Key Controller Reset** 0 = Touch Key controller normal operation. 1 = Touch Key controller reset.
[24:18]	**Reserved**	Reserved.
[17]	**PWM1RST**	**PWM1 Controller Reset** 0 = PWM1 controller normal operation. 1 = PWM1 controller reset.
[16]	**PWM0RST**	**PWM0 Controller Reset** 0 = PWM0 controller normal operation. 1 = PWM0 controller reset.
[15:13]	**Reserved**	Reserved.
[12]	**DACRST**	**DAC Controller Reset** 0 = DAC controller normal operation. 1 = DAC controller reset.
[11:1]	**Reserved**	Reserved.
[0]	**SC0RST**	**SC0 Controller Reset** 0 = SC0 controller normal operation. 1 = SC0 controller reset.

Brown-out Detector Control Register (SYS_BODCTL)

Partial of the SYS_BODCTL control registers values are initiated by the flash configuration and partial bits are write-protected bit.

Register	Offset	R/W	Description	Reset Value
SYS_BODCTL	SYS_BA+0x18	R/W	Brown-Out Detector Control Register	0x0000_038X

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved	LVRDGSEL			Reserved	BODDGSEL		
7	6	5	4	3	2	1	0
LVREN	BODOUT	BODLPM	BODIF	BODRSTEN	BODVL		BODEN

Bits	Description	
[31:15]	**Reserved**	Reserved.
[14:12]	**LVRDGSEL**	**LVR Output De-glitch Time Select (Write Protect)** 000 = Without de-glitch function. 001 = 4 system clock (HCLK). 010 = 8 system clock (HCLK). 011 = 16 system clock (HCLK). 100 = 32 system clock (HCLK). 101 = 64 system clock (HCLK). 110 = 128 system clock (HCLK). 111 = 256 system clock (HCLK). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.
[11]	**Reserved**	Reserved.
[10:8]	**BODDGSEL**	**Brown-out Detector Output De-glitch Time Select (Write Protect)** 000 = BOD output is sampled by RC10K clock. 001 = 4 system clock (HCLK). 010 = 8 system clock (HCLK). 011 = 16 system clock (HCLK). 100 = 32 system clock (HCLK). 101 = 64 system clock (HCLK). 110 = 128 system clock (HCLK). 111 = 256 system clock (HCLK). **Note:** These bits are write protected. Refer to the SYS_REGLCTL register.

Bits	Description	
[7]	**LVREN**	**Low Voltage Reset Enable Bit (Write Protect)** The LVR function resets the chip when the input power voltage is lower than LVR circuit setting. LVR function is enabled by default. 0 = Low Voltage Reset function Disabled. 1 = Low Voltage Reset function Enabled. **Note1:** After enabling the bit, the LVR function will be active with 100us delay for LVR output stable (default). **Note2:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note3:** LIRC must be enabled before enable LVR.
[6]	**BODOUT**	**Brown-out Detector Output Status** 0 = Brown-out Detector output status is 0. It means the detected voltage is higher than BODVL setting or BODEN is 0. 1 = Brown-out Detector output status is 1. It means the detected voltage is lower than BODVL setting. If the BODEN is 0, BOD function disabled , this bit always responds 0.
[5]	**BODLPM**	**Brown-out Detector Low Power Mode (Write Protect)** 0 = BOD operate in normal mode (default). 1 = BOD Low Power mode Enabled. **Note1:** The BOD consumes about 100uA in normal mode, the low power mode can reduce the current to about 1/10 but slow the BOD response. **Note2:** This bit is write protected. Refer to the SYS_REGLCTL register.
[4]	**BODIF**	**Brown-out Detector Interrupt Flag** 0 = Brown-out Detector does not detect any voltage draft at V_{DD} down through or up through the voltage of BODVL setting. 1 = When Brown-out Detector detects the V_{DD} is dropped down through the voltage of BODVL setting or the V_{DD} 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. **Note:** Write 1 to clear this bit to 0.
[3]	**BODRSTEN**	**Brown-out Reset Enable Bit (Write Protect)** The default value is set by flash controller user configuration register CBORST(CONFIG0[20]) bit . 0 = Brown-out “INTERRUPT” function Enabled. 1 = Brown-out “RESET” function Enabled. **Note1:** While the Brown-out Detector function is enabled (BODEN high) and BOD reset function is enabled (BODRSTEN high), BOD will assert a signal to reset chip when the detected voltage is lower than the threshold (BODOUT high). While the BOD function is enabled (BODEN high) and BOD interrupt function is enabled (BODRSTEN low), BOD will assert an interrupt if BODOUT is high. BOD interrupt will keep till to the BODEN set to 0. BOD interrupt can be blocked by disabling the NVIC BOD interrupt or disabling BOD function (set BODEN low). **Note2:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2:1]	**BODVL**	**Brown-out Detector Threshold Voltage Selection (Write Protect)** The default value is set by flash controller user configuration register CBOV (CONFIG0 [22:21]). 00 = Brown-Out Detector threshold voltage is 2.2V. 01 = Brown-Out Detector threshold voltage is 2.7V. 10 = Brown-Out Detector threshold voltage is 3.7V. 11 = Brown-Out Detector threshold voltage is 4.4V. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

Bits	Description	
[0]	**BODEN**	**Brown-out Detector Enable Bit (Write Protect)** The default value is set by flash controller user configuration register CBODEN (CONFIG0 [23]). 0 = Brown-out Detector function Disabled. 1 = Brown-out Detector function Enabled. **Note1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note2:** LIRC must be enabled before enable BOD.

Internal Voltage Source Control Register (SYS_IVSCTL)

Register	Offset	R/W	Description	Reset Value
SYS_IVSCTL	SYS_BA+0x1C	R/W	Internal Voltage Source Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						VBATUGEN	VTEMPEN

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**VBATUGEN**	**VBAT Unity Gain Buffer Enable Bit** This bit is used to enable/disable VBAT unity gain buffer function. 0 = VBAT unity gain buffer function Disabled (default). 1 = VBAT unity gain buffer function Enabled. **Note:** After this bit is set to 1, the value of VBAT unity gain buffer output voltage can be obtained from ADC conversion result
[0]	**VTEMPEN**	**Temperature Sensor Enable Bit** This bit is used to enable/disable temperature sensor function. 0 = Temperature sensor function Disabled (default). 1 = Temperature sensor function Enabled. **Note:** After this bit is set to 1, the value of temperature sensor output can be obtained from ADC conversion result. Please refer to ADC function chapter for details.

Power-on Reset Controller Register (SYS_PORCTL)

Register	Offset	R/W	Description	Reset Value
SYS_PORCTL	SYS_BA+0x24	R/W	Power-On-Reset Controller Register	0x0000_XXXX

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
POROFF							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
POROFF							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**POROFF**	**Power-on Reset Enable Bit (Write Protect)** When powered on, the POR circuit generates a reset signal to reset the whole chip function, but noise on the power may cause the POR active again. User can disable internal POR circuit to avoid unpredictable noise to cause chip reset by writing 0x5AA5 to this field. The POR function will be active again when this field is set to another value or chip is reset by other reset source, including: nRESET, Watchdog, LVR reset, BOD reset, ICE reset command and the software-chip reset function. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

VREF Control Register (SYS_VREFCTL)

Register	Offset	R/W	Description	Reset Value
SYS_VREFCTL	SYS_BA+0x28	R/W	VREF Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			VREFCTL				

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4:0]	**VREFCTL**	**V_{REF} Control Bits (Write Protect)** 00000 = V_{REF} is from external pin. 00011 = V_{REF} is internal 2.56V. 00111 = V_{REF} is internal 2.048V. 01011 = V_{REF} is internal 3.072V. 01111 = V_{REF} is internal 4.096V. Others = Reserved. **Note1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note2:** Connecting a 1uF capacitor to AV_{SS} will make internal reference voltage more stable.

USB PHY Control Register (SYS_USBPHY)

Register	Offset	R/W	Description	Reset Value
SYS_USBPHY	SYS_BA+0x2C	R/W	USB PHY Control Register	0x0000_0003

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							LDO33EN
7	6	5	4	3	2	1	0
Reserved						USBROLE	

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**LDO33EN**	**USB LDO33 Enable Bit (Write Protect)** 0 = USB LDO33 Disabled. 1 = USB LDO33 Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[7:2]	**Reserved**	Reserved.
[1:0]	**USBROLE**	**USB Role Option (Write Protect)** These two bits are used to select the role of USB. 00 = Standard USB Device mode. 01 = Standard USB Host mode. 10 = ID dependent mode. 11 = On-The-Go device mode. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

GPIOA Low Byte Multiple Function Control Register (SYS_GPA_MFPL)

Please refer to 4.3.4 GPIO Multi-function Pin Summary.

Register	Offset	R/W	Description	Reset Value
SYS_GPA_MFPL	SYS_BA+0x30	R/W	GPIOA Low Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PA7MFP				PA6MFP			
23	22	21	20	19	18	17	16
PA5MFP				PA4MFP			
15	14	13	12	11	10	9	8
PA3MFP				PA2MFP			
7	6	5	4	3	2	1	0
PA1MFP				PA0MFP			

Bits	Description	
[31:28]	PA7MFP	PA.7 Multi-function Pin Selection
[27:24]	PA6MFP	PA.6 Multi-function Pin Selection
[23:20]	PA5MFP	PA.5 Multi-function Pin Selection
[19:16]	PA4MFP	PA.4 Multi-function Pin Selection
[15:12]	PA3MFP	PA.3 Multi-function Pin Selection
[11:8]	PA2MFP	PA.2 Multi-function Pin Selection
[7:4]	PA1MFP	PA.1 Multi-function Pin Selection
[3:0]	PA0MFP	PA.0 Multi-function Pin Selection

GPIOA High Byte Multiple Function Control Register (SYS_GPA_MFPH)

Please refer to 4.3.4 GPIO Multi-function Pin Summary.

Register	Offset	R/W	Description	Reset Value
SYS_GPA_MFPH	SYS_BA+0x34	R/W	GPIOA High Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PA15MFP				PA14MFP			
23	22	21	20	19	18	17	16
PA13MFP				PA12MFP			
15	14	13	12	11	10	9	8
PA11MFP				PA10MFP			
7	6	5	4	3	2	1	0
PA9MFP				PA8MFP			

Bits	Description	
[31:28]	PA15MFP	PA.15 Multi-function Pin Selection
[27:24]	PA14MFP	PA.14 Multi-function Pin Selection
[23:20]	PA13MFP	PA.13 Multi-function Pin Selection
[19:16]	PA12MFP	PA.12 Multi-function Pin Selection
[15:12]	PA11MFP	PA.11 Multi-function Pin Selection
[11:8]	PA10MFP	PA.10 Multi-function Pin Selection
[7:4]	PA9MFP	PA.9 Multi-function Pin Selection
[3:0]	PA8MFP	PA.8 Multi-function Pin Selection

GPIOB Low Byte Multiple Function Control Register (SYS_GPB_MFPL)

Please refer to 4.3.4 GPIO Multi-function Pin Summary.

Register	Offset	R/W	Description	Reset Value
SYS_GPB_MFPL	SYS_BA+0x38	R/W	GPIOB Low Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PB7MFP				PB6MFP			
23	22	21	20	19	18	17	16
PB5MFP				PB4MFP			
15	14	13	12	11	10	9	8
PB3MFP				PB2MFP			
7	6	5	4	3	2	1	0
PB1MFP				PB0MFP			

Bits	Description	
[31:28]	PB7MFP	PB.7 Multi-function Pin Selection
[27:24]	PB6MFP	PB.6 Multi-function Pin Selection
[23:20]	PB5MFP	PB.5 Multi-function Pin Selection
[19:16]	PB4MFP	PB.4 Multi-function Pin Selection
[15:12]	PB3MFP	PB.3 Multi-function Pin Selection
[11:8]	PB2MFP	PB.2 Multi-function Pin Selection
[7:4]	PB1MFP	PB.1 Multi-function Pin Selection
[3:0]	PB0MFP	PB.0 Multi-function Pin Selection

GPIOB High Byte Multiple Function Control Register (SYS_GPB_MFPH)

Please refer to 4.3.4 GPIO Multi-function Pin Summary.

Register	Offset	R/W	Description	Reset Value
SYS_GPB_MFPH	SYS_BA+0x3C	R/W	GPIOB High Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PB15MFP				PB14MFP			
23	22	21	20	19	18	17	16
PB13MFP				PB12MFP			
15	14	13	12	11	10	9	8
PB11MFP				PB10MFP			
7	6	5	4	3	2	1	0
PB9MFP				PB8MFP			

Bits	Description	
[31:28]	PB15MFP	PB.15 Multi-function Pin Selection
[27:24]	PB14MFP	PB.14 Multi-function Pin Selection
[23:20]	PB13MFP	PB.13 Multi-function Pin Selection
[19:16]	PB12MFP	PB.12 Multi-function Pin Selection
[15:12]	PB11MFP	PB.11 Multi-function Pin Selection
[11:8]	PB10MFP	PB.10 Multi-function Pin Selection
[7:4]	PB9MFP	PB.9 Multi-function Pin Selection
[3:0]	PB8MFP	PB.8 Multi-function Pin Selection

GPIOC Low Byte Multiple Function Control Register (SYS_GPC_MFPL)

Please refer to 4.3.4 GPIO Multi-function Pin Summary.

Register	Offset	R/W	Description	Reset Value
SYS_GPC_MFPL	SYS_BA+0x40	R/W	GPIOC Low Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PC7MFP				PC6MFP			
23	22	21	20	19	18	17	16
PC5MFP				PC4MFP			
15	14	13	12	11	10	9	8
PC3MFP				PC2MFP			
7	6	5	4	3	2	1	0
PC1MFP				PC0MFP			

Bits	Description	
[31:28]	PC7MFP	PC.7 Multi-function Pin Selection
[27:24]	PC6MFP	PC.6 Multi-function Pin Selection
[23:20]	PC5MFP	PC.5 Multi-function Pin Selection
[19:16]	PC4MFP	PC.4 Multi-function Pin Selection
[15:12]	PC3MFP	PC.3 Multi-function Pin Selection
[11:8]	PC2MFP	PC.2 Multi-function Pin Selection
[7:4]	PC1MFP	PC.1 Multi-function Pin Selection
[3:0]	PC0MFP	PC.0 Multi-function Pin Selection

GPIOC High Byte Multiple Function Control Register (SYS_GPC_MFPH)

Please refer to 4.3.4 GPIO Multi-function Pin Summary.

Register	Offset	R/W	Description	Reset Value
SYS_GPC_MFPH	SYS_BA+0x44	R/W	GPIOC High Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PC15MFP				PC14MFP			
23	22	21	20	19	18	17	16
PC13MFP				PC12MFP			
15	14	13	12	11	10	9	8
PC11MFP				PC10MFP			
7	6	5	4	3	2	1	0
PC9MFP				PC8MFP			

Bits	Description	
[31:28]	PC15MFP	PC.15 Multi-function Pin Selection
[27:24]	PC14MFP	PC.14 Multi-function Pin Selection
[23:20]	PC13MFP	PC.13 Multi-function Pin Selection
[19:16]	PC12MFP	PC.12 Multi-function Pin Selection
[15:12]	PC11MFP	PC.11 Multi-function Pin Selection
[11:8]	PC10MFP	PC.10 Multi-function Pin Selection
[7:4]	PC9MFP	PC.9 Multi-function Pin Selection
[3:0]	PC8MFP	PC.8 Multi-function Pin Selection

GPIOD Low Byte Multiple Function Control Register (SYS_GPD_MFPL)

Please refer to 4.3.4 GPIO Multi-function Pin Summary.

Register	Offset	R/W	Description	Reset Value
SYS_GPD_MFPL	SYS_BA+0x48	R/W	GPIOD Low Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PD7MFP				PD6MFP			
23	22	21	20	19	18	17	16
PD5MFP				PD4MFP			
15	14	13	12	11	10	9	8
PD3MFP				PD2MFP			
7	6	5	4	3	2	1	0
PD1MFP				PD0MFP			

Bits	Description	
[31:28]	PD7MFP	PD.7 Multi-function Pin Selection
[27:24]	PD6MFP	PD.6 Multi-function Pin Selection
[23:20]	PD5MFP	PD.5 Multi-function Pin Selection
[19:16]	PD4MFP	PD.4 Multi-function Pin Selection
[15:12]	PD3MFP	PD.3 Multi-function Pin Selection
[11:8]	PD2MFP	PD.2 Multi-function Pin Selection
[7:4]	PD1MFP	PD.1 Multi-function Pin Selection
[3:0]	PD0MFP	PD.0 Multi-function Pin Selection

GPIOD High Byte Multiple Function Control Register (SYS_GPD_MFPH)

Please refer to 4.3.4 GPIO Multi-function Pin Summary

Register	Offset	R/W	Description	Reset Value
SYS_GPD_MFPH	SYS_BA+0x4C	R/W	GPIOD High Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PD15MFP				PD14MFP			
23	22	21	20	19	18	17	16
PD13MFP				PD12MFP			
15	14	13	12	11	10	9	8
PD11MFP				PD10MFP			
7	6	5	4	3	2	1	0
PD9MFP				PD8MFP			

Bits	Description	
[31:28]	PD15MFP	PD.15 Multi-function Pin Selection
[27:24]	PD14MFP	PD.14 Multi-function Pin Selection
[23:20]	PD13MFP	PD.13 Multi-function Pin Selection
[19:16]	PD12MFP	PD.12 Multi-function Pin Selection
[15:12]	PD11MFP	PD.11 Multi-function Pin Selection
[11:8]	PD10MFP	PD.10 Multi-function Pin Selection
[7:4]	PD9MFP	PD.9 Multi-function Pin Selection
[3:0]	PD8MFP	PD.8 Multi-function Pin Selection

GPIOE Low Byte Multiple Function Control Register (SYS_GPE_MFPL)

Please refer to 4.3.4 GPIO Multi-function Pin Summary.

Register	Offset	R/W	Description	Reset Value
SYS_GPE_MFPL	SYS_BA+0x50	R/W	GPIOE Low Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PE7MFP				PE6MFP			
23	22	21	20	19	18	17	16
PE5MFP				PE4MFP			
15	14	13	12	11	10	9	8
PE3MFP				PE2MFP			
7	6	5	4	3	2	1	0
PE1MFP				PE0MFP			

Bits	Description	
[31:28]	PE7MFP	PE.7 Multi-function Pin Selection
[27:24]	PE6MFP	PE.6 Multi-function Pin Selection
[23:20]	PE5MFP	PE.5 Multi-function Pin Selection
[19:16]	PE4MFP	PE.4 Multi-function Pin Selection
[15:12]	PE3MFP	PE.3 Multi-function Pin Selection
[11:8]	PE2MFP	PE.2 Multi-function Pin Selection
[7:4]	PE1MFP	PE.1 Multi-function Pin Selection
[3:0]	PE0MFP	PE.0 Multi-function Pin Selection

GPIOE High Byte Multiple Function Control Register (SYS_GPE_MFPH)

Please refer to 4.3.4 GPIO Multi-function Pin Summary.

Register	Offset	R/W	Description	Reset Value
SYS_GPE_MFPH	SYS_BA+0x54	R/W	GPIOE High Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				PE14_MFP			
23	22	21	20	19	18	17	16
PE13MFP				PE12MFP			
15	14	13	12	11	10	9	8
PE11MFP				PE10MFP			
7	6	5	4	3	2	1	0
PE9MFP				PE8MFP			

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:24]	**PE14_MFP**	**PE.14 Multi-function Pin Selection**
[23:20]	**PE13MFP**	**PE.13 Multi-function Pin Selection**
[19:16]	**PE12MFP**	**PE.12 Multi-function Pin Selection**
[15:12]	**PE11MFP**	**PE.11 Multi-function Pin Selection**
[11:8]	**PE10MFP**	**PE.10 Multi-function Pin Selection**
[7:4]	**PE9MFP**	**PE.9 Multi-function Pin Selection**
[3:0]	**PE8MFP**	**PE.8 Multi-function Pin Selection**

GPIOF Low Byte Multiple Function Control Register (SYS_GPF_MFPL)

Please refer to 4.3.4 GPIO Multi-function Pin Summary.

Register	Offset	R/W	Description	Reset Value
SYS_GPF_MFPL	SYS_BA+0x58	R/W	GPIOF Low Byte Multiple Function Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PF7MFP				PF6MFP			
23	22	21	20	19	18	17	16
PF5MFP				PF4MFP			
15	14	13	12	11	10	9	8
PF3MFP				PF2MFP			
7	6	5	4	3	2	1	0
PF1MFP				PF0MFP			

Bits	Description	
[31:28]	PF7MFP	PF.7 Multi-function Pin Selection
[27:24]	PF6MFP	PF.6 Multi-function Pin Selection
[23:20]	PF5MFP	PF.5 Multi-function Pin Selection
[19:16]	PF4MFP	PF.4 Multi-function Pin Selection
[15:12]	PF3MFP	PF.3 Multi-function Pin Selection
[11:8]	PF2MFP	PF.2 Multi-function Pin Selection
[7:4]	PF1MFP	PF.1 Multi-function Pin Selection
[3:0]	PF0MFP	PF.0 Multi-function Pin Selection

System SRAM Parity Error Interrupt Enable Control Register (SYS_SRAM_INTCTL)

Register	Offset	R/W	Description	Reset Value
SYS_SRAM_INTCTL	SYS_BA+0xC0	R/W	System SRAM Interrupt Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							PERRIEN

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**PERRIEN**	**SRAM Parity Check Error Interrupt Enable Bit** 0 = SRAM parity check error interrupt Disabled. 1 = SRAM parity check error interrupt Enabled.

System SRAM Parity Check Status Register (SYS_SRAM_STATUS)

Register	Offset	R/W	Description	Reset Value
SYS_SRAM_STATUS	SYS_BA+0xC4	R	System SRAM Parity Error Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							PERRIF

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**PERRIF**	**SRAM Parity Check Error Flag** 0 = No System SRAM parity error. 1 = System SRAM parity error occur.

System SRAM Parity Error Address Register (SYS_SRAM_ERRADDR)

Register	Offset	R/W	Description	Reset Value
SYS_SRAM_ERRADDR	SYS_BA+0xC8	R	System SRAM Parity Check Error Address Register	0x0000_0000

31	30	29	28	27	26	25	24
ERRADDR							
23	22	21	20	19	18	17	16
ERRADDR							
15	14	13	12	11	10	9	8
ERRADDR							
7	6	5	4	3	2	1	0
ERRADDR							

Bits	Description	
[31:0]	ERRADDR	**System SRAM Parity Error Address** This register shows system SRAM parity error byte address.

System SRAM BIST Test Control Register (SYS_SRAM_BISTCTL)

Register	Offset	R/W	Description	Reset Value
SYS_SRAM_BISTCTL	SYS_BA+0xD0	R/W	System SRAM BIST Test Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			USBBIST	CANBIST	CRBIST	SRBIST1	SRBIST0

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**USBBIST**	**USB BIST Enable Bit (Write Protect)** This bit enables BIST test for USB RAM 0 = system USB BIST Disabled. 1 = system USB BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[3]	**CANBIST**	**CAN BIST Enable Bit (Write Protect)** This bit enables BIST test for CAN RAM 0 = system CAN BIST Disabled. 1 = system CAN BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2]	**CRBIST**	**CACHE BIST Enable Bit (Write Protect)** This bit enables BIST test for CACHE RAM 0 = system CACHE BIST Disabled. 1 = system CACHE BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[1]	**SRBIST1**	**2nd SRAM BIST Enable Bit (Write Protect)** This bit enables BIST test for SRAM located in address 0x2000_4000 ~0x2000_7FFF 0 = system SRAM BIST Disabled. 1 = system SRAM BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[0]	**SRBIST0**	**1st SRAM BIST Enable Bit (Write Protect)** This bit enables BIST test for SRAM located in address 0x2000_0000 ~0x2000_3FFF 0 = system SRAM BIST Disabled. 1 = system SRAM BIST Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

System SRAM BIST Test Status Register (SYS_SRAM_BISTSTS)

Register	Offset	R/W	Description	Reset Value
SYS_SRAM_BISTSTS	SYS_BA+0xD4	R	System SRAM BIST Test Status Register	0x00xx_00xx

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			USBBEND	CANBEND	CRBEND	SRBEND1	SRBEND0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			USBBEF	CANBEF	CRBISTEF	SRBISTEF1	SRBISTEF0

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20]	**USBBEND**	**USB SRAM BIST Test Finish** 0 = USB SRAM BIST is active. 1 = USB SRAM BIST test finish.
[19]	**CANBEND**	**CAN SRAM BIST Test Finish** 0 = CAN SRAM BIST is active. 1 = CAN SRAM BIST test finish.
[18]	**CRBEND**	**CACHE SRAM BIST Test Finish** 0 = System CACHE RAM BIST is active. 1 = System CACHE RAM BIST test finish.
[17]	**SRBEND1**	**2nd SRAM BIST Test Finish** 0 = 2nd system SRAM BIST is active. 1 = 2nd system SRAM BIST finish.
[16]	**SRBEND0**	**1st SRAM BIST Test Finish** 0 = 1st system SRAM BIST active. 1 =1st system SRAM BIST finish.
[15:5]	**Reserved**	Reserved.
[4]	**USBBEF**	**USB SRAM BIST Fail Flag** 0 = USB SRAM BIST test pass. 1 = USB SRAM BIST test fail.
[3]	**CANBEF**	**CAN SRAM BIST Fail Flag** 0 = CAN SRAM BIST test pass. 1 = CAN SRAM BIST test fail.
[2]	**CRBISTEF**	**CACHE SRAM BIST Fail Flag** 0 = System CACHE RAM BIST test pass.

		1 = System CACHE RAM BIST test fail.
[1]	**SRBISTEF1**	**2nd System SRAM BIST Fail Flag** 0 = 2nd system SRAM BIST test pass. 1 = 2nd system SRAM BIST test fail.
[0]	**SRBISTEF0**	**1st System SRAM BIST Fail Flag** 0 = 1st system SRAM BIST test pass. 1 = 1st system SRAM BIST test fail.

HIRC Trim Control Register (SYS_IRCTCTL)

Register	Offset	R/W	Description	Reset Value
SYS_IRCTCTL	SYS_BA+0xF0	R/W	HIRC Trim Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							CESTOPEN
7	6	5	4	3	2	1	0
RETRYCNT		LOOPSEL		Reserved		FREQSEL	

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**CESTOPEN**	**Clock Error Stop Enable Bit** 0 = The trim operation is keep going if clock is inaccuracy. 1 = The trim operation is stopped if clock is inaccuracy.
[7:6]	**RETRYCNT**	**Trim Value Update Limitation Count** This field defines that how many times the auto trim circuit will try to update the HIRC trim value before the frequency of HIRC locked. Once the HIRC locked, the internal trim value update counter will be reset. If the trim value update counter reached this limitation value and frequency of HIRC still doesn't lock, the auto trim operation will be disabled and FREQSEL will be cleared to 00. 00 = Trim retry count limitation is 64 loops. 01 = Trim retry count limitation is 128 loops. 10 = Trim retry count limitation is 256 loops. 11 = Trim retry count limitation is 512 loops.
[5:4]	**LOOPSEL**	**Trim Calculation Loop Selection** This field defines that trim value calculation is based on how many 32.768 kHz clock. 00 = Trim value calculation is based on average difference in 4 32.768 kHz clock. 01 = Trim value calculation is based on average difference in 8 32.768 kHz clock. 10 = Trim value calculation is based on average difference in 16 32.768 kHz clock. 11 = Trim value calculation is based on average difference in 32 32.768 kHz clock. **Note:** For example, if LOOPSEL is set as 00, auto trim circuit will calculate trim value based on the average frequency difference in 4 32.768 kHz clock.
[3:2]	**Reserved**	Reserved.
[1:0]	**FREQSEL**	**Trim Frequency Selection** This field indicates the target frequency of 22.1184 MHz internal high speed RC oscillator (HIRC) auto trim. During auto trim operation, if clock error detected with CESTOPEN is set to 1 or trim retry limitation count reached, this field will be cleared to 00 automatically.

		00 = Disable HIRC auto trim function. 01 = Enable HIRC auto trim function and trim HIRC to 22.1184 MHz. Others = Reserved. **Note:** HIRC auto trim cannot work normally at power down mode. These bits must be cleared before entering power down mode.

HIRC Trim Interrupt Enable Register (SYS_IRCTIEN)

Register	Offset	R/W	Description	Reset Value
SYS_IRCTIEN	SYS_BA+0xF4	R/W	HIRC Trim Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					CLKEIEN	TFAILIEN	Reserved

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**CLKEIEN**	**Clock Error Interrupt Enable Bit** This bit controls if CPU would get an interrupt while clock is inaccuracy during auto trim operation. If this bit is set to1, and CLKERRIF(SYS_IRCTSTS[2]) is set during auto trim operation, an interrupt will be triggered to notify the clock frequency is inaccuracy. 0 = Disable CLKERRIF(SYS_IRCTSTS[2]) status to trigger an interrupt to CPU. 1 = Enable CLKERRIF(SYS_IRCTSTS[2]) status to trigger an interrupt to CPU.
[1]	**TFAILIEN**	**Trim Failure Interrupt Enable Bit** This bit controls if an interrupt will be triggered while HIRC trim value update limitation count reached and HIRC frequency still not locked on target frequency set by FREQSEL(SYS_IRCTCTL[1:0]). If this bit is high and TFAILIF(SYS_IRCTSTS[1]) is set during auto trim operation, an interrupt will be triggered to notify that HIRC trim value update limitation count was reached. 0 = Disable TFAILIF(SYS_IRCTSTS[1]) status to trigger an interrupt to CPU. 1 = Enable TFAILIF(SYS_IRCTSTS[1]) status to trigger an interrupt to CPU.
[0]	**Reserved**	Reserved.

HIRC Trim Interrupt Status Register (SYS_IRCTISTS)

Register	Offset	R/W	Description	Reset Value
SYS_IRCTISTS	SYS_BA+0xF8	R/W	HIRC Trim Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					CLKERRIF	TFAILIF	FREQLOCK

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**CLKERRIF**	**Clock Error Interrupt Status** When the frequency of 32.768 kHz external low speed crystal oscillator (LXT) or 22.1184 MHz internal high speed RC oscillator (HIRC) is shift larger to unreasonable value, this bit will be set and to be an indicate that clock frequency is inaccuracy Once this bit is set to 1, the auto trim operation stopped and FREQSEL(SYS_IRCTCL[1:0]) will be cleared to 00 by hardware automatically if CESTOPEN(SYS_IRCTCTL[8]) is set to 1. If this bit is set and CLKEIEN(SYS_IRCTIEN[2]) is high, an interrupt will be triggered to notify the clock frequency is inaccuracy. Write 1 to clear this to 0. 0 = Clock frequency is accuracy. 1 = Clock frequency is inaccuracy.
[1]	**TFAILIF**	**Trim Failure Interrupt Status** This bit indicates that HIRC trim value update limitation count reached and the HIRC clock frequency still doesn't be locked. Once this bit is set, the auto trim operation stopped and FREQSEL(SYS_iRCTCTL[1:0]) will be cleared to 00 by hardware automatically. If this bit is set and TFAILIEN(SYS_IRCTIEN[1]) is high, an interrupt will be triggered to notify that HIRC trim value update limitation count was reached. Write 1 to clear this to 0. 0 = Trim value update limitation count does not reach. 1 = Trim value update limitation count reached and HIRC frequency still not locked.
[0]	**FREQLOCK**	**HIRC Frequency Lock Status** This bit indicates the HIRC frequency is locked. This is a status bit and doesn't trigger any interrupt. 0 = The internal high-speed oscillator frequency doesn't lock at 22.1184 MHz yet. 1 = The internal high-speed oscillator frequency locked at 22.1184 MHz.

Register Lock Control Register (SYS_REGLCTL)

Some of the system control registers need to be protected to avoid inadvertent write and disturb the chip operation. These system control registers are protected after the power-on reset till user to disable register protection. For user to program these protected registers, a register protection disable sequence needs to be followed by a special programming. The register protection disable sequence is writing the data “59h”, “16h” “88h” to the register SYS_REGLCTL address at 0x4000_0100 continuously. Any different data value, different sequence or any other write to other address during these three data writing will abort the whole sequence.

After the protection is disabled, user can check the protection disable bit at address 0x4000_0100 bit0, 1 is protection disable, and 0 is protection enable. Then user can update the target protected register value and then write any data to the address “0x4000_0100” to enable register protection.

This register is written to disable/enable register protection and read for the REGLCTL status.

Register	Offset	R/W	Description	Reset Value
SYS_REGLCTL	SYS_BA+0x100	R/W	Register Lock Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
REGLCTL[7:0]							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[7:0]	**REGLCTL**	**Register Lock Control Code (Write Only)** Some registers have write-protection function. Writing these registers have to disable the protected function by writing the sequence value “59h”, “16h”, “88h” to this field. After this sequence is completed, the REGLCTL bit will be set to 1 and write-protection registers can be normal write.
[0]	**REGLCTL**	**Register Lock Control 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. The Protected registers are: **SYS_IPRST0**: address 0x4000_0008 **SYS_BODCTL**: address 0x4000_0018 **SYS_PORCTL**: address 0x4000_0024 **SYS_VREFCTL:** address 0x4000_0028 **SYS_USBPHY:** address 0x4000_002C **CLK_PWRCTL**: address 0x4000_0200 (bit[6] is not protected for power-down wake-up

		interrupt clear) **SYS_SRAM_BISTCTL:** address 0x4000_00D0 **CLK_APBCLK0 [0]**: address 0x4000_0208 (bit[0] is watchdog clock enable) **CLK_CLKSEL0**: address 0x4000_0210 (for HCLK and CPU STCLK clock source select) **CLK_CLKSEL1 [1:0]**: address 0x4000_0214 (for watchdog clock source select) **CLK_CLKSEL1 [31:30]**: address 0x4000_0214 (for window watchdog clock source select) **CLK_CLKDSTS:** address 0x4000_0274 **NMIEN**: address 0x4000_0300 **FMC_ISPCTL**: address 0x4000_C000 (Flash ISP Control register) **FMC_ISPTRG**: address 0x4000_C010 (ISP Trigger Control register) **FMC_ISPSTS**: address 0x4000_C040 **WDT_CTL**: address 0x4004_0000 **FMC_FTCTL**: address 0x4000_5018 **SYS_AHBMCTL:** address 0x40000400 **CLK_PLLCTL:** address 0x40000240 **PWM_CTL0:** address 0x4005_8000 **PWM_CTL0:** address 0x4005_9000 **PWM_DTCTL0_1:** address 0x4005_8070 **PWM_DTCTL0_1:** address 0x4005_9070 **PWM_DTCTL2_3:** address 0x4005_8074 **PWM_DTCTL2_3:** address 0x4005_9074 **PWM_DTCTL4_5:** address 0x4005_8078 **PWM_DTCTL4_5:** address 0x4005_9078 **PWM_BRKCTL0_1:** address 0x4005_80C8 **PWM_BRKCTL0_1:** address 0x4005_90C8 **PWM_BRKCTL2_3:** address 0x4005_80CC **PWM_BRKCTL2_3:** address 0x4005_90CC **PWM_BRKCTL4_5:** address 0x4005_80D0 **PWM_BRKCTL4_5:** address 0x4005_90D0 **PWM_INTEN1:** address 0x4005_80E4 **PWM_INTEN1:** address 0x4005_90E4 **PWM_INTSTS1:** address 0x4005_80EC **PWM_INTSTS1:** address 0x4005_90EC

AHB Bus Matrix Priority Control Register (SYS_AHBMCTL)

Register	Offset	R/W	Description	Reset Value
SYS_AHBMCTL	SYS_BA+0x400	R/W	AHB Bus Matrix Priority Control Register	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**INTACTEN**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**INTACTEN**	**Highest AHB Bus Priority of Cortex M4 Core Enable Bit (Write Protect)** Enable Cortex®-M4 Core With Highest AHB Bus Priority In AHB Bus Matrix 0 = Run robin mode. 1 = Cortex®-M4 CPU with highest bus priority when interrupt occusr. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

6.2.9 System Timer (SysTick)

The Cortex®-M4 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 cycle, 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 will 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.

For more detailed information, please refer to the "*ARM® Cortex™-M4 Technical Reference Manual*" and "*ARM® v6-M Architecture Reference Manual*".

6.2.9.1 System Timer Control Register Map

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

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

6.2.9.2 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

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							COUNTFLAG
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					CLKSRC	TICKINT	ENABLE

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 Selection** 0 = Clock source is the (optional) external reference clock. 1 = Core clock used for SysTick.
[1]	**TICKINT**	**System Tick Interrupt Enabled** 0 = Counting down to 0 does not cause the SysTick exception to be pended. Software can use COUNTFLAG to determine if a count to zero 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 Enabled** 0 = Counter Disabled. 1 = Counter 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	0xXXXX_XXXX

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
RELOAD							
15	14	13	12	11	10	9	8
RELOAD							
7	6	5	4	3	2	1	0
RELOAD							

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	0xXXXX_XXXX

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
CURRENT							
15	14	13	12	11	10	9	8
CURRENT							
7	6	5	4	3	2	1	0
CURRENT							

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).

6.2.10 Nested Vectored Interrupt Controller (NVIC)

The NVIC and the processor core interface are closely coupled to enable low latency interrupt processing and efficient processing of late arriving interrupts. The NVIC maintains knowledge of the stacked, or nested, interrupts to enable tail-chaining of interrupts. You can only fully access the NVIC from privileged mode, but you can cause interrupts to enter a pending state in user mode if you enable the Configuration and Control Register. Any other user mode access causes a bus fault. You can access all NVIC registers using byte, halfword, and word accesses unless otherwise stated. NVIC registers are located within the SCS (System Control Space). All NVIC registers and system debug registers are little-endian regardless of the endianness state of the processor.

The NVIC supports:

- An implementation-defined number of interrupts, in the range 1-240 interrupts.
- A programmable priority level of 0-15 for each interrupt; a higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.
- Level and pulse detection of interrupt signals.
- Dynamic reprioritization of interrupts.
- Grouping of priority values into group priority and subpriority fields.
- Interrupt tail-chaining.
- An external Non Maskable Interrupt (NMI)
- WIC with Ultra-low Power Sleep mode support

The processor automatically stacks its state on exception entry and unstacks this state on exception exit, with no instruction overhead. This provides low latency exception handling.

6.2.10.1 Exception Model and System Interrupt Map

The Table 6-6 lists the exception model supported by M4TK Series. Software can set 16 levels of priority on some of these exceptions as well as on all interrupts. The highest user-configurable priority is denoted as “0x00” and the lowest priority is denoted as “0xF0” (The 4-LSB always 0). The default priority of all the user-configurable interrupts is “0x00”. Note that priority “0” is treated as the fourth priority on the system, after three system exceptions “Reset”, “NMI” and “Hard Fault”.

When any interrupts is accepted, the processor will automatically fetch the starting address of the interrupt service routine (ISR) from a vector table in memory. On system reset, the vector table is fixed at address 0x00000000. Privileged software can write to the VTOR to relocate the vector table start address to a different memory location, in the range 0x00000080 to 0x3FFFFF80,

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 exception handler entry as illustrated in previous section.

Exception Type	Vector Number	Vector Address	Priority
Reset	1	0x00000004	-3
NMI	2	0x00000008	-2
Hard Fault	3	0x0000000C	-1
Memory Manager Fault	4	0x00000010	Configurable

Bus Fault	5	0x00000014	Configurable
Usage Fault	6	0x00000018	Configurable
Reserved	7 ~ 10		Reserved
SVCall	11	0x0000002C	Configurable
Debug Monitor	12	0x00000030	Configurable
Reserved	13		Reserved
PendSV	14	0x00000038	Configurable
SysTick	15	0x0000003C	Configurable
Interrupt (IRQ0 ~ IRQ)	16 ~ 79	0x00000000 + (Vector Number)*4	Configurable

Table 6-6 Exception Model

Vector Number	Interrupt Number (Bit In Interrupt Registers)	Interrupt Name	Interrupt Description
0 ~ 15	-	-	System exceptions
16	0	BODOUT	Brown-Out low voltage detected interrupt
17	1	IRC_INT	IRC TRIM interrupt
18	2	PWRWU_INT	Clock controller interrupt for chip wake-up from power-down state
19	3	SRAM_PERR	SRAM parity check error interrupt
20	4	CLKFAIL	Clock fail detected interrupt
21	5	Reserved	Reserved
22	6	RTC_INT	Real time clock interrupt
23	7	TAMPER_INT	Backup register tamper interrupt
24	8	WDT_INT	Watchdog Timer interrupt
25	9	WWDT_INT	Window Watchdog Timer interrupt
26	10	EINT0	External interrupt from PA.0, PD.2 or PE.4 pins
27	11	EINT1	External interrupt from PB.0, PD.3 or PE.5 pins
28	12	EINT2	External interrupt from PC.0 pin
29	13	EINT3	External interrupt from PD.0 pin
30	14	EINT4	External interrupt from PE.0 pin
31	15	EINT5	External interrupt from PF.0 pin
32	16	GPA_INT	External interrupt from PA[15:0] pin
33	17	GPB_INT	External interrupt from PB[15:0] pin
34	18	GPC_INT	External interrupt from PC[15:0] pin
35	19	GPD_INT	External interrupt from PD[15:0] pin

36	20	GPE_INT	External interrupt from PE[14:0] pin
37	21	GPF_INT	External interrupt from PF[7:0] pin
38	22	SPI0_INT	SPI0 interrupt
39	23	SPI1_INT	SPI1 interrupt
40	24	BRAKE0_INT	PWM0 brake interrupt
41	25	PWM0_P0_INT	PWM0 pair 0 interrupt
42	26	PWM0_P1_INT	PWM0 pair 1 interrupt
43	27	PWM0_P2_INT	PWM0 pair 2 interrupt
44	28	BRAKE1_INT	PWM1 brake interrupt
45	29	PWM1_P0_INT	PWM1 pair 0 interrupt
46	30	PWM1_P1_INT	PWM1 pair 1 interrupt
47	31	PWM1_P2_INT	PWM1 pair 2 interrupt
48	32	TMR0_INT	Timer 0 interrupt
49	33	TMR1_INT	Timer 1 interrupt
50	34	TMR2_INT	Timer 2 interrupt
51	35	TMR3_INT	Timer 3 interrupt
52	36	UART0_INT	UART0 interrupt
53	37	UART1_INT	UART1 interrupt
54	38	I2C0_INT	I2C0 interrupt
55	39	I2C1_INT	I2C1 interrupt
56	40	PDMA_INT	PDMA interrupt
57	41	DAC_INT	DAC interrupt
58	42	EADC0_INT	EADC interrupt source 0
59	43	EADC1_INT	EADC interrupt source 1
60	44	ACMP01_INT	ACMP0 and ACMP1 interrupt
61	45	Reserved	Reserved
62	46	EADC2_INT	EADC interrupt source 2
63	47	EADC3_INT	EADC interrupt source 3
64	48	UART2_INT	UART2 interrupt
65	49	UART3_INT	UART3 interrupt
66	50	Reserved	Reserved
67	51	SPI2_INT	SPI2 interrupt
68	52	Reserved	Reserved
69	53	USBD_INT	USB device interrupt
70	54	USBH_INT	USB host interrupt

71	55	USBOTG_INT	USB OTG interrupt
72	56	CAN0_INT	CAN0 interrupt
73	57	Reserved	Reserved
74	58	SC0_INT	Smart card host 0 interrupt
75	59	Reserved	Reserved
76	60	Reserved	Reserved
77	61	Reserved	Reserved
78	62	Reserved	Reserved
79	63	TOUCHKEY_INT	Touch key interrupt

Table 6-7 Interrupt Number Table

6.2.10.2 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 activate. 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.

6.2.10.3 NVIC Control Registers

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

Register	Offset	R/W	Description	Reset Value
NVIC Base Address: **NVIC_BA = 0xE000_E100**				
NVIC_ISER1	NVIC_BA+0x000	R/W	IRQ0 ~ IRQ63 Set-Enable Control Register	0x0000_0000
NVIC_ISER2	NVIC_BA+0x004	R/W	IRQ0 ~ IRQ63 Set-Enable Control Register	0x0000_0000
NVIC_ICER1	NVIC_BA+0x080	R/W	IRQ0 ~ IRQ63 Clear-Enable Control Register	0x0000_0000
NVIC_ICER2	NVIC_BA+0x084	R/W	IRQ0 ~ IRQ63 Clear-Enable Control Register	0x0000_0000
NVIC_ISPR1	NVIC_BA+0x100	R/W	IRQ0 ~ IRQ63 Set-Pending Control Register	0x0000_0000
NVIC_ISPR2	NVIC_BA+0x104	R/W	IRQ0 ~ IRQ63 Set-Pending Control Register	0x0000_0000
NVIC_ICPR1	NVIC_BA+0x180	R/W	IRQ0 ~ IRQ63 Clear-Pending Control Register	0x0000_0000
NVIC_ICPR2	NVIC_BA+0x184	R/W	IRQ0 ~ IRQ63 Clear-Pending Control Register	0x0000_0000
NVIC_IABR1	NVIC_BA+0x200	R/W	IRQ0 ~ IRQ63 Active Bit Register	0x0000_0000
NVIC_IABR2	NVIC_BA+0x204	R/W	IRQ0 ~ IRQ63 Active Bit Register	0x0000_0000
NVIC_IPR1	NVIC_BA+0x300	R/W	IRQ0 ~ IRQ63 Priority Control Register	0x0000_0000
NVIC_IPR2	NVIC_BA+0x33C	R/W	IRQ0 ~ IRQ63 Priority Control Register	0x0000_0000
STIR	NVIC_BA+0xE00	R/W	Software Trigger Interrupt Registers	0x0000_0000

IRQ0 ~ IRQ63 Set-Enable Control Register (NVIC_ISER1)

Register	Offset	R/W	Description	Reset Value
NVIC_ISER1	NVIC_BA+0x000	R/W	IRQ0 ~ IRQ63 Set-Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SETENA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SETENA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SETENA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SETENA							

Bits	Description	
[31:0]	**SETENA**	**Interrupt Set Enable Bit** The NVIC_ISER0-NVIC_ISER3 registers enable interrupts, and show which interrupts are enabled Write Operation: 0 = No effect. 1 = Interrupt Enabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ0 ~ IRQ63 Set-Enable Control Register (NVIC_ISER2)

Register	Offset	R/W	Description	Reset Value
NVIC_ISER2	NVIC_BA+0x004	R/W	IRQ0 ~ IRQ63 Set-Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SETENA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SETENA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SETENA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SETENA							

Bits	Description	
[31:0]	**SETENA**	**Interrupt Set Enable Bit** The NVIC_ISER0-NVIC_ISER3 registers enable interrupts, and show which interrupts are enabled Write Operation: 0 = No effect. 1 = Interrupt Enabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ0 ~ IRQ63 Clear-Enable Control Register (NVIC_ICER1)

Register	Offset	R/W	Description	Reset Value
NVIC_ICER1	NVIC_BA+0x080	R/W	IRQ0 ~ IRQ63 Clear-Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CALENA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CALENA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CALENA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CALENA							

Bits	Description	
[31:0]	**CALENA**	**Interrupt Clear Enable Bit** The NVIC_ICER0-NVIC_ICER3 registers disable interrupts, and show which interrupts are enabled. Write Operation: 0 = No effect. 1 = Interrupt Disabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ0 ~ IRQ63 Clear-Enable Control Register (NVIC_ICER2)

Register	Offset	R/W	Description	Reset Value
NVIC_ICER2	NVIC_BA+0x084	R/W	IRQ0 ~ IRQ63 Clear-Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CALENA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CALENA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CALENA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CALENA							

Bits	Description	
[31:0]	**CALENA**	**Interrupt Clear Enable Bit** The NVIC_ICER0-NVIC_ICER3 registers disable interrupts, and show which interrupts are enabled. Write Operation: 0 = No effect. 1 = Interrupt Disabled. Read Operation: 0 = Interrupt Disabled. 1 = Interrupt Enabled.

IRQ0 ~ IRQ63 Set-Pending Control Register (NVIC_ISPR1)

Register	Offset	R/W	Description	Reset Value
NVIC_ISPR1	NVIC_BA+0x100	R/W	IRQ0 ~ IRQ63 Set-Pending Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SETPEND							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SETPEND							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SETPEND							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SETPEND							

Bits	Description	
[31:0]	**SETPEND**	**Interrupt Set-pending** The NVIC_ISPR0-NVIC_ISPR3 registers force interrupts into the pending state, and show which interrupts are pending Write Operation: 0 = No effect. 1 = Changes interrupt state to pending. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ0 ~ IRQ63 Set-Pending Control Register (NVIC_ISPR2)

Register	Offset	R/W	Description	Reset Value
NVIC_ISPR2	NVIC_BA+0x104	R/W	IRQ0 ~ IRQ63 Set-Pending Control Register	0x0000_0000

31	30	29	28	27	26	25	24
SETPEND							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SETPEND							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SETPEND							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SETPEND							

Bits	Description	
[31:0]	**SETPEND**	**Interrupt Set-pending** The NVIC_ISPR0-NVIC_ISPR3 registers force interrupts into the pending state, and show which interrupts are pending Write Operation: 0 = No effect. 1 = Changes interrupt state to pending. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ0 ~ IRQ63 Clear-Pending Control Register (NVIC_ICPR1)

Register	Offset	R/W	Description	Reset Value
NVIC_ICPR1	NVIC_BA+0x180	R/W	IRQ0 ~ IRQ63 Clear-Pending Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CALPEND							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CALPEND							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CALPEND							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CALPEND							

Bits	Description	
[31:0]	**CALPEND**	**Interrupt Clear-pending** The NVIC_ICPR0-NVIC_ICPR2 registers remove the pending state from interrupts, and show which interrupts are pending Write Operation: 0 = No effect. 1 = Removes pending state an interrupt. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ0 ~ IRQ63 Clear-Pending Control Register (NVIC_ICPR2)

Register	Offset	R/W	Description	Reset Value
NVIC_ICPR2	NVIC_BA+0x184	R/W	IRQ0 ~ IRQ63 Clear-Pending Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CALPEND							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CALPEND							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CALPEND							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CALPEND							

Bits	Description	
[31:0]	**CALPEND**	**Interrupt Clear-pending** The NVIC_ICPR0-NVIC_ICPR2 registers remove the pending state from interrupts, and show which interrupts are pending Write Operation: 0 = No effect. 1 = Removes pending state an interrupt. Read Operation: 0 = Interrupt is not pending. 1 = Interrupt is pending.

IRQ0 ~ IRQ63 Active Bit Register (NVIC_IABR1)

Register	Offset	R/W	Description	Reset Value
NVIC_IABR1	NVIC_BA+0x200	R/W	IRQ0 ~ IRQ63 Active Bit Register	0x0000_0000

31	30	29	28	27	26	25	24
ACTIVE							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ACTIVE							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ACTIVE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ACTIVE							

Bits	Description	
[31:0]	**ACTIVE**	**Interrupt Active Flags** The NVIC_IABR0-NVIC_IABR3 registers indicate which interrupts are active. 0 = interrupt not active. 1 = interrupt active.

IRQ0 ~ IRQ63 Active Bit Register (NVIC_IABR2)

Register	Offset	R/W	Description	Reset Value
NVIC_IABR2	NVIC_BA+0x204	R/W	IRQ0 ~ IRQ63 Active Bit Register	0x0000_0000

31	30	29	28	27	26	25	24
ACTIVE							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ACTIVE							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ACTIVE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ACTIVE							

Bits	Description	
[31:0]	**ACTIVE**	**Interrupt Active Flags** The NVIC_IABR0-NVIC_IABR3 registers indicate which interrupts are active. 0 = interrupt not active. 1 = interrupt active.

IRQ0 ~ IRQ63 Interrupt Priority Register (NVIC_IPR1)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR1	NVIC_BA+0x300	R/W	IRQ0 ~ IRQ63 Priority Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PRI_4n_3				Reserved			
23	22	21	20	19	18	17	16
PRI_4n_2				Reserved			
15	14	13	12	11	10	9	8
PRI_4n_1				Reserved			
7	6	5	4	3	2	1	0
PRI_4n_0				Reserved			

Bits	Description	
[31:28]	**PRI_4n_3**	**Priority of IRQ_4n+3** "0" denotes the highest priority and "15" denotes the lowest priority
[27:24]	**Reserved**	Reserved.
[23:20]	**PRI_4n_2**	**Priority of IRQ_4n+2** "0" denotes the highest priority and "15" denotes the lowest priority
[19:16]	**Reserved**	Reserved.
[15:12]	**PRI_4n_1**	**Priority of IRQ_4n+1** "0" denotes the highest priority and "15" denotes the lowest priority
[11:8]	**Reserved**	Reserved.
[7:4]	**PRI_4n_0**	**Priority of IRQ_4n+0** "0" denotes the highest priority and "15" denotes the lowest priority
[3:0]	**Reserved**	Reserved.

IRQ0 ~ IRQ63 Interrupt Priority Register (NVIC_IPR2)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR2	NVIC_BA+0x33C	R/W	IRQ0 ~ IRQ63 Priority Control Register	0x0000_0000

31	30	29	28	27	26	25	24
PRI_4n_3				Reserved			
23	22	21	20	19	18	17	16
PRI_4n_2				Reserved			
15	14	13	12	11	10	9	8
PRI_4n_1				Reserved			
7	6	5	4	3	2	1	0
PRI_4n_0				Reserved			

Bits	Description	
[31:28]	**PRI_4n_3**	**Priority of IRQ_4n+3** "0" denotes the highest priority and "15" denotes the lowest priority
[27:24]	**Reserved**	Reserved.
[23:20]	**PRI_4n_2**	**Priority of IRQ_4n+2** "0" denotes the highest priority and "15" denotes the lowest priority
[19:16]	**Reserved**	Reserved.
[15:12]	**PRI_4n_1**	**Priority of IRQ_4n+1** "0" denotes the highest priority and "15" denotes the lowest priority
[11:8]	**Reserved**	Reserved.
[7:4]	**PRI_4n_0**	**Priority of IRQ_4n+0** "0" denotes the highest priority and "15" denotes the lowest priority
[3:0]	**Reserved**	Reserved.

Software Trigger Interrupt Register (STIR)

Register	Offset	R/W	Description	Reset Value
STIR	NVIC_BA+0xE00	R/W	Software Trigger Interrupt Registers	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							INTID
7	6	5	4	3	2	1	0
INTID							

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:0]	**INTID**	**Interrupt ID** Write to the STIR To Generate An Interrupt from Software When the USERSETMPEND bit in the SCR is set to 1, unprivileged software can access the STIR Interrupt ID of the interrupt to trigger, in the range 0-63. For example, a value of 0x03 specifies interrupt IRQ3.

6.2.10.4 NMI Control Registers

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

Register	Offset	R/W	Description	Reset Value
NMI Base Address: **NMI_BA = 0x4000_0300**				
NMIEN	NMI_BA+0x00	R/W	NMI Source Interrupt Enable Register	0x0000_0000
NMISTS	NMI_BA+0x04	R	NMI source interrupt Status Register	0x0000_0000

NMI Source Interrupt Enable Register (NMIEN)

Register	Offset	R/W	Description	Reset Value
NMIEN	NMI_BA+0x00	R/W	NMI Source Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
UART1_INT	UART0_INT	EINT5	EINT4	EINT3	EINT2	EINT1	EINT0
7	6	5	4	3	2	1	0
TAMPER_INT	RTC_INT	Reserved	CLKFAIL	SRAM_PERR	PWRWU_INT	IRC_INT	BODOUT

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**UART1_INT**	**UART1 NMI Source Enable (Write Protect)** 0 = UART1 NMI source Disabled. 1 = UART1 NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[14]	**UART0_INT**	**UART0 NMI Source Enable (Write Protect)** 0 = UART0 NMI source Disabled. 1 = UART0 NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[13]	**EINT5**	**External Interrupt From PF.0 Pin NMI Source Enable (Write Protect)** 0 = External interrupt from PF.0 pin NMI source Disabled. 1 = External interrupt from PF.0 pin NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[12]	**EINT4**	**External Interrupt From PE.0 Pin NMI Source Enable (Write Protect)** 0 = External interrupt from PE.0 pin NMI source Disabled. 1 = External interrupt from PE.0 pin NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[11]	**EINT3**	**External Interrupt From PD.0 Pin NMI Source Enable (Write Protect)** 0 = External interrupt from PD.0 pin NMI source Disabled. 1 = External interrupt from PD.0 pin NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[10]	**EINT2**	**External Interrupt From PC.0 Pin NMI Source Enable (Write Protect)** 0 = External interrupt from PC.0 pin NMI source Disabled. 1 = External interrupt from PC.0 pin NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[9]	**EINT1**	**External Interrupt From PB.0, PD.3 or PE.5 Pin NMI Source Enable (Write Protect)**

		0 = External interrupt from PB.0, PD.3 or PE.5 pin NMI source Disabled. 1 = External interrupt from PB.0, PD.3 or PE.5 pin NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[8]	**EINT0**	**External Interrupt From PA.0, PD.2 or PE.4 Pin NMI Source Enable (Write Protect)** 0 = External interrupt from PA.0, PD.2 or PE.4 pin NMI source Disabled. 1 = External interrupt from PA.0, PD.2 or PE.4 pin NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[7]	**TAMPER_INT**	**TAMPER_INT NMI Source Enable (Write Protect)** 0 = Backup register tamper detected interrupt.NMI source Disabled. 1 = Backup register tamper detected interrupt.NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[6]	**RTC_INT**	**RTC NMI Source Enable (Write Protect)** 0 = RTC NMI source Disabled. 1 = RTC NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[5]	**Reserved**	Reserved.
[4]	**CLKFAIL**	**Clock Fail Detected NMI Source Enable (Write Protect)** 0 = Clock fail detected interrupt NMI source Disabled. 1 = Clock fail detected interrupt NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[3]	**SRAM_PERR**	**SRAM ParityCheck Error NMI Source Enable (Write Protect)** 0 = SRAM parity check error NMI source Disabled. 1 = SRAM parity check error NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2]	**PWRWU_INT**	**Power-down Mode Wake-up NMI Source Enable (Write Protect)** 0 = Power-down mode wake-up NMI source Disabled. 1 = Power-down mode wake-up NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[1]	**IRC_INT**	**IRC TRIM NMI Source Enable (Write Protect)** 0 = IRC TRIM NMI source Disabled. 1 = IRC TRIM NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[0]	**BODOUT**	**BOD NMI Source Enable (Write Protect)** 0 = BOD NMI source Disabled. 1 = BOD NMI source Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

NMI Source Interrupt Status Register (NMISTS)

Register	Offset	R/W	Description	Reset Value
NMISTS	NMI_BA+0x04	R	NMI source interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
UART1_INT	UART0_INT	EINT5	EINT4	EINT3	EINT2	EINT1	EINT0
7	6	5	4	3	2	1	0
TAMPER_INT	RTC_INT	Reserved	CLKFAIL	SRAM_PERR	PWRWU_INT	IRC_INT	BODOUT

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**UART1_INT**	**UART1 Interrupt Flag (Read Only)** 0 = UART1 interrupt is deasserted. 1 = UART1 interrupt is asserted.
[14]	**UART0_INT**	**UART0 Interrupt Flag (Read Only)** 0 = UART1 interrupt is deasserted. 1 = UART1 interrupt is asserted.
[13]	**EINT5**	**External Interrupt From PF.0 Pin Interrupt Flag (Read Only)** 0 = External Interrupt from PF.0 interrupt is deasserted. 1 = External Interrupt from PF.0 interrupt is asserted.
[12]	**EINT4**	**External Interrupt From PE.0 Pin Interrupt Flag (Read Only)** 0 = External Interrupt from PE.0 interrupt is deasserted. 1 = External Interrupt from PE.0 interrupt is asserted.
[11]	**EINT3**	**External Interrupt From PD.0 Pin Interrupt Flag (Read Only)** 0 = External Interrupt from PD.0 interrupt is deasserted. 1 = External Interrupt from PD.0 interrupt is asserted.
[10]	**EINT2**	**External Interrupt From PC.0 Pin Interrupt Flag (Read Only)** 0 = External Interrupt from PC.0 interrupt is deasserted. 1 = External Interrupt from PC.0 interrupt is asserted.
[9]	**EINT1**	**External Interrupt From PB.0, PD.3 or PE.5 Pin Interrupt Flag (Read Only)** 0 = External Interrupt from PB.0, PD.3 or PE.5 interrupt is deasserted. 1 = External Interrupt from PB.0, PD.3 or PE.5 interrupt is asserted.
[8]	**EINT0**	**External Interrupt From PA.0, PD.2 or PE.4 Pin Interrupt Flag (Read Only)** 0 = External Interrupt from PA.0, PD.2 or PE.4 interrupt is deasserted. 1 = External Interrupt from PA.0, PD.2 or PE.4 interrupt is asserted.
[7]	**TAMPER_INT**	**TAMPER_INT Interrupt Flag (Read Only)**

		0 = Backup register tamper detected interrupt is deasserted. 1 = Backup register tamper detected interrupt is asserted.
[6]	**RTC_INT**	**RTC Interrupt Flag (Read Only)** 0 = RTC interrupt is deasserted. 1 = RTC interrupt is asserted.
[5]	**Reserved**	Reserved.
[4]	**CLKFAIL**	**Clock Fail Detected Interrupt Flag (Read Only)** 0 = Clock fail detected interrupt is deasserted. 1 = Clock fail detected interrupt is asserted.
[3]	**SRAM_PERR**	**SRAM ParityCheck Error Interrupt Flag (Read Only)** 0 = SRAM parity check error interrupt is deasserted. 1 = SRAM parity check error interrupt is asserted.
[2]	**PWRWU_INT**	**Power-down Mode Wake-up Interrupt Flag (Read Only)** 0 = Power-down mode wake-up interrupt is deasserted. 1 = Power-down mode wake-up interrupt is asserted.
[1]	**IRC_INT**	**IRC TRIM Interrupt Flag (Read Only)** 0 = HIRC TRIM interrupt is deasserted. 1 = HIRC TRIM interrupt is asserted.
[0]	**BODOUT**	**BOD Interrupt Flag (Read Only)** 0 = BOD interrupt is deasserted. 1 = BOD interrupt is asserted.

6.2.11 System Control Register

The Cortex®-M4 status and operation mode control are managed by System Control Registers. Including CPUID, Cortex®-M4 interrupt priority and Cortex®-M0 power management can be controlled through these system control registers.

For more detailed information, please refer to the "*ARM® Cortex™-M4 Technical Reference Manual*" and "*ARM® v6-M Architecture Reference Manual*".

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

Register	Offset	R/W	Description	Reset Value
SCR Base Address: **SCS_BA = 0xE000_E000**				
ICSR	SCS_BA+0xD04	R/W	Interrupt Control and State Register	0x0000_0000
AIRCR	SCS_BA+0xD0C	R/W	Application Interrupt and Reset Control Register	0xFA05_0000
SCR	SCS_BA+0xD10	R/W	System Control Register	0x0000_0000
SHPR1	SCS_BA+0xD18	R/W	System Handler Priority Register 1	0x0000_0000
SHPR2	SCS_BA+0xD1C	R/W	System Handler Priority Register 2	0x0000_0000
SHPR3	SCS_BA+0xD20	R/W	System Handler Priority Register 3	0x0000_0000

Interrupt Control State Register (ICSR)

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

31	30	29	28	27	26	25	24
NMIPENDSET	Reserved		PENDSVSET	PENDSVRTC_CAL	PENDSTSET	PENDSTRTC_CAL	Reserved
23	22	21	20	19	18	17	16
ISRPREEMPT	ISRPENDING	Reserved				VECTPENDING	
15	14	13	12	11	10	9	8
VECTPENDING				Reserved			
7	6	5	4	3	2	1	0
Reserved		VECTACTIVE					

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]	**PENDSVRTC_CAL**	**PendSV Clear-pending Bit** Write Operation: 0 = No effect. 1 = Removes the pending state from the PendSV exception. **Note:** This is a write only bit. To clear the PENDSV bit, you must “write 0 to PENDSVSET and write 1 to PENDSVRTC_CAL” 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]	**PENDSTRTC_CAL**	**SysTick Exception Clear-pending Bit** Write Operation: 0 = No effect. 1 = Removes the pending state from the SysTick exception. **Note:** This is a write only bit. To clear the PENDST bit, you must "write 0 to PENDSTSET and write 1 to PENDSTRTC_CAL" at the same time.
[24]	**Reserved**	Reserved.
[23]	**ISRPREEMPT**	**Interrupt Preempt Bit (Read Only)** If set, a pending exception will be serviced on exit from the debug halt state.
[22]	**ISRPENDING**	**Interrupt Pending Flag, Excluding NMI and Faults (Read Only)** 0 = Interrupt not pending. 1 = Interrupt pending.
[21:18]	**Reserved**	Reserved.
[17:12]	**VECTPENDING**	**Number of the Highest Pended Exception** Indicate the Exception Number of the Highest Priority Pending Enabled Exception 0 = no pending exceptions. Nonzero = the exception number of the highest priority pending enabled exception. The value indicated by this field includes the effect of the BASEPRI and FAULTMASK registers, but not any effect of the PRIMASK register.
[11]	**RETTOBASE**	**Preempted Active Exceptions Indicator** Indicate whether There are Preempted Active Exceptions 0 = there are preempted active exceptions to execute. 1 = there are no active exceptions, or the currently-executing exception is the only active exception.
[10:6]	**Reserved**	Reserved.
[5:0]	**VECTACTIVE**	**Number of the Current Active Exception** 0 = Thread mode. Non-zero = The exception number of the currently active exception.

Application Interrupt and Reset Control Register (AIRCR)

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

31	30	29	28	27	26	25	24
VECTORKEY							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
VECTORKEY							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ENDIANNESS	**Reserved**				**PRIGROUP**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**SYSRESETREQ**	**VECTCLRACTIVE**	**VECTRESET**

Bits	Description	
[31:16]	**VECTORKEY**	**Register Access Key** When writing this register, this field should be 0x05FA, otherwise the write action will be unpredictable. The VECTORKEY filed is used to prevent accidental write to this register from resetting the system or clearing of the exception status.
[15]	**ENDIANNESS**	**Data Endianness** 0 = Little-endian. 1 = Big-endian.
[14:11]	**Reserved**	Reserved.
[10:8]	**PRIGROUP**	**Interrupt Priority Grouping** This field determines the Split Of Group priority from subpriority,
[7:3]	**Reserved**	Reserved.
[2]	**SYSRESETREQ**	**System Reset Request** Writing This Bit to 1 Will Cause A Reset Signal To Be Asserted To The Chip And Indicate A Reset Is Requested This bit is write only and self-cleared as part of the reset sequence.
[1]	**VECTCLRACTIVE**	**Exception Active Status Clear Bit** Setting This Bit To 1 Will Clears All Active State Information For Fixed And Configurable Exceptions This bit is write only and can only be written when the core is halted. **Note:** It is the debugger's responsibility to re-initialize the stack.
[0]	**Reserved**	Reserved.

PRIGROUP	Binary Point	Group Priority Bits	Subpriority Bits	Number Of Group Priorities	Subpriorities
0b000	bxxxxxxx.y	[7:1]	[0]	128	2
0b001	bxxxxxx.yy	[7:2]	[1:0]	64	4
0b010	bxxxxx.yyy	[7:3]	[2:0]	32	8
0b011	bxxxx.yyyy	[7:4]	[3;0]	16	16
0b100	bxxx.yyyyy	[7:5]	[4:0]	8	32
0b101	bxx.yyyyyy	[7:6]	[5:0]	4	64
0b110	bx.yyyyyyy	[7]	[6:0]	2	128
0b111	b.yyyyyyyy	None	[7:0]	1	256

Table 6-8 Priority Grouping

System Control Register (SCR)

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

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved			SEVONPEND	Reserved	SLEEPDEEP	SLEEPONEXIT	Reserved

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**SEVONPEND**	**Send Event on Pending** 0 = Only enabled interrupts or events can wake up the processor, while 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 the 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** Control Whether the Processor Uses Sleep Or Deep Sleep as its Low Power Mode. 0 = Sleep. 1 = Deep sleep.
[1]	**SLEEPONEXIT**	**Sleep-on-exit Enable Control** This bit indicate Sleep-On-Exit when Returning from Handler Mode to Thread Mode. 0 = Do not sleep when returning to Thread mode. 1 = Enters sleep, or deep sleep, on return from an 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 1 (SHPR1)

Register	Offset	R/W	Description	Reset Value
SHPR1	SCS_BA+0xD18	R/W	System Handler Priority Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PRI_6							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PRI_5							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRI_4							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:16]	**PRI_6**	Priority of system handler 6, UsageFault
[15:8]	**PRI_5**	Priority of system handler 5, BusFault
[7:0]	**PRI_4**	Priority of system handler 4, MemManage

System Handler Priority Register 2 (SHPR2)

Register	Offset	R/W	Description	Reset Value
SHPR2	SCS_BA+0xD1C	R/W	System Handler Priority Register 2	0x0000_0000

31	30	29	28	27	26	25	24
PRI_11		**Reserved**					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:30]	**PRI_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 (SHPR3)

Register	Offset	R/W	Description	Reset Value
SHPR3	SCS_BA+0xD20	R/W	System Handler Priority Register 3	0x0000_0000

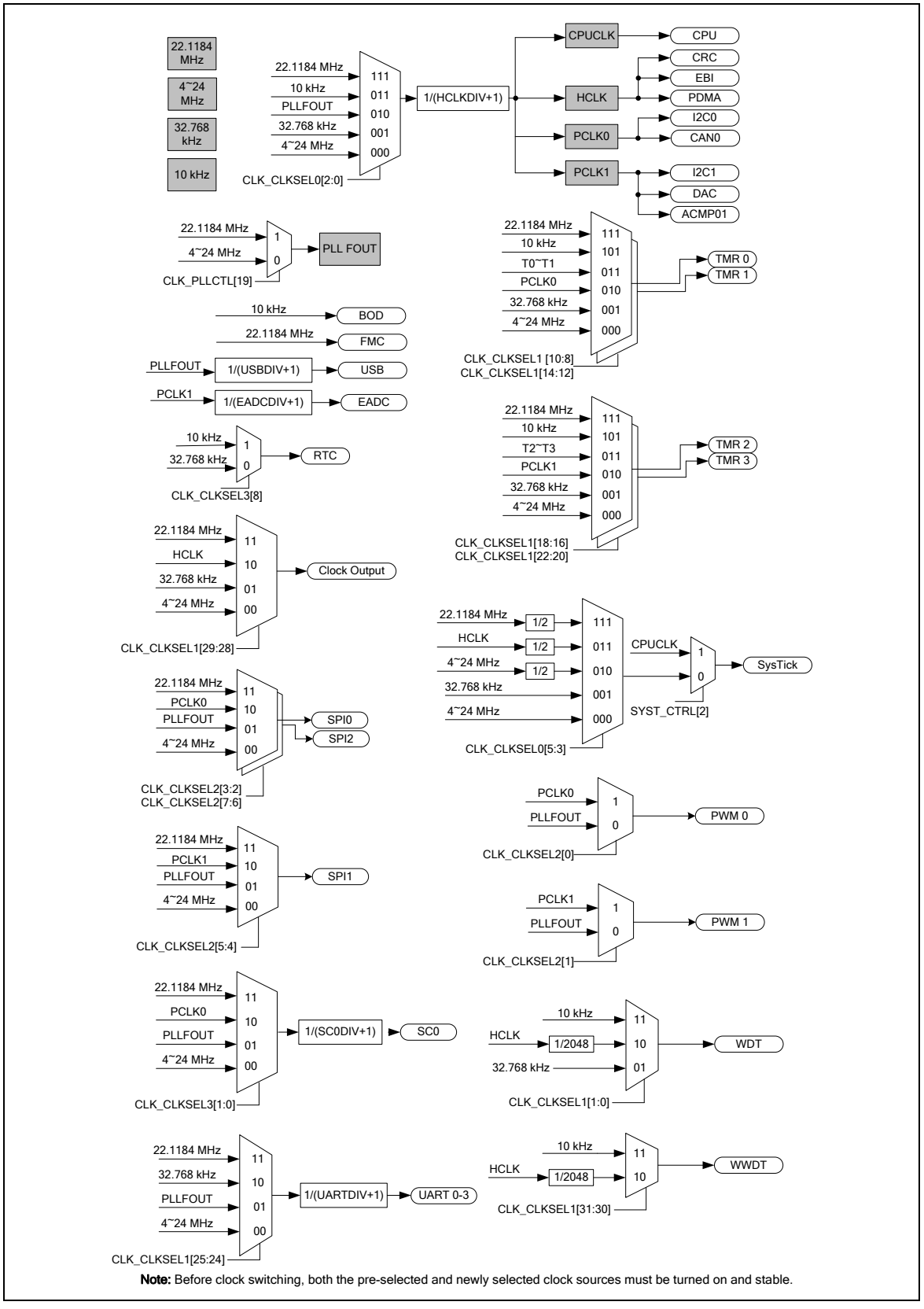
31	30	29	28	27	26	25	24
PRI_15		**Reserved**					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PRI_14		**Reserved**					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:30]	**PRI_15**	**Priority of System Handler 15 – SysTick** "0" denotes the highest priority and "3" denotes the lowest priority.
[29:24]	**Reserved**	Reserved.
[23:22]	**PRI_14**	**Priority of System Handler 14 – PendSV** "0" denotes the highest priority and "3" denotes the lowest priority.
[21:0]	**Reserved**	Reserved.

6.3 Clock Controller

6.3.1 Overview

The clock controller generates clocks for the whole chip, including system clocks and all peripheral clocks. The clock controller also implements the power control function with the individually clock ON/OFF control, clock source selection and a clock divider. The chip will not enter Power-down mode until CPU sets the Power-down enable bit PDEN(CLK_PWRCTL[7]) and Cortex®-M4 core executes the WFI instruction. After that, chip enters Power-down mode and wait for wake-up interrupt source triggered to leave Power-down mode. In Power-down mode, the clock controller turns off the 4~20 MHz external high speed crystal (HXT) and 22.1184 MHz internal high speed RC oscillator (HIRC) to reduce the overall system power consumption. The Figure 6.3-1 shows the clock generator and the overview of the clock source control.



Note: Before clock switching, both the pre-selected and newly selected clock sources must be turned on and stable.

Figure 6.3-1 Clock Generator Global View Diagram

6.3.2 Clock Generator

The clock generator consists of 5 clock sources, which are listed below:

- 32.768 kHz external low speed crystal oscillator (LXT)
- 4~20 MHz external high speed crystal oscillator (HXT)
- Programmable PLL output clock frequency (PLLFOUT), PLL source can be selected from external 4~20 MHz external high speed crystal (HXT) or 22.1184 MHz internal high speed oscillator (HIRC)
- 22.1184 MHz internal high speed RC oscillator (HIRC)
- 10 kHz internal low speed RC oscillator (LIRC)

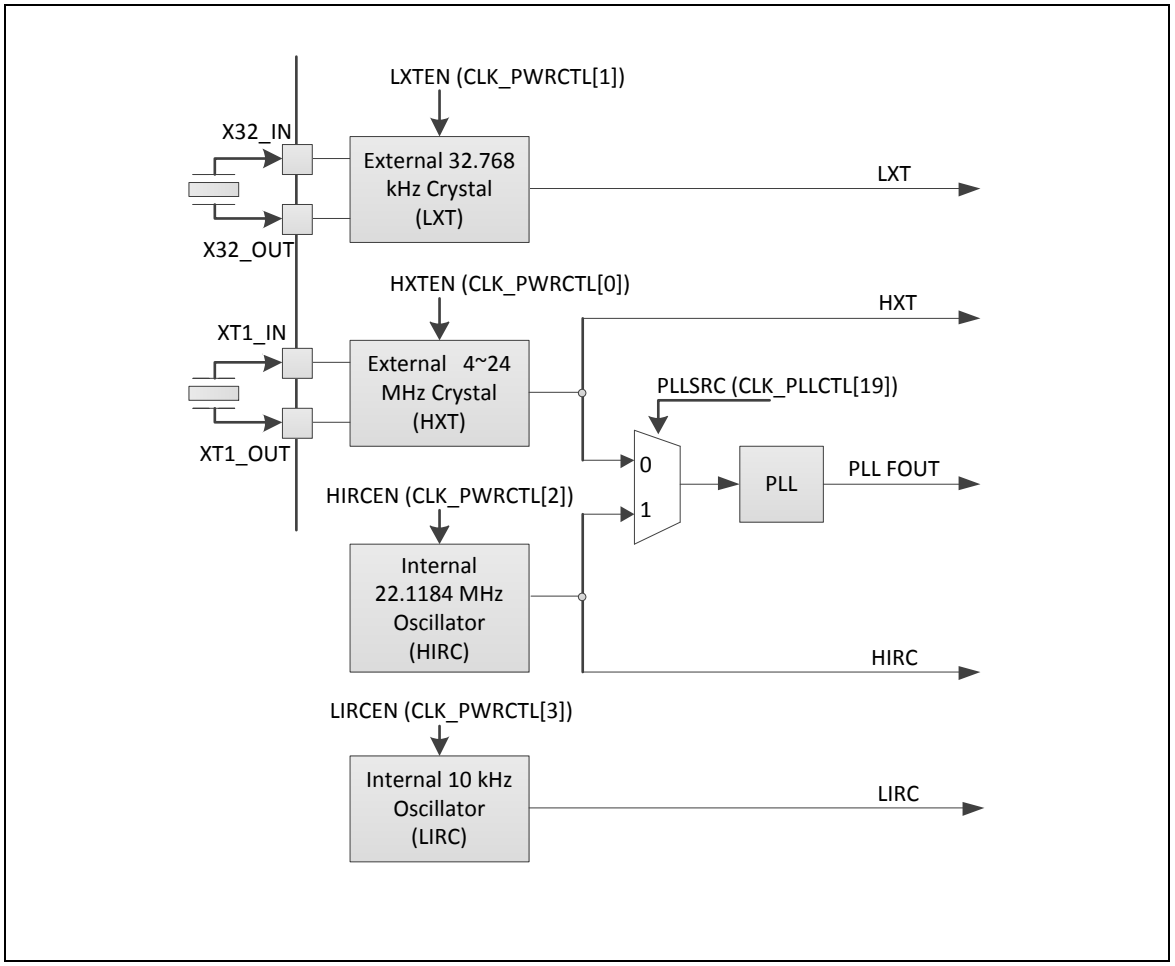
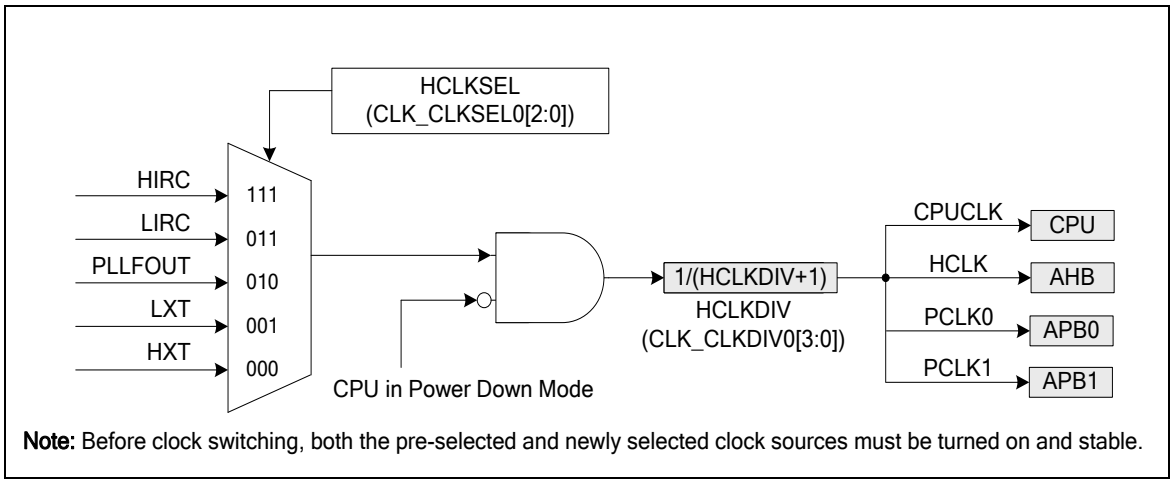


Figure 6.3-2 Clock Generator Block Diagram

6.3.3 System Clock and SysTick Clock

The system clock has 5 clock sources, which were generated from clock generator block. The clock source switch depends on the register HCLKSEL (CLK_CLKSEL0[2:0]). The block diagram is shown in the Figure 6.3-3.



Note: Before clock switching, both the pre-selected and newly selected clock sources must be turned on and stable.

Figure 6.3-3 System Clock Block Diagram

There are two clock fail detectors to observe HXT and LXT clock source and they have individual enable and interrupt control. When HXT detector is enabled, the HIRC clock is enabled automatically. When LXT detector is enabled, the LIRC clock is enabled automatically.

When HXT clock detector is enabled, the system clock will auto switch to HIRC if HXT clock stop being detected on the following condition: system clock source comes from HXT or system clock source comes from PLL with HXT as the input of PLL. If HXT clock stop condition is detected, the HXTFIF (CLK_CLKDSTS[0]) is set to 1 and chip will enter interrupt if HXTFIE (CLK_CLKDCTL[5]) is set to 1. User can trying to recover HXT by disable HXT and enable HXT again to check if the clock stable bit is set to 1 or not. If HXT clock stable bit is set to 1, it means HXT is recover to oscillate after re-enable action and user can switch system clock to HXT again.

The HXT clock stop detect and system clock switch to HIRC procedure is shown in the Figure 6.3-4.

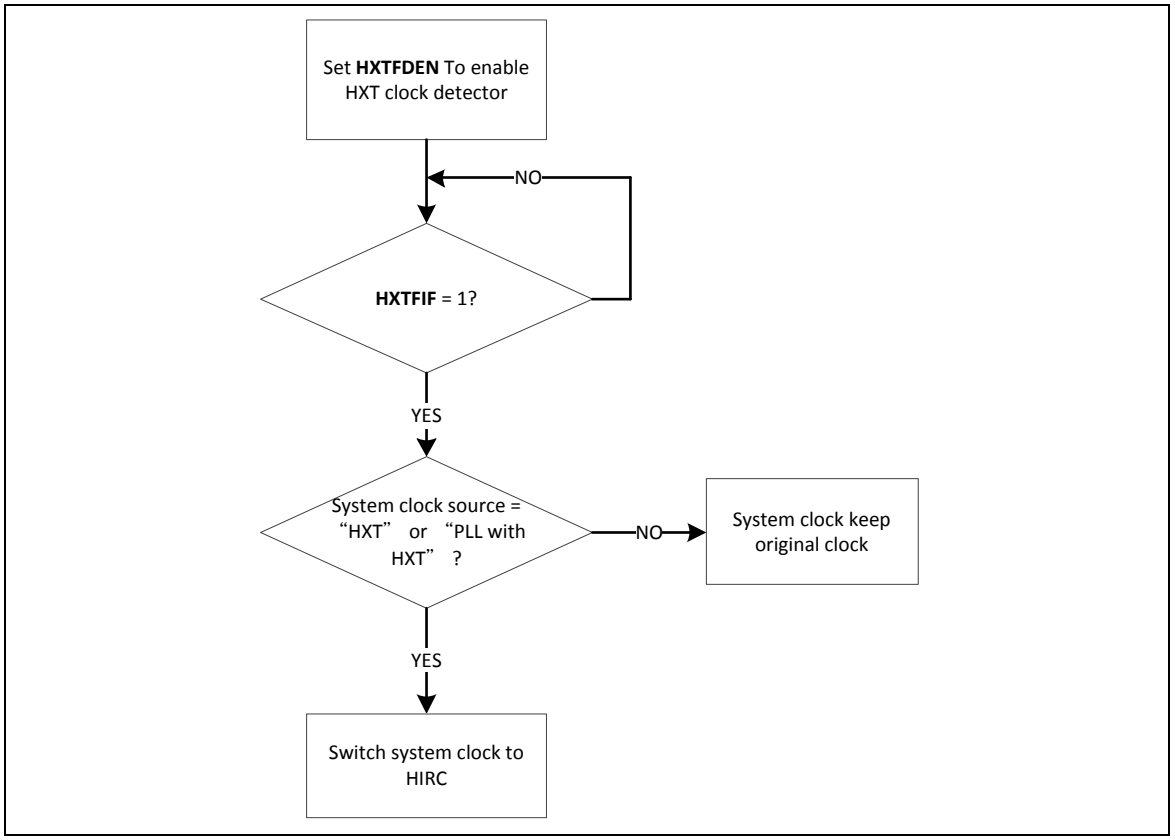
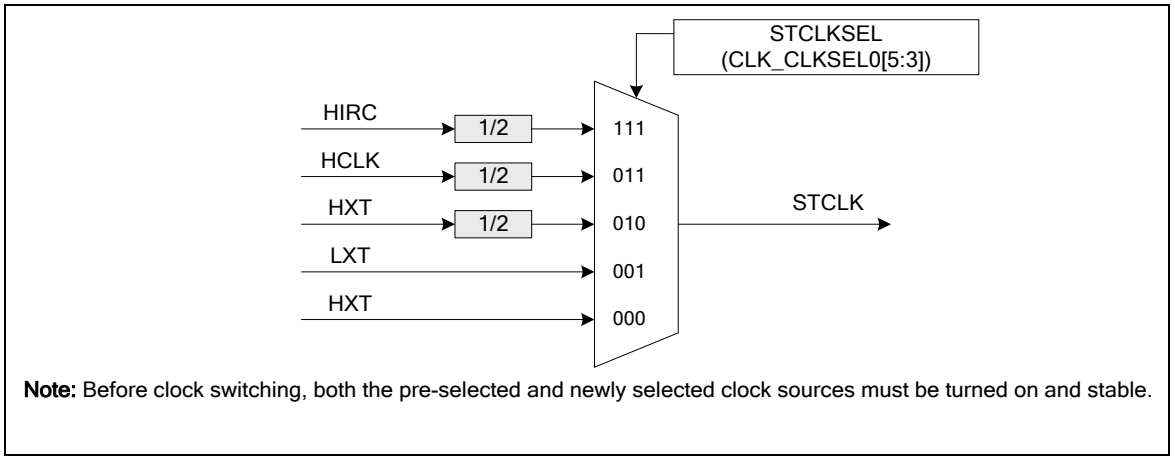


Figure 6.3-4 HXT Stop Protect Procedure

The clock source of SysTick in Cortex®-M4 core can use CPU clock or external clock (SYST_CTRL[2]). If using external clock, the SysTick clock (STCLK) has 5 clock sources. The clock source switch depends on the setting of the register STCLKSEL (CLK_CLKSEL0[5:3]). The block diagram is shown in the Figure 6.3-5.



Note: Before clock switching, both the pre-selected and newly selected clock sources must be turned on and stable.

Figure 6.3-5 SysTick Clock Control Block Diagram

6.3.4 Peripherals Clock

The peripherals clock has different clock source switch setting, which depends on the different peripheral. Please refer to the CLK_CLKSEL1 and CLK_CLKSEL2 register description in 5.3.8.

6.3.5 Power-down Mode Clock

When entering Power-down mode, system clocks, some clock sources and some peripheral clocks are disabled. Some clock sources and peripherals clock are still active in Power-down mode.

For theses clocks, which still keep active, are listed below:

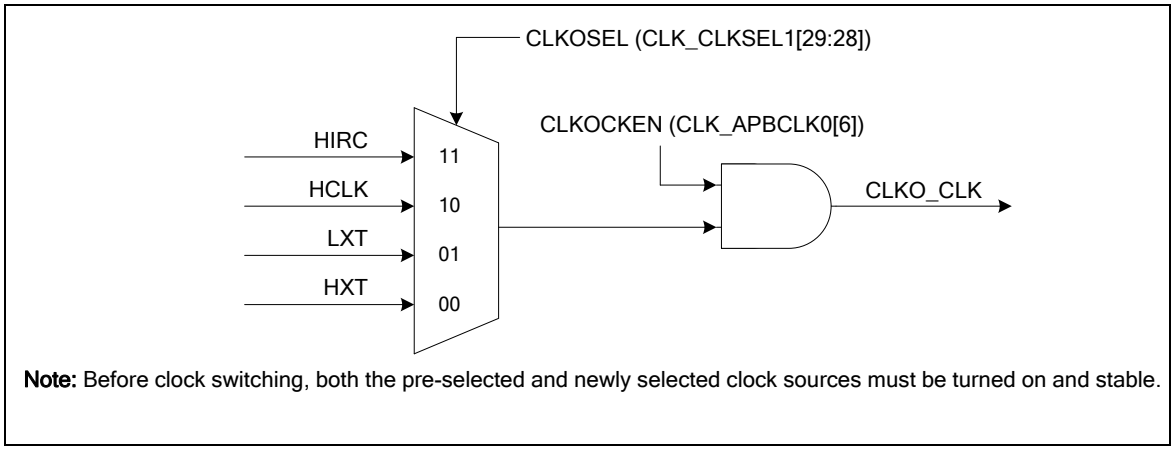
- Clock Generator
  - 10 kHz internal low speed RC oscillator (LIRC) clock
  - 32.768 kHz external low speed crystal oscillator (LXT) clock
- Peripherals Clock (When the modules adopt LXT or LIRC as clock source)

6.3.6 Clock Output

This device is equipped with a power-of-2 frequency divider which is composed by16 chained divide-by-2 shift registers. One of the 16 shift register outputs selected by a sixteen to one multiplexer is reflected to CLKO function pin. Therefore there are 16 options of power-of-2 divided clocks with the frequency from $F_{in}/2^1$ to $F_{in}/2^{16}$ where F_{in} is input clock frequency to the clock divider.

The output formula is $\mathbf{F_{out} = F_{in}/2^{(N+1)}}$, where F_{in} is the input clock frequency, F_{out} is the clock divider output frequency and N is the 4-bit value in FREQSEL (CLK_CLKOCTL[3:0]).

When writing 1 to CLKOEN (CLK_CLKOCTL[4]), the chained counter starts to count. When writing 0 to CLKOEN (CLK_CLKOCTL[4]), the chained counter continuously runs till divided clock reaches low state and stays in low state.



Note: Before clock switching, both the pre-selected and newly selected clock sources must be turned on and stable.

Figure 6.3-6 Clock Source of Clock Output

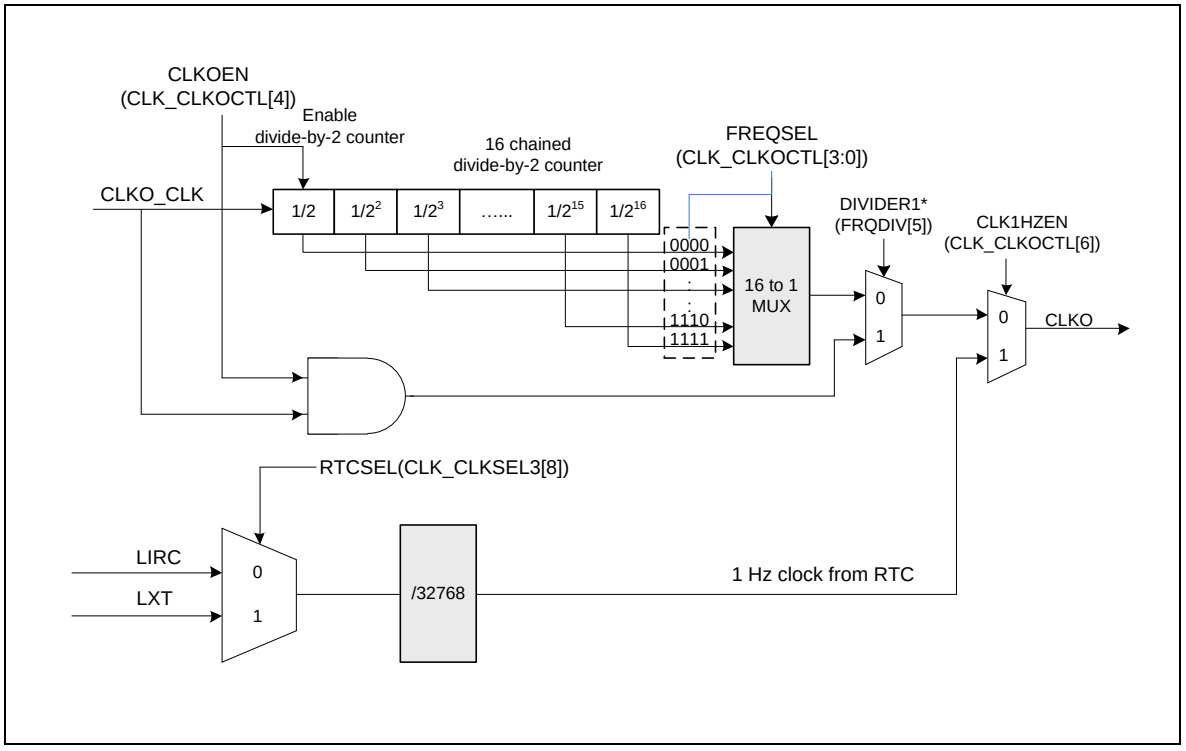


Figure 6.3-7 Clock Output Block Diagram

6.3.7 Register Map

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

Register	Offset	R/W	Description	Reset Value
CLK Base Address: **CLK_BA = 0x4000_0200**				
CLK_PWRCTL	CLK_BA+0x00	R/W	System Power-down Control Register	0x0000_001X
CLK_AHBCLK	CLK_BA+0x04	R/W	AHB Devices Clock Enable Control Register	0x0000_8004
CLK_APBCLK0	CLK_BA+0x08	R/W	APB Devices Clock Enable Control Register 0	0x0000_0001
CLK_APBCLK1	CLK_BA+0x0C	R/W	APB Devices Clock Enable Control Register 1	0x0000_0000
CLK_CLKSEL0	CLK_BA+0x10	R/W	Clock Source Select Control Register 0	0x0000_003X
CLK_CLKSEL1	CLK_BA+0x14	R/W	Clock Source Select Control Register 1	0xB377_770F
CLK_CLKSEL2	CLK_BA+0x18	R/W	Clock Source Select Control Register 2	0x0000_00AB
CLK_CLKSEL3	CLK_BA+0x1C	R/W	Clock Source Select Control Register 3	0x0000_0003
CLK_CLKDIV0	CLK_BA+0x20	R/W	Clock Divider Number Register 0	0x0000_0000
CLK_CLKDIV1	CLK_BA+0x24	R/W	Clock Divider Number Register 1	0x0000_0000
CLK_PLLCTL	CLK_BA+0x40	R/W	PLL Control Register	0x0005_C02E
CLK_STATUS	CLK_BA+0x50	R	Clock Status Monitor Register	0x0000_00XX
CLK_CLKOCTL	CLK_BA+0x60	R/W	Clock Output Control Register	0x0000_0000
CLK_CLKDCTL	CLK_BA+0x70	R/W	Clock Fail Detector Control Register	0x0000_0000
CLK_CLKDSTS	CLK_BA+0x74	R/W	Clock Fail Detector Status Register	0x0000_0000
CLK_CDUPB	CLK_BA+0x78	R/W	Clock Frequency Detector Upper Boundary Register	0x0000_0000
CLK_CDLOWB	CLK_BA+0x7c	R/W	Clock Frequency Detector Lower Boundary Register	0x0000_0000

6.3.8 Register Description

System Power-down Control Register (CLK_PWRCTL)

Register	Offset	R/W	Description	Reset Value
CLK_PWRCTL	CLK_BA+0x00	R/W	System Power-down Control Register	0x0000_001X

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			HXTSELTYP	HXTGAIN		Reserved	PDWTCPU
7	6	5	4	3	2	1	0
PDEN	PDWKIF	PDWKIEN	PDWKDLY	LIRCEN	HIRCEN	LXTEN	HXTEN

Bits	Description	
[31:13]	**Reserved**	Reserved.
[12]	**HXTSELTYP**	**HXT Crystal Type Select Bit (Write Protect)** This is a protected register. Please refer to open lock sequence to program it. 0 = Select INV type. 1 = Select GM type. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[11:10]	**HXTGAIN**	**HXT Gain Control Bit (Write Protect)** This is a protected register. Please refer to open lock sequence to program it. Gain control is used to enlarge the gain of crystal to make sure crystal work normally. If gain control is enabled, crystal will consume more power than gain control off. 00 = HXT frequency is lower than from 8 MHz. 01 = HXT frequency is from 8 MHz to 12 MHz. 10 = HXT frequency is from 12 MHz to 16 MHz. 11 = HXT frequency is higher than 16 MHz. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[9]	**Reserved**	Reserved.
[8]	**PDWTCPU**	**this Bit Control the Power-down Entry Condition (Write Protect)** 0 = Chip enters Power-down mode when the PDEN bit is set to 1. 1 = Chip enters Power-down mode when the both PDWTCPU and PDEN bits are set to 1 and CPU runs WFI instruction. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[7]	**PDEN**	**System Power-down Enable (Write Protect)** When this bit is set to 1, Power-down mode is enabled and chip Power-down behavior will depend on the PDWTCPU bit. (a) If the PDWTCPU is 0, then the chip enters Power-down mode immediately after the PDEN bit set. (default)

		(b) if the PDWTCPU is 1, then the chip keeps active till the CPU sleep mode is also active and then the chip enters Power-down mode. When chip wakes up from Power-down mode, this bit is auto cleared. Users need to set this bit again for next Power-down. In Power-down mode, HXT and the HIRC will be disabled in this mode, but LXT and LIRC are not controlled by Power-down mode. In Power-down mode, the PLL and system clock are disabled, and ignored the clock source selection. The clocks of peripheral are not controlled by Power-down mode, if the peripheral clock source is from LXT or LIRC. 0 = Chip operating normally or chip in idle mode because of WFI command. 1 = Chip enters Power-down mode instant or wait CPU sleep command WFI. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[6]	**PDWKIF**	**Power-down Mode Wake-up Interrupt Status** Set by “Power-down wake-up event”, it indicates that resume from Power-down mode” The flag is set if the EINT0~5, GPIO, USBH, USBD, OTG, UART0~3, WDT, CAN0, ACMP01, BOD, RTC, TMR0~3, I^2C0~1 or TK wake-up occurred. **Note1:** Write 1 to clear the bit to 0. **Note2:** This bit works only if PDWKIEN (CLK_PWRCTL[5]) set to 1.
[5]	**PDWKIEN**	**Power-down Mode Wake-up Interrupt Enable Bit (Write Protect)** 0 = Power-down mode wake-up interrupt Disabled. 1 = Power-down mode wake-up interrupt Enabled. **Note1:** The interrupt will occur when both PDWKIF and PDWKIEN are high. **Note2:** This bit is write protected. Refer to the SYS_REGLCTL register.
[4]	**PDWKDLY**	**Enable the Wake-up Delay Counter (Write Protect)** When the chip wakes up from Power-down mode, the clock control will delay certain clock cycles to wait system clock stable. The delayed clock cycle is 4096 clock cycles when chip works at 4~20 MHz external high speed crystal oscillator (HXT), and 256 clock cycles when chip works at 22.1184 MHz internal high speed RC oscillator (HIRC). 0 = Clock cycles delay Disabled. 1 = Clock cycles delay Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[3]	**LIRCEN**	**LIRC Enable Bit (Write Protect)** 0 = 10 kHz internal low speed RC oscillator (LIRC) Disabled. 1 = 10 kHz internal low speed RC oscillator (LIRC) Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2]	**HIRCEN**	**HIRC Enable Bit (Write Protect)** 0 = 22.1184 MHz internal high speed RC oscillator (HIRC) Disabled. 1 = 22.1184 MHz internal high speed RC oscillator (HIRC) Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[1]	**LXTEN**	**LXT Enable Bit (Write Protect)** 0 = 32.768 kHz external low speed crystal (LXT) Disabled. 1 = 32.768 kHz external low speed crystal (LXT) Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[0]	**HXTEN**	**HXT Enable Bit (Write Protect)** The bit default value is set by flash controller user configuration register CONFIG0 [26:24]. When the default clock source is from HXT, this bit is set to 1 automatically. 0 = 4~20 MHz xxternal high speed crystal (HXT) Disabled. 1 = 4~20 MHz external high speed crystal (HXT) Enabled.

		Note: This bit is write protected. Refer to the SYS_REGLCTL register.

Register/Instruction Mode	SLEEPDEEP (SCR[2])	PDEN (CLK_PWRCTL [7])	PDWTCPU (CLK_PWRCTL [8])	CPU Run WFI Instruction	Clock Disable
Normal operation	0	0	0	NO	All clocks are controlled by control register.
Idle mode (CPU enter Sleep mode)	0	0	0	YES	Only CPU clock is disabled.
Power-down mode (CPU enters Deep Sleep mode)	1	1	1	YES	Most clocks are disabled except LIRC/LXT, and only RTC/WDT/Timer peripheral clocks still enable if their clock sources are selected as LIRC/LXT.

Table 6-9 Power-down Mode Control Table

When the chip enters Power-down mode, user can wake up chip by some interrupt sources. User should enable the related interrupt sources and NVIC IRQ enable bits (NVIC_ISER) before set PDEN bit in CLK_PWRCTL[7] to ensure chip can enter Power-down and wake up successfully.

AHB Devices Clock Enable Control Register (CLK_AHBCLK)

The bits in this register are used to enable/disable clock for system clock, AHB bus devices clock.

Register	Offset	R/W	Description	Reset Value
CLK_AHBCLK	CLK_BA+0x04	R/W	AHB Devices Clock Enable Control Register	0x0000_8004

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
FMCIDLE	Reserved						
7	6	5	4	3	2	1	0
CRCCKEN	Reserved		USBHCKEN	EBICKEN	ISPCKEN	PDMACKEN	Reserved

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**FMCIDLE**	**Flash Memory Controller Clock Enable Bit in IDLE Mode** 0 = FMC peripheral clock Disabled when chip operating at IDLE mode. 1 = FMC peripheral clock Enabled when chip operating at IDLE mode.
[14:8]	**Reserved**	Reserved.
[7]	**CRCCKEN**	**CRC Generator Controller Clock Enable Bit** 0 = CRC peripheral clock Disabled. 1 = CRC peripheral clock Enabled.
[6:5]	**Reserved**	Reserved.
[4]	**USBHCKEN**	**USB HOST Controller Clock Enable Bit** 0 = USB HOST peripheral clock Disabled. 1 = USB HOST peripheral clock Enabled.
[3]	**EBICKEN**	**EBI Controller Clock Enable Bit** 0 = EBI peripheral clock Disabled. 1 = EBI peripheral clock Enabled.
[2]	**ISPCKEN**	**Flash ISP Controller Clock Enable Bit** 0 = Flash ISP peripheral clock Disabled. 1 = Flash ISP peripheral clock Enabled.
[1]	**PDMACKEN**	**PDMA Controller Clock Enable Bit** 0 = PDMA peripheral clock Disabled. 1 = PDMA peripheral clock Enabled.
[0]	**Reserved**	Reserved.

APB Devices Clock Enable Control Register (CLK_APBCLK0)

The bits in this register are used to enable/disable clock for peripheral controller clocks.

Register	Offset	R/W	Description	Reset Value
CLK_APBCLK0	CLK_BA+0x08	R/W	APB Devices Clock Enable Control Register 0	0x0000_0001

31	30	29	28	27	26	25	24
Reserved			EADCCKEN	USBDCKEN	OTGCKEN	Reserved	CAN0CKEN
23	22	21	20	19	18	17	16
Reserved				UART3CKEN	UART2CKEN	UART1CKEN	UART0CKEN
15	14	13	12	11	10	9	8
Reserved	SPI2CKEN	SPI1CKEN	SPI0CKEN	Reserved		I2C1CKEN	I2C0CKEN
7	6	5	4	3	2	1	0
ACMP01CKEN	CLKOCKEN	TMR3CKEN	TMR2CKEN	TMR1CKEN	TMR0CKEN	RTCCKEN	WDTCKEN

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28]	**EADCCKEN**	**Enhanced Analog-digital-converter (EADC) Clock Enable Bit** 0 = EADC clock Disabled. 1 = EADC clock Enabled.
[27]	**USBDCKEN**	**USB Device Clock Enable Bit** 0 = USB Device clock Disabled. 1 = USB Device clock Enabled.
[26]	**OTGCKEN**	**USB OTG Clock Enable Bit** 0 = USB OTG clock Disabled. 1 = USB OTG clock Enabled.
[25]	**Reserved**	Reserved.
[24]	**CAN0CKEN**	**CAN0 Clock Enable Bit** 0 = CAN0 clock Disabled. 1 = CAN0 clock Enabled.
[23:20]	**Reserved**	Reserved.
[19]	**UART3CKEN**	**UART3 Clock Enable Bit** 0 = UART3 clock Disabled. 1 = UART3 clock Enabled.
[18]	**UART2CKEN**	**UART2 Clock Enable Bit** 0 = UART2 clock Disabled. 1 = UART2 clock Enabled.
[17]	**UART1CKEN**	**UART1 Clock Enable Bit** 0 = UART1 clock Disabled.

		1 = UART1 clock Enabled.
[16]	**UART0CKEN**	**UART0 Clock Enable Bit** 0 = UART0 clock Disabled. 1 = UART0 clock Enabled.
[15]	**Reserved**	Reserved.
[14]	**SPI2CKEN**	**SPI2 Clock Enable Bit** 0 = SPI2 clock Disabled. 1 = SPI2 clock Enabled.
[13]	**SPI1CKEN**	**SPI1 Clock Enable Bit** 0 = SPI1 clock Disabled. 1 = SPI1 clock Enabled.
[12]	**SPI0CKEN**	**SPI0 Clock Enable Bit** 0 = SPI0 clock Disabled. 1 = SPI0 clock Enabled.
[11:10]	**Reserved**	Reserved.
[9]	**I2C1CKEN**	**I2C1 Clock Enable Bit** 0 = I2C1 clock Disabled. 1 = I2C1 clock Enabled.
[8]	**I2C0CKEN**	**I2C0 Clock Enable Bit** 0 = I2C0 clock Disabled. 1 = I2C0 clock Enabled.
[7]	**ACMP01CKEN**	**Analog Comparator 0/1 Clock Enable Bit** 0 = Analog comparator 0/1 clock Disabled. 1 = Analog comparator 0/1 clock Enabled.
[6]	**CLKOCKEN**	**CLKO Clock Enable Bit** 0 = CLKO clock Disabled. 1 = CLKO clock Enabled.
[5]	**TMR3CKEN**	**Timer3 Clock Enable Bit** 0 = Timer3 clock Disabled. 1 = Timer3 clock Enabled.
[4]	**TMR2CKEN**	**Timer2 Clock Enable Bit** 0 = Timer2 clock Disabled. 1 = Timer2 clock Enabled.
[3]	**TMR1CKEN**	**Timer1 Clock Enable Bit** 0 = Timer1 clock Disabled. 1 = Timer1 clock Enabled.
[2]	**TMR0CKEN**	**Timer0 Clock Enable Bit** 0 = Timer0 clock Disabled. 1 = Timer0 clock Enabled.
[1]	**RTCCKEN**	**Real-time-clock APB Interface Clock Enable Bit** This bit is used to control the RTC APB clock only. The RTC peripheral clock source is selected from RTCSEL(CLK_CLKSEL3[8]). It can be selected to 32.768 kHz external low speed crystal or 10 kHz internal low speed RC oscillator (LIRC). 0 = RTC APB clock Disabled.

		1 = RTC APB clock Enabled.
[0]	**WDTCKEN**	**Watchdog Timer Clock Enable Bit (Write Protect)** 0 = Watchdog timer clock Disabled. 1 = Watchdog timer clock Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

APB Devices Clock Enable Control Register 1 (CLK_APBCLK1)

The bits in this register are used to enable/disable clock for peripheral controller clocks.

Register	Offset	R/W	Description	Reset Value
CLK_APBCLK1	CLK_BA+0x0C	R/W	APB Devices Clock Enable Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						TKCKEN	Reserved
23	22	21	20	19	18	17	16
Reserved						PWM1CKEN	PWM0CKEN
15	14	13	12	11	10	9	8
Reserved			DACCKEN	Reserved			
7	6	5	4	3	2	1	0
Reserved							SC0CKEN

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**TKCKEN**	**Touch Key Clock Enable Bit** 0 = Touch key clock Disabled. 1 = Touch key clock Enabled.
[24:18]	**Reserved**	Reserved.
[17]	**PWM1CKEN**	**PWM1 Clock Enable Bit** 0 = PWM1 clock Disabled. 1 = PWM1 clock Enabled.
[16]	**PWM0CKEN**	**PWM0 Clock Enable Bit** 0 = PWM0 clock Disabled. 1 = PWM0 clock Enabled.
[15:13]	**Reserved**	Reserved.
[12]	**DACCKEN**	**DAC Clock Enable Bit** 0 = DAC clock Disabled. 1 = DAC clock Enabled.
[11:1]	**Reserved**	Reserved.
[0]	**SC0CKEN**	**SC0 Clock Enable Bit** 0 = SC0 clock Disabled. 1 = SC0 clock Enabled.

Clock Source Select Control Register 0 (CLK_CLKSEL0)

Register	Offset	R/W	Description	Reset Value
CLK_CLKSEL0	CLK_BA+0x10	R/W	Clock Source Select Control Register 0	0x0000_003X

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
PCLK1SEL	PCLK0SEL	STCLKSEL			HCLKSEL		

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**PCLK1SEL**	**PCLK1 Clock Source Selection (Write Protect)** 0 = APB1 bUS clock source from HCLK. 1 = APB1 bUS clock source from HCLK/2. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[6]	**PCLK0SEL**	**PCLK0 Clock Source Selection (Write Protect)** 0 = APB0 bUS clock source from HCLK. 1 = APB0 bUS clock source from HCLK/2. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[5:3]	**STCLKSEL**	**Cortex®-M4 SysTick Clock Source Selection (Write Protect)** If SYST_CTRL[2]=0, SysTick uses listed clock source below. 000 = Clock source from HXT. 001 = Clock source from LXT. 010 = Clock source from HXT/2. 011 = Clock source from HCLK/2. 111 = Clock source from HIRC/2. **Note:** if SysTick clock source is not from HCLK (i.e. SYST_CTRL[2] = 0), SysTick clock source must less than or equal to HCLK/2. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2:0]	**HCLKSEL**	**HCLK Clock Source Selection (Write Protect)** Before clock switching, the related clock sources (both pre-select and new-select) must be turned on. The default value is reloaded from the value of CFOSC (CONFIG0[26:24]) in user configuration register of Flash controller by any reset. Therefore the default value is either 000b or 111b. 000 = Clock source from HXT. 001 = Clock source from LXT. 010 = Clock source from PLL.

		011 = Clock source from LIRC. 111= Clock source from HIRC. Other = Reserved. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

Clock Source Select Control Register 1 (CLK_CLKSEL1)

Before clock switching, the related clock sources (pre-selected and newly-selected) must be turned on.

Register	Offset	R/W	Description	Reset Value
CLK_CLKSEL1	CLK_BA+0x14	R/W	Clock Source Select Control Register 1	0xB377_770F

31	30	29	28	27	26	25	24
WWDTSEL		CLKOSEL		Reserved		UARTSEL	
23	22	21	20	19	18	17	16
Reserved	TMR3SEL			Reserved	TMR2SEL		
15	14	13	12	11	10	9	8
Reserved	TMR1SEL			Reserved	TMR0SEL		
7	6	5	4	3	2	1	0
Reserved						WDTSEL	

Bits	Description	
[31:30]	**WWDTSEL**	**Window Watchdog Timer Clock Source Selection** 10 = Clock source from HCLK/2048. 11 = Clock source from 10 kHz internal low speed RC oscillator (LIRC). Others = Reserved.
[29:28]	**CLKOSEL**	**Clock Divider Clock Source Selection** 00 = Clock source from 4~20 MHz external high speed crystal oscillator (HXT). 01 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 10 = Clock source from HCLK. 11 = Clock source from 22.1184 MHz internal high speed RC oscillator (HIRC).
[27:26]	**Reserved**	Reserved.
[25:24]	**UARTSEL**	**UART Clock Source Selection** 00 = Clock source from 4~20 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL. 10 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 11 = Clock source from 22.1184 MHz internal high speed RC oscillator (HIRC).
[23]	**Reserved**	Reserved.
[22:20]	**TMR3SEL**	**TIMER3 Clock Source Selection** 000 = Clock source from 4~20 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 010 = Clock source from PCLK1. 011 = Clock source from external clock T3 pin. 101 = Clock source from 10 kHz internal low speed RC oscillator (LIRC). 111 = Clock source from 22.1184 MHz internal high speed RC oscillator (HIRC). Others = Reserved.

[19]	**Reserved**	Reserved.
[18:16]	**TMR2SEL**	**TIMER2 Clock Source Selection** 000 = Clock source from 4~20 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 010 = Clock source from PCLK1. 011 = Clock source from external clock T2 pin. 101 = Clock source from 10 kHz internal low speed RC oscillator (LIRC). 111 = Clock source from 22.1184 MHz internal high speed RC oscillator (HIRC). Others = Reserved.
[15]	**Reserved**	Reserved.
[14:12]	**TMR1SEL**	**TIMER1 Clock Source Selection** 000 = Clock source from 4~20 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 010 = Clock source from PCLK0. 011 = Clock source from external clock T1 pin. 101 = Clock source from 10 kHz internal low speed RC oscillator (LIRC). 111 = Clock source from 22.1184 MHz internal high speed RC oscillator (HIRC). Others = Reserved.
[11]	**Reserved**	Reserved.
[10:8]	**TMR0SEL**	**TIMER0 Clock Source Selection** 000 = Clock source from 4~20 MHz external high speed crystal oscillator (HXT). 001 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 010 = Clock source from PCLK0. 011 = Clock source from external clock T0 pin. 101 = Clock source from 10 kHz internal low speed RC oscillator (LIRC). 111 = Clock source from 22.1184 MHz internal high speed RC oscillator (HIRC). Others = Reserved.
[7:2]	**Reserved**	Reserved.
[1:0]	**WDTSEL**	**Watchdog Timer Clock Source Selection (Write Protect)** 00 = Reserved. 01 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 10 = Clock source from HCLK/2048. 11 = Clock source from 10 kHz internal low speed RC oscillator (LIRC). **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

Clock Source Select Control Register 2 (CLK_CLKSEL2)

Before clock switching, the related clock sources (pre-select and new-select) must be turned on.

Register	Offset	R/W	Description	Reset Value
CLK_CLKSEL2	CLK_BA+0x18	R/W	Clock Source Select Control Register 2	0x0000_00AB

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
SPI2SEL		SPI1SEL		SPI0SEL		PWM1SEL	PWM0SEL

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:6]	**SPI2SEL**	**SPI2 Clock Source Selection** 00 = Clock source from 4~20 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL. 10 = Clock source from PCLK0. 11 = Clock source from 22.1184 MHz internal high speed RC oscillator (HIRC).
[5:4]	**SPI1SEL**	**SPI1 Clock Source Selection** 00 = Clock source from 4~20 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL. 10 = Clock source from PCLK1. 11 = Clock source from 22.1184 MHz internal high speed RC oscillator (HIRC).
[3:2]	**SPI0SEL**	**SPI0 Clock Source Selection** 00 = Clock source from 4~20 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL. 10 = Clock source from PCLK0. 11 = Clock source from 22.1184 MHz internal high speed RC oscillator (HIRC).
[1]	**PWM1SEL**	**PWM1 Clock Source Selection** The peripheral clock source of PWM1 is defined by PWM1SEL. 0 = Clock source from PLL. 1 = Clock source from PCLK1.
[0]	**PWM0SEL**	**PWM0 Clock Source Selection** The peripheral clock source of PWM0 is defined by PWM0SEL. 0 = Clock source from PLL. 1 = Clock source from PCLK0.

Clock Source Select Control Register 3 (CLK_CLKSEL3)

Before clock switching, the related clock sources (pre-select and new-select) must be turned on.

Register	Offset	R/W	Description	Reset Value
CLK_CLKSEL3	CLK_BA+0x1C	R/W	Clock Source Select Control Register 3	0x0000_0003

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							RTCSEL
7	6	5	4	3	2	1	0
Reserved						SC0SEL	

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**RTCSEL**	**RTC Clock Source Selection** 0 = Clock source from 32.768 kHz external low speed crystal oscillator (LXT). 1 = Clock source from 10 kHz internal low speed RC oscillator (LIRC).
[7:2]	**Reserved**	Reserved.
[1:0]	**SC0SEL**	**SC0 Clock Source Selection** 00 = Clock source from 4~20 MHz external high speed crystal oscillator (HXT). 01 = Clock source from PLL. 10 = Clock source from PCLK0. 11 = Clock source from 22.1184 MHz internal high speed RC oscillator (HIRC).

Clock Divider Number Register 0 (CLK_CLKDIV0)

Register	Offset	R/W	Description	Reset Value
CLK_CLKDIV0	CLK_BA+0x20	R/W	Clock Divider Number Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
EADCDIV							
15	14	13	12	11	10	9	8
Reserved				UARTDIV			
7	6	5	4	3	2	1	0
USBDIV				HCLKDIV			

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:16]	**EADCDIV**	**EADC Clock Divide Number From EADC Clock Source** EADC clock frequency = (EADC clock source frequency) / (EADCDIV + 1).
[15:12]	**Reserved**	Reserved.
[11:8]	**UARTDIV**	**UART Clock Divide Number From UART Clock Source** UART clock frequency = (UART clock source frequency) / (UARTDIV + 1).
[7:4]	**USBDIV**	**USB Clock Divide Number From PLL Clock** USB clock frequency = (PLL frequency) / (USBDIV + 1).
[3:0]	**HCLKDIV**	**HCLK Clock Divide Number From HCLK Clock Source** HCLK clock frequency = (HCLK clock source frequency) / (HCLKDIV + 1).

Clock Divider Number Register 1 (CLK_CLKDIV1)

Register	Offset	R/W	Description	Reset Value
CLK_CLKDIV1	CLK_BA+0x24	R/W	Clock Divider Number Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SC0DIV							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**SC0DIV**	**SC0 Clock Divide Number From SC0 Clock Source** SC0 clock frequency = (SC0 clock source frequency) / (SC0DIV + 1).

PLL Control Register (CLK_PLLCTL)

The PLL reference clock input is from the 4~20 MHz external high speed crystal oscillator (HXT) clock input or from the 22.1184 MHz internal high speed RC oscillator (HIRC). This register is used to control the PLL output frequency and PLL operation mode.

Programming these bits needs to write “59h”, “16h”, “88h” to address 0x4000_0100 to disable register protection. Refer to the register SYS_REGLCTL at address SYS_BA+0x100.

Register	Offset	R/W	Description	Reset Value
CLK_PLLCTL	CLK_BA+0x40	R/W	PLL Control Register	0x0005_C02E

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
STBSEL	Reserved			PLLSRC	OE	BP	PD
15	14	13	12	11	10	9	8
OUTDIV		INDIV					FBDIV
7	6	5	4	3	2	1	0
FBDIV							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23]	**STBSEL**	**PLL Stable Counter Selection (Write Protect)** 0 = PLL stable time is 6144 PLL source clock (suitable for source clock is equal to or less than 12 MHz). 1 = PLL stable time is 12288 PLL source clock (suitable for source clock is larger than 12 MHz). **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[22:20]	**Reserved**	Reserved.
[19]	**PLLSRC**	**PLL Source Clock Selection (Write Protect)** 0 = PLL source clock from 4~20 MHz external high-speed crystal oscillator (HXT). 1 = PLL source clock from 22.1184 MHz internal high-speed oscillator (HIRC). **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[18]	**OE**	**PLL OE (FOUT Enable) Pin Control (Write Protect)** 0 = PLL FOUT Enabled. 1 = PLL FOUT is fixed low. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[17]	**BP**	**PLL Bypass Control (Write Protect)** 0 = PLL is in normal mode (default). 1 = PLL clock output is same as PLL input clock FIN. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[16]	**PD**	**Power-down Mode (Write Protect)** If set the PDEN bit to 1 in CLK_PWRCTL register, the PLL will enter Power-down mode,

		too. 0 = PLL is in normal mode. 1 = PLL is in Power-down mode (default). **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[15:14]	**OUTDIV**	**PLL Output Divider Control (Write Protect)** Refer to the formulas below the table. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[13:9]	**INDIV**	**PLL Input Divider Control (Write Protect)** Refer to the formulas below the table. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[8:0]	**FBDIV**	**PLL Feedback Divider Control (Write Protect)** Refer to the formulas below the table. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

Output Clock Frequency Setting

$$FOUT = FIN \times \frac{NF}{NR} \times \frac{1}{NO}$$

Constraint:

1. $3.2MHz < FIN < 150MHz$
2. $800kHz < \frac{FIN}{2 * NR} < 8MHz$
3. $200MHz < FCO = FIN * \frac{NF}{NR} < 500MHz,$
   $FCO > 250MHz \ \ is \ \ preferred$

Symbol	Description
FOUT	Output Clock Frequency
FIN	Input (Reference) Clock Frequency
NR	Input Divider (INDIV + 2)
NF	Feedback Divider (FBDIV + 2)
NO	OUTDIV = “00” : NO = 1 OUTDIV = “01” : NO = 2 OUTDIV = “10” : NO = 2 OUTDIV = “11” : NO = 4

Clock Status Monitor Register (CLK_STATUS)

The bits in this register are used to monitor if the chip clock source is stable or not, and whether the clock switch is failed.

Register	Offset	R/W	Description	Reset Value
CLK_STATUS	CLK_BA+0x50	R	Clock Status Monitor Register	0x0000_00XX

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
CLKSFAIL	Reserved		HIRCSTB	LIRCSTB	PLLSTB	LXTSTB	HXTSTB

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**CLKSFAIL**	**Clock Switching Fail Flag (Read Only)** This bit is updated when software switches system clock source. If switch target clock is stable, this bit will be set to 0. If switch target clock is not stable, this bit will be set to 1. 0 = Clock switching success. 1 = Clock switching failure. **Note:** Write 1 to clear the bit to 0.
[6:5]	**Reserved**	Reserved.
[4]	**HIRCSTB**	**HIRC Clock Source Stable Flag (Read Only)** 0 = 22.1184 MHz internal high speed RC oscillator (HIRC) clock is not stable or disabled. 1 = 22.1184 MHz internal high speed RC oscillator (HIRC) clock is stabe and enabled.
[3]	**LIRCSTB**	**LIRC Clock Source Stable Flag (Read Only)** 0 = 10 kHz internal low speed RC oscillator (LIRC) clock is not stable or disabled. 1 = 10 kHz internal low speed RC oscillator (LIRC) clock is stable and enabled.
[2]	**PLLSTB**	**Internal PLL Clock Source Stable Flag (Read Only)** 0 = Internal PLL clock is not stable or disabled. 1 = Internal PLL clock is stable and enabled.
[1]	**LXTSTB**	**LXT Clock Source Stable Flag (Read Only)** 0 = 32.768 kHz external low speed crystal oscillator (LXT) clock is not stable or disabled. 1 = 32.768 kHz external low speed crystal oscillator (LXT) clock is stabled and enabled.
[0]	**HXTSTB**	**HXT Clock Source Stable Flag (Read Only)** 0 = 4~20 MHz external high speed crystal oscillator (HXT) clock is not stable or disabled. 1 = 4~20 MHz external high speed crystal oscillator (HXT) clock is stable and enabled.

Clock Output Control Register (CLK_CLKOCTL)

Register	Offset	R/W	Description	Reset Value
CLK_CLKOCTL	CLK_BA+0x60	R/W	Clock Output Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	CLK1HZEN	DIV1EN	CLKOEN	FREQSEL			

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**CLK1HZEN**	**Clock Output 1Hz Enable Bit** 0 = 1 Hz clock output for 32.768 kHz frequency compensation Disabled. 1 = 1 Hz clock output for 32.768 kHz frequency compensation Enabled.
[5]	**DIV1EN**	**Clock Output Divide One Enable Bit** 0 = Clock Output will output clock with source frequency divided by FREQSEL. 1 = Clock Output will output clock with source frequency.
[4]	**CLKOEN**	**Clock Output Enable Bit** 0 = Clock Output function Disabled. 1 = Clock Output function Enabled.
[3:0]	**FREQSEL**	**Clock Output Frequency Selection** The formula of output frequency is $\mathbf{F_{out} = F_{in}/2^{(N+1)}}$. F_{in} is the input clock frequency. F_{out} is the frequency of divider output clock. N is the 4-bit value of FREQSEL[3:0].

Clock Fail Detector Control Register (CLK_CLKDCTL)

Register	Offset	R/W	Description	Reset Value
CLK_CLKDCTL	CLK_BA+0x70	R/W	Clock Fail Detector Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						HXTFQIEN	HXTFQDEN
15	14	13	12	11	10	9	8
Reserved		LXTFIEN	LXTFDEN	Reserved			
7	6	5	4	3	2	1	0
Reserved		HXTFIEN	HXTFDEN	Reserved			

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**HXTFQIEN**	**HXT Clock Frequency Monitor Interrupt Enable Bit** 0 = 4~20 MHz external high speed crystal oscillator (HXT) clock frequency monitor fail interrupt Disabled. 1 = 4~20 MHz external high speed crystal oscillator (HXT) clock frequency monitor fail interrupt Enabled.
[16]	**HXTFQDEN**	**HXT Clock Frequency Monitor Enable Bit** 0 = 4~20 MHz external high speed crystal oscillator (HXT) clock frequency monitor Disabled. 1 = 4~20 MHz external high speed crystal oscillator (HXT) clock frequency monitor Enabled.
[15:14]	**Reserved**	Reserved.
[13]	**LXTFIEN**	**LXT Clock Fail Interrupt Enable Bit** 0 = 32.768 kHz external low speed crystal oscillator (LXT) clock fail interrupt Disabled. 1 = 32.768 kHz external low speed crystal oscillator (LXT) clock fail interrupt Enabled.
[12]	**LXTFDEN**	**LXT Clock Fail Detector Enable Bit** 0 = 32.768 kHz external low speed crystal oscillator (LXT) clock fail detector Disabled. 1 = 32.768 kHz external low speed crystal oscillator (LXT) clock fail detector Enabled.
[11:6]	**Reserved**	Reserved.
[5]	**HXTFIEN**	**HXT Clock Fail Interrupt Enable Bit** 0 = 4~20 MHz external high speed crystal oscillator (HXT) clock fail interrupt Disabled. 1 = 4~20 MHz external high speed crystal oscillator (HXT) clock fail interrupt Enabled.
[4]	**HXTFDEN**	**HXT Clock Fail Detector Enable Bit** 0 = 4~20 MHz external high speed crystal oscillator (HXT) clock fail detector Disabled. 1 = 4~20 MHz external high speed crystal oscillator (HXT) clock fail detector Enabled.
[3:0]	**Reserved**	Reserved.

Clock Fail Detector Status Register (CLK_CLKDSTS)

Register	Offset	R/W	Description	Reset Value
CLK_CLKDSTS	CLK_BA+0x74	R/W	Clock Fail Detector Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							HXTFQIF
7	6	5	4	3	2	1	0
Reserved						LXTFIF	HXTFIF

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**HXTFQIF**	**HXT Clock Frequency Monitor Interrupt Flag** 0 = 4~20 MHz external high speed crystal oscillator (HXT) clock is normal. 1 = 4~20 MHz external high speed crystal oscillator (HXT) clock frequency is abnormal. **Note:** Write 1 to clear the bit to 0.
[7:2]	**Reserved**	Reserved.
[1]	**LXTFIF**	**LXT Clock Fail Interrupt Flag** 0 = 32.768 kHz external low speed crystal oscillator (LXT) clock is normal. 1 = 32.768 kHz external low speed crystal oscillator (LXT) stops. **Note:** Write 1 to clear the bit to 0.
[0]	**HXTFIF**	**HXT Clock Fail Interrupt Flag** 0 = 4~20 MHz external high speed crystal oscillator (HXT) clock is normal. 1 = 4~20 MHz external high speed crystal oscillator (HXT) clock stops. **Note:** Write 1 to clear the bit to 0.

Clock Frequency Detector Upper Boundary Register (CLK_CDUPB)

Register	Offset	R/W	Description	Reset Value
CLK_CDUPB	CLK_BA+0x78	R/W	Clock Frequency Detector Upper Boundary Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						UPERBD	
7	6	5	4	3	2	1	0
UPERBD							

Bits	Description	
[31:10]	Reserved	Reserved.
[9:0]	UPERBD	**HXT Clock Frequency Detector Upper Boundary** The bits define the high value of frequency monitor window. When HXT frequency monitor value higher than this register, the HXT frequency detect fail interrupt flag will set to 1.

Clock Frequency Detector Lower Boundary Register (CLK_CDLOWB)

Register	Offset	R/W	Description	Reset Value
CLK_CDLOWB	CLK_BA+0x7c	R/W	Clock Frequency Detector Lower Boundary Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						LOWERBD	
7	6	5	4	3	2	1	0
LOWERBD							

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:0]	**LOWERBD**	**HXT Clock Frequency Detector Lower Boundary** The bits define the low value of frequency monitor window. When HXT frequency monitor value lower than this register, the HXT frequency detect fail interrupt flag will set to 1.

6.4 Flash Memeory Controller (FMC)

6.4.1 Overview

The NuMicro® M4TK series is equipped with 128/256 KB on-chip embedded flash for application and configurable Data Flash to store some application dependent data. A User Configuration block provides for system initiation. A 4 KB loader ROM (LDROM) is used for In-System-Programming (ISP) function. A 16KB Boot Loader consists of native ISP functions. A 4KB cache with zero wait cycle is used to improve flash access performance. This chip also supports In-Application-Programming (IAP) function, user switches the code executing without the chip reset after the embedded flash updated.

6.4.2 Features

- Supports 128/256 KB application ROM (APROM).
- Supports 4 KB loader ROM (LDROM).
- Supports Data Flash with configurable memory size.
- Supports 8 bytes User Configuration block to control system initiation.
- Supports 2 KB page erase for all embedded flash.
- Supports Boot Loader with native In-System-Programming (ISP) functions.
- Supports 32-bit/64-bit and multi-word flash programming function.
- Supports fast flash programming verification function.
- Supports checksum calculation function.
- Supports In-System-Programming (ISP) / In-Application-Programming (IAP) to update embedded flash memory.
- Supports cache memory to improve flash access performance and reduce power consumption.

6.4.3 Block Diagram

The flash memory controller (FMC) consists of AHB slave interface, cache memory controller, boot loader, flash control registers, flash initialization controller, flash operation control and embedded flash memory. The block diagram of flash memory controller is shown as follows.

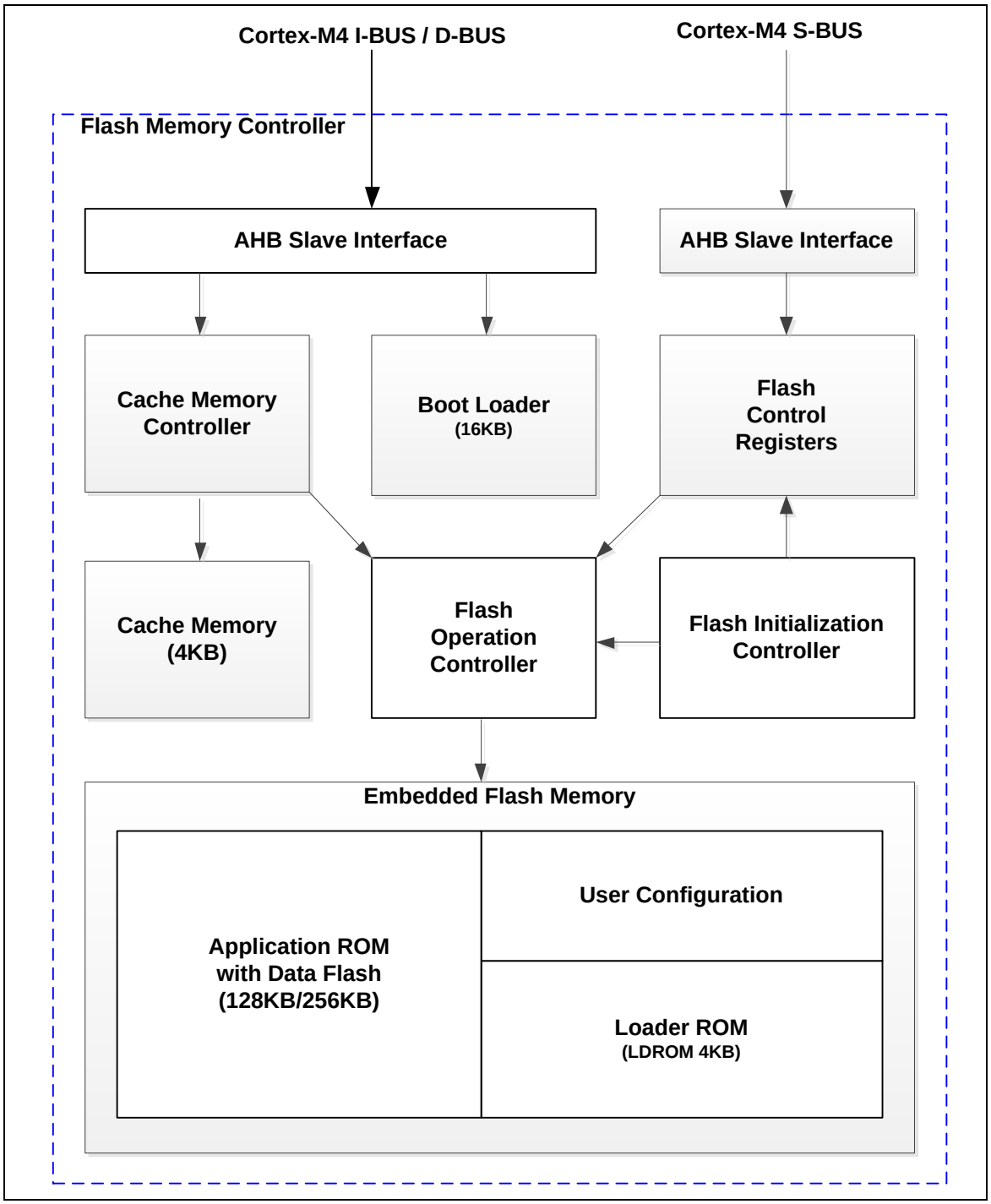


Figure 6.4-1 Flash Memory Controller Block Diagram

AHB Slave Interface

There are two AHB slave interfaces in flash memory controller, one is from both Cortex®-M4 I-Bus and D-Bus for the instruction and data fetch; the other is from Cortex®-M4 S-Bus for flash control registers access including ISP registers.

Cache Memory Controller

A 4 KB cache with zero wait cycle is implemented between Cortex®-M4 CPU and embedded flash memory. This cache memory controller improves the flash access performance and reduces power consumption of the embedded flash memory.

Boot Loader

The Boot Loader is 16KB, and consists of native ISP functions to update embedded flash memory. The Boot Loader content is read only, but not programmable.

Flash Control Registers

All of ISP control and status registers are in the flash control registers. The detail registers description is in the Register Description section

Flash Initialization Controller

When chip is powered on or active from reset, the flash initialization controller will start to access flash automatically and check the flash stability, and also reload User Configuration content to the flash control registers for system initiation.

Flash Operation Controller

The flash operations, such as flash erase, flash program, and flash read operation, have specific control timing for embedded flash memory. The flash operation controller generates those control timing by requested from the cache memory controller, the flash control registers and the flash initialization controller.

Embedded Flash Memory

The embedded flash memory is the main memory for user application code and parameters. It is consists of the user configuration block, 4KB LDROM and 128KB/ 256KB APROM with Data Flash. The page erase flash size is 2KB, and minimum program bit size is 32 bits.

6.4.4 Functional Description

FMC functions include the memory organization, boot selection, IAP, ISP, the embedded flash programming, and checksum calculation. The flash memory map and system memory map are also introduced in the memory organization.

6.4.4.1 Memory Organization

The FMC memory consists of the embedded flash memory and boot loader. The embedded flash memory is programmable, and includes APROM, LDROM, Data Flash and the User Configuration block. The address map includes flash memory map and five system address maps: LDROM with IAP, LDROM without IAP, APROM with IAP, APROM without IAP, and Boot Loader with IAP functions.

LDROM APROM and Data Flash

LDROM is designed for a loader to implement In-System-Programming (ISP) function by user. LDROM is a 4KB embedded flash memory, the flash address range is from 0x0010_0000 to 0x0010_0FFF. APROM is main memory for user applications. APROM size is 128KB or 256KB. Data Flash is used to store application parameters (not instruction). Data Flash is shared with APROM and size is configurable. The base address of Data Flash is determined by DFBA (CONFIG1[19:0]). All of embedded flash memory is 2KB page erased.

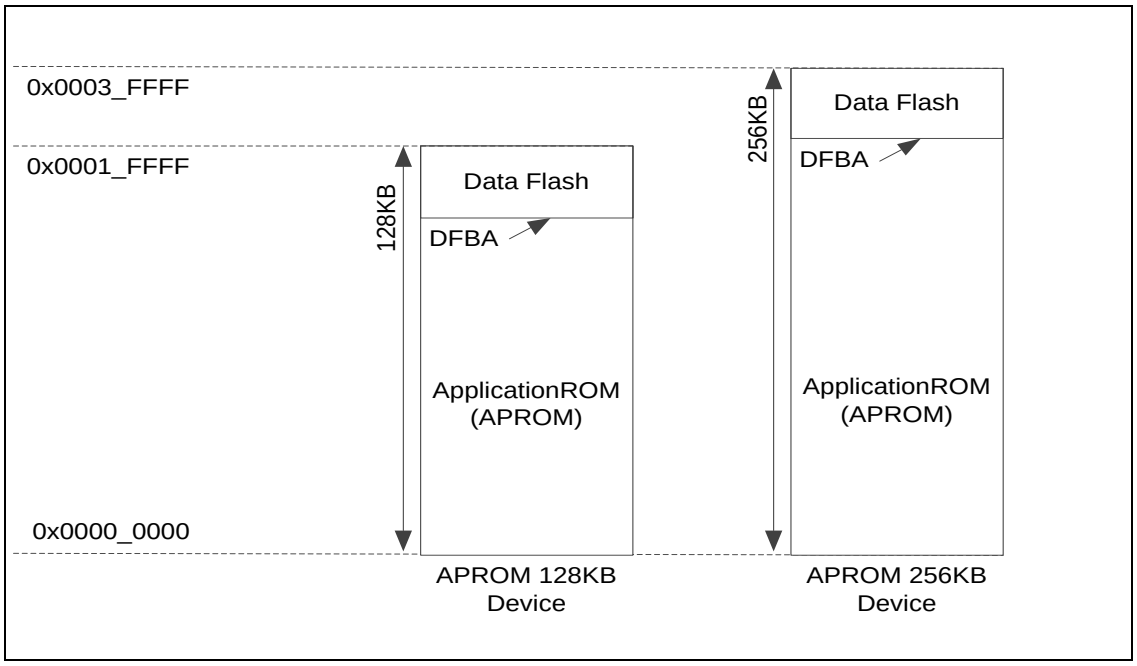


Figure 6.4-2 Data Flash Shared with APROM

User Configuration Block

User Configuration block is internal programmable configuration area for boot options, such as flash security lock, boot select, brown-out voltage level, and Data Flash base address. It works like a fuse for power on setting. It is loaded from flash memory to its corresponding control registers during chip power on. User can set these bits according to different application requests. User Configuration block can be updated by ISP function and located at 0x0030_0000 with two 32 bits words (CONFIG0 and CONFIG1). Any change on User Configuration block will take effect after system reboot

CONFIG0 (Address = 0x0030_0000)

31	30	29	28	27	26	25	24
CWDTEN[2]	CWDTPDEN	Reserved		CFGXT1	CFOSC	Reserved	
23	22	21	20	19	18	17	16
CBODEN	CBOV		CBORST	Reserved			
15	14	13	12	11	10	9	8
Reserved					CIOINI	Reserved	
7	6	5	4	3	2	1	0
CBS		MBS	CWDTEN[1:0]		Reserved	LOCK	DFEN

Bits	Descriptions	
[31]	**CWDTEN[2]**	**Watchdog Timer Hardware Enable Bit** When watchdog timer hardware enable function is enabled, the watchdog enable bit WDTEN (WDT_CTL[7]) and watchdog reset enable bit RSTEN (WDT_CTL[1]) is set to 1 automatically after power on. The clock source of watchdog timer is force at LIRC and LIRC can't be disabled. **CWDTEN[2:0]** is CONFIG0[31][4][3], 011 = WDT hardware enable function is active. WDT clock is always on except chip enters Power- down mode. When chip enter Power-down mode, WDT clock is always on if CWDTPDEN is 0 or WDT clock is controlled by LIRCEN (CLK_PWRCTL[3]) if CWDTPDEN is 1. Please refer to bit field description of CWDTPDEN. 111 = WDT hardware enable function is inactive. Others = WDT hardware enable function is active. WDT clock is always on.
[30]	**CWDTPDEN**	**Watchdog Clock Power-down Enable Bit** 0 = Watchdog Timer clock kept enabled when chip enters Power-down. 1 = Watchdog Timer clock is controlled by LIRCEN (CLK_PWRCTL[3]) when chip enters Power-down. **Note:** This bit only works if CWDTEN[2:0] is set to 011
[29:28]	**Reserved**	Reserved.
[27]	**CFGXT1**	**PF[4:3] Multi-Function Select** 0 = PF[4:3] pins are configured as GPIO pins. 1 =PF[4:3] pins are configured as external 4~20 MHz external high speed crystal oscillator (HXT) pins.
[26]	**CFOSC**	**CPU Clock Source Selection After Reset** The value of CFOSC will be loaded to HCLK (CLK_CLKSEL0[2:0]) in system clock controller after any reset occurs. HCLK[2:0] = 111 if CFOSC = 1, HCLK[2:0] = 000 if CFGSC=0. 0 = 4~20 MHz external high speed crystal oscillator (HXT) 1 = 22.1184 MHz internal high speed RC oscillator (HIRC)
[25:24]	**Reserved**	Reserved.
[23]	**CBODEN**	**Brown-Out Detector Enable Bit** 0= Brown-out detect Enabled after powered on. 1= Brown-out detect Disabled after powered on.

[22:21]	**CBOV**	**Brown-Out Voltage Selection** 00 = Brown-out voltage is 2.2V. 01 = Brown-out voltage is 2.7V. 10 = Brown-out voltage is 3.7V. 11 = Brown-out voltage is 4.5V.
[20]	**CBORST**	**Brown-Out Reset Enable Bit** 0 = Brown-out reset Enabled after powered on. 1 = Brown-out reset Disabled after powered on.
[19:11]	**Reserved**	Reserved.
[10]	**CIOINI**	**I/O Initial State Selection** 0 = All GPIO set as Quasi-bidirectional mode after chip powered on. 1 = All GPIO set as input tri-state mode after powered on.
[9:8]	**Reserved**	Reserved.
[7:6]	**CBS**	**Chip Booting Selection** When CBS[0] = 0, the LDROM base address is mapping to 0x100000 and APROM base address is mapping to 0x0. User could access both APROM and LDROM without boot switching. In other words, if IAP mode is supported, the code in LDROM and APROM can be called by each other. CBS value is valid when MBS =1. 00 = Boot from LDROM with IAP mode. 01 = Boot from LDROM without IAP mode. 10 = Boot from APROM with IAP mode. 11 = Boot from APROM without IAP mode. **Note:** BS (FMC_ISPCTL[1]) is only be used to control boot switching when CBS[0] = 1 and MBS = 1. VECMAP (FMC_ISPSTS[23:9]) is only be used to remap 0x0~0x1ff when CBS[0] = 0 or MBS = 0.
[5]	**MBS**	**Boot Loader Booting Selection** 0 = Booting from Boot Loader, and ignored CBS setting. 1 = Booting from APROM or LDROM, depended on CBS value. **Note:** BS (FMC_ISPCTL[1]) is only be used to control boot switching when CBS[0] = 1 and MBS = 1. VECMAP (FMC_ISPSTS[23:9]) is only be used to remap 0x0~0x1ff when CBS[0] = 0 or MBS = 0.
[4:3]	**CWDTEN[1:0]**	**Watchdog Timer Hardware Enable Bit** When watchdog timer hardware enable function is enabled, the watchdog enable bit WDTEN (WDT_CTL[7]) and watchdog reset enable bit RSTEN (WDT_CTL[1]) is set to 1 automatically after power on. The clock source of watchdog timer is force at LIRC and LIRC can't be disable. **CWDTEN[2:0]** is CONFIG0[31][4][3], 011 = WDT hardware enable function is active. WDT clock is always on except chip enter Power-down mode. When chip enter Power-down mode, WDT clock is always on if CWDTPDEN is 0 or WDT clock is controlled by LIRCEN (CLK_PWRCTL[3]) if CWDTPDEN is 1. Please refer to bit field description of CWDTPDEN. 111 = WDT hardware enable function is inactive. Others = WDT hardware enable function is active. WDT clock is always on.
[2]	**Reserved**	Reserved.

[1]	**LOCK**	**Security Lock Control** 0 = Flash memory content is locked. 1 = Flash memory content is not locked.
[0]	**DFEN**	**Data Flash Enable Bit** The Data Flash is shared with APROM, and the base address of Data Flash is decided by DFBA (CONFIG1[19:0]) when DFEN is 0. 0 = Data Flash Enabled. 1 = Data Flash Disabled.

CONFIG1 (Address = 0x0030_0004)

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved				DFBA			
15	14	13	12	11	10	9	8
DFBA							
7	6	5	4	3	2	1	0
DFBA							

Bits	Descriptions	
[31:20]	**Reserved**	Reserved.
[19:0]	**DFBA**	**Data Flash Base Address** This register works only when DFEN (CONFIG0[0])set to 0. If DFEN (CONFIG0[0]) is set to 0, the Data Flash base address is defined by user. Since on-chip flash erase unit is 2 KB, it is mandatory to keep bit 10-0 as 0.

Flash Memory Map

In the NuMicro® M4TK series, the flash memory map is different from system memory map. The system memory map is used by CPU fetch code or data from FMC memory. The flash memory map is used for ISP function to read, program or erase FMC memory. Figure 6.4-4 shows the flash memory map.

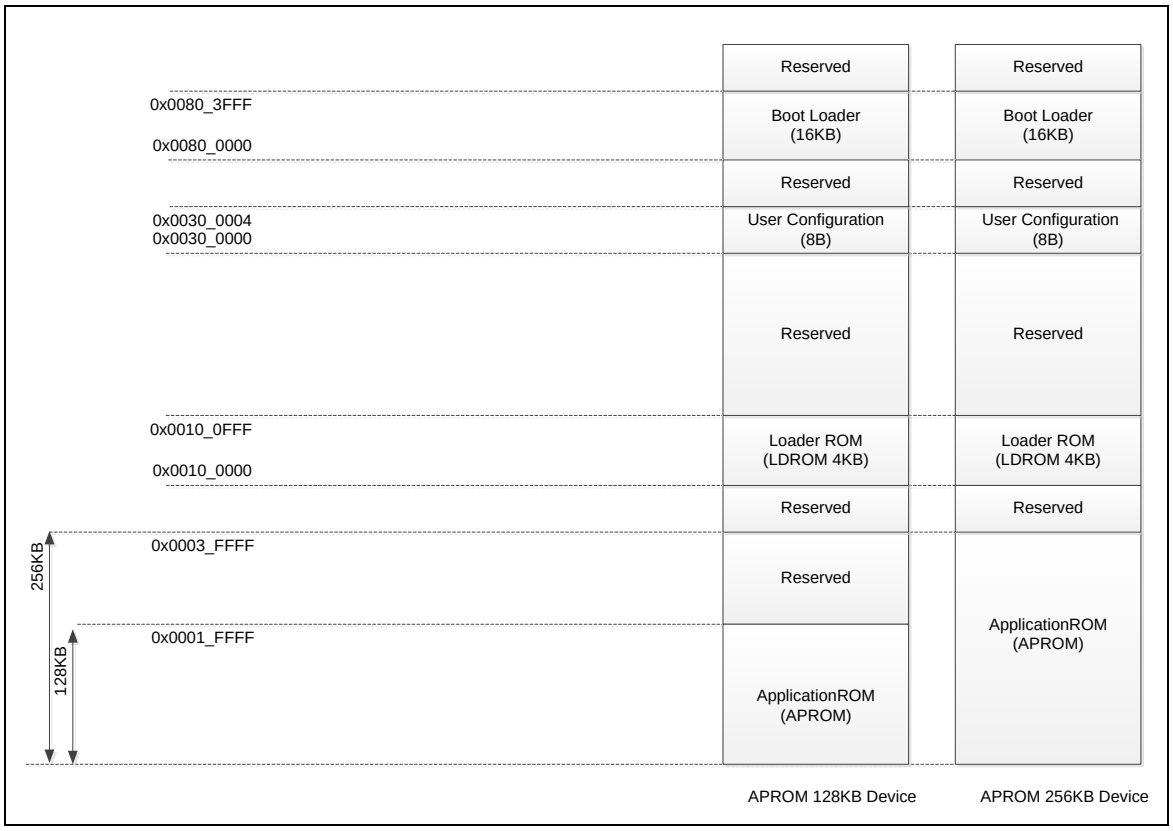


Figure 6.4-3 Flash Memory Map

System Memory Map with IAP mode

The system memory map is used by CPU to fetch code or data from FMC memory. Boot Loader(0x0080_0000~0x0080_3FFF) and LDROM(0x0010_0000~0x0010_0FFF) address map are the same as in the flash memory map. The Data Flash is shared with APROM and the Data Flash base address is defined by CONFIG1. The content of CONFIG1 is loaded into DFBA (Data Flash Base Address Register) at the flash initiation. The DFBA~(0x0001_FFFF/0x0003_FFFF) is the Data Flash region for Cortex®-M4 data access, and 0x0000_0200~(DFBA-1) is APROM region for Cortex®-M4 instruction access.

The address from 0x0000_0000 to 0x0000_01FF is called system memory vector. APROM, LDROM and Boot loader can map to the system memory vector for CPU start up. There are three kinds of system memory map with IAP mode when chip booting: (1) LDROM with IAP, (2) APROM with IAP, and (3) Boot Loader with IAP.

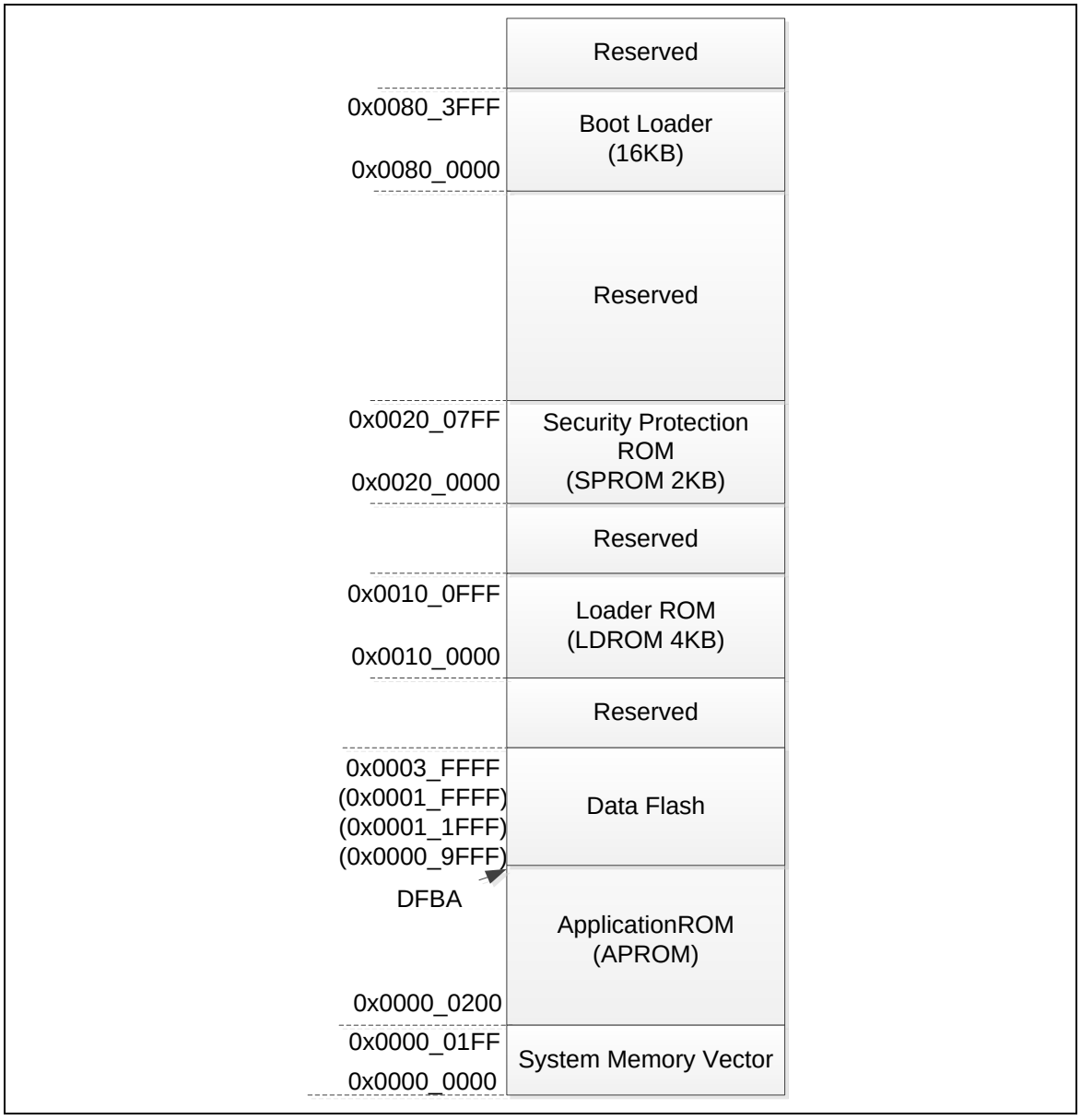


Figure 6.4-4 System Memory Map with IAP Mode

In LDROM with IAP mode, the LDROM (0x0010_0000~0x0010_01FF) is mapping to the system memory vector for Cortex®-M4 instruction or data access.

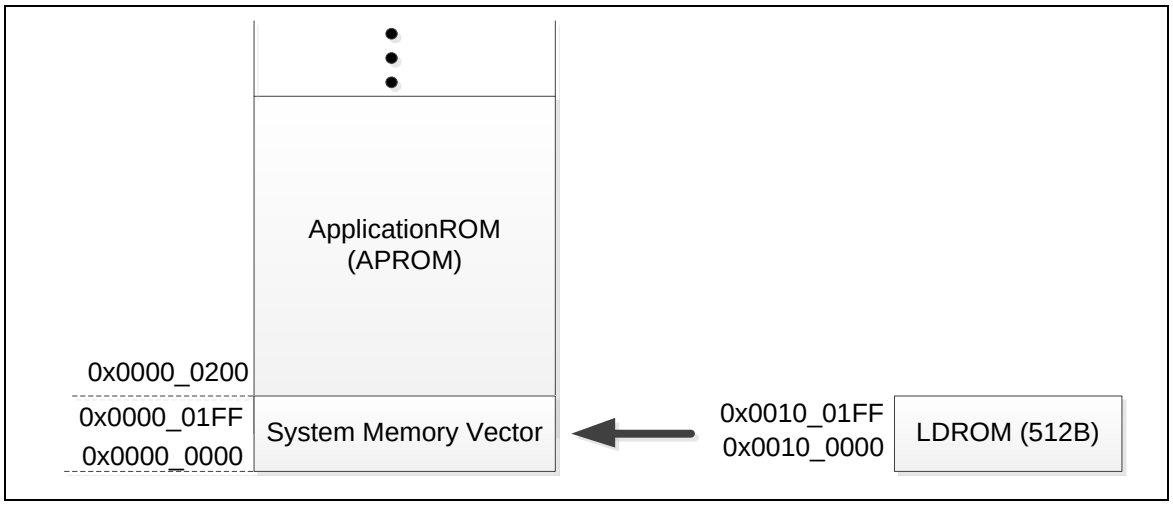


Figure 6.4-5 LDROM with IAP Mode

In APROM with IAP mode, the APROM (0x0000_0000~0x0000_01FF) is mapping to the system memory vector for Cortex®-M4 instruction or data access.

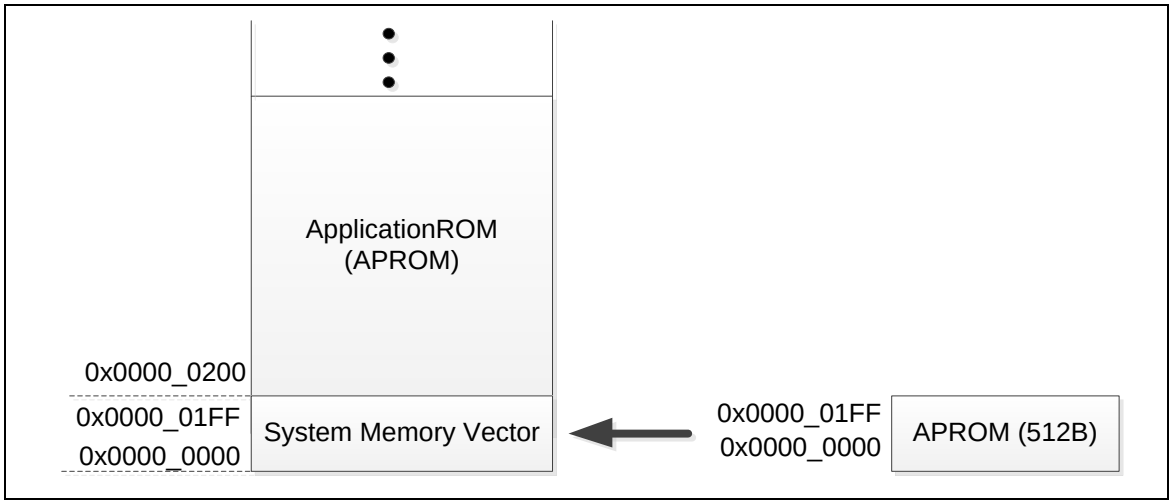


Figure 6.4-6 APROM with IAP Mode

In Boot Loader with IAP mode, the Boot Loader (0x0080_0000~0x0080_01FF) is mapping to the system memory vector for Cortex®-M4 instruction or data access.

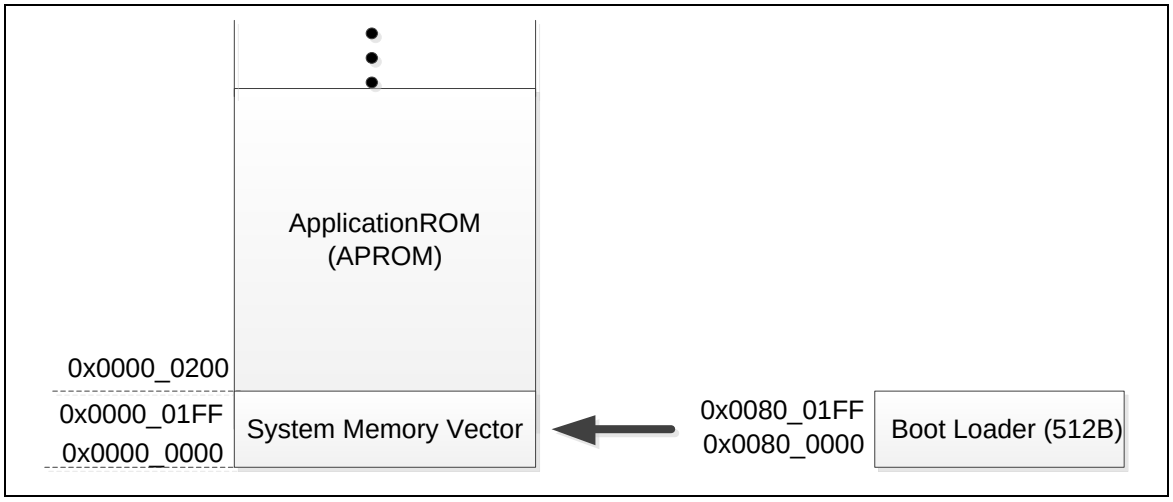


Figure 6.4-7 Boot Loader with IAP Mode

In system memory map with IAP mode, APROM, LDROM, APROM and Boot Loader can remap to the system memory vector when CPU running. User can write the target remap address to FMC_ISPADDR register and then trigger ISP procedure with the “Vector Remap” command (0x2E). In VECMAP (FMC_ISPSTS[23:9]), shows the finial system memory vector mapping address.

System Memory Map without IAP mode

In system memory map without IAP mode, CPU still can access Boot Loader(0x0080_0000~0x0080_3FFF), but the system memory vector mapping is not supported. There are two kinds of system memory map without IAP mode when chip booting: (1) LDROM without IAP, (2) APROM without IAP. In LDROM without IAP mode, LDROM base is mapping to 0x0000_0000. CPU program cannot run to access APROM. In APROM without IAP mode, APROM base is mapping to 0x0000_0000. CPU program cannot run to access LDROM. The Data Flash is shared with APROM and the Data Flash base address is defined by CONFIG1. The content of CONFIG1 is loaded into DFBA (Data Flash Base Address Register) at the flash initiation. The DFBA~(0x0001_FFFF/0x0003_FFFF) is the Data Flash region for Cortex®-M4 data access, and 0x0000_0000~(DFBA-1) is APROM region for Cortex®-M4 instruction access.

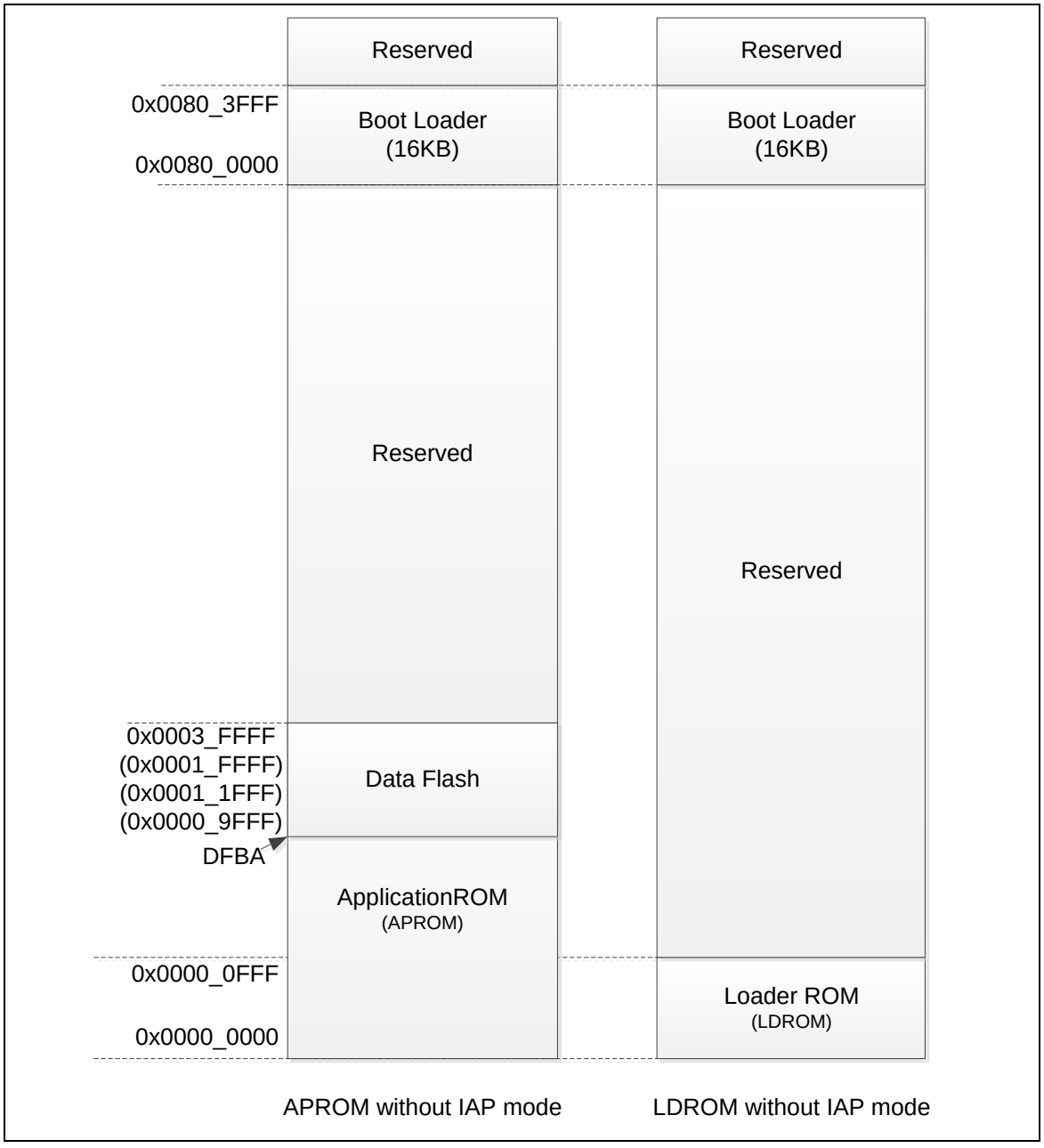


Figure 6.4-8 System Memory Map without IAP mode

6.4.4.2 Boot Selection

The NuMicro® M4TK provides five booting sources for user to select, including LDROM with IAP, LDROM without IAP, APROM with IAP, APROM without IAP, and Boot Loader with IAP. In any time, user can use both PF.6 pin and the reset pin (PF.6 is low at reset pin rising) to make CPU fetch code from Boot Load. In the PF.6 NC (No connected) case, PF.6 is internal pull high, the booting source and system memory map are setting by CBS (CONFIG0[7:6]) and MBS (CONFIG0[5]).

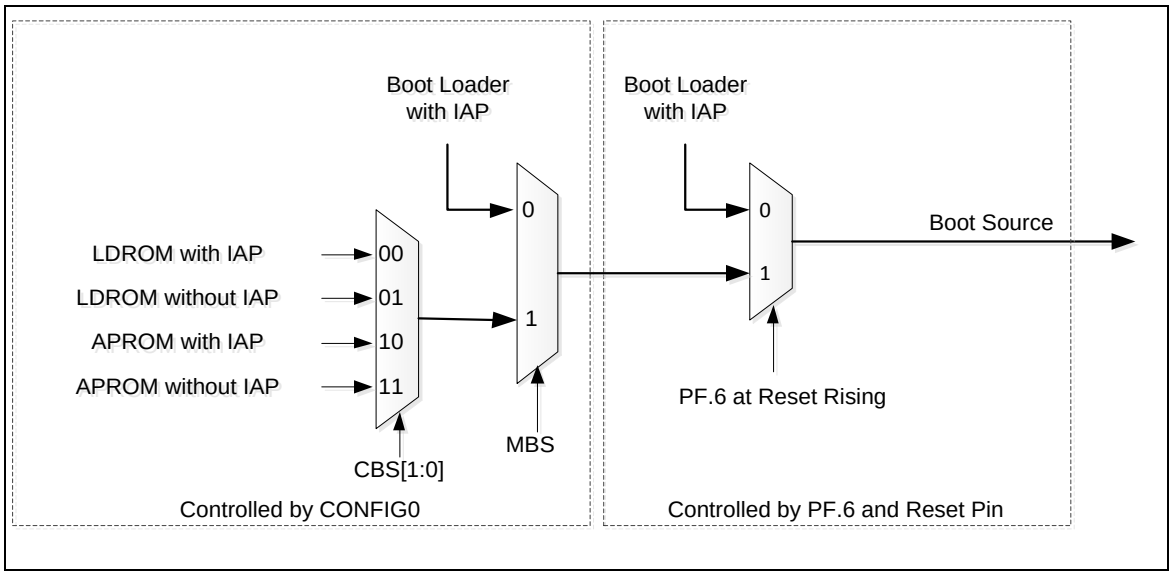


Figure 6.4-9 Boot Source Selection

MBS	CBS[1:0]	Boot Selection/System Memory Map	Vector Mapping Supporting
1	00	**LDROM with IAP**	**Yes,**
1	01	**LDROM without IAP**	**No**
1	10	**APROM with IAP**	**Yes**
1	11	**APROM without IAP**	**No**
0	XX	**Boot Loader with IAP**	**Yes**

Table 6.4-1 Vector Mapping Supporting

6.4.4.3 In-Application-Programming (IAP)

The NuMicro® M4TK Series provides In-Application-Programming (IAP) function for user to switch the code executing between APROM, LDROM and Boot Loader. User can enable the IAP function by booting chip and setting the chip boot selection bits in CBS (CONFIG0[7:6]) as 10 or 00, or in MBS (CONFIG0[5]) as 0. (if MSB(CONFIG0[5]) bit is set as 0, the CBS (FMC_ISPSTS[2:1]) registers will show 10 and ignore CONFIG0[7:6] setting.)

When chip boots with IAP function enabled, any executable code (align to 512 bytes) is allowed to map to the system memory vector any time. User can change the remap address to FMC_ISPADDR and then trigger ISP procedure with the “Vector Remap” command.

6.4.4.4 *In-System-Programming (ISP)*

NuMicro® M4TK series supports In-System-Programming (ISP) function allowing the embedded flash memory to be reprogrammed under software control. ISP is performed without removing the microcontroller from the system through the firmware and on-chip connectivity interface, such as UART, USB, I^2C, SPI, and CAN (depended on chip feature).

The NuMicro® M4TK ISP provides the following functions for embedded flash memory.

- Supports flash page erase function
- Supports flash data program function
- Supports flash data read function
- Supports company ID read function
- Supports device ID read function
- Supports unique ID read function
- Supports memory checksum calculation function
- Supports system memory vector remap function

ISP CMDs

ISP CMD	FMC_ISPCMD	FMC_ISPADDR	FMC_ISPDAT FMC_MPDAT0~FMC_MPDAT3
FLASH Page Erase	0x22	Valid address of flash memory organization. It must be 2 KB page alignment.	N/A
FLASH 32-bit Program	0x21	Valid address of flash memory organization	FMC_ISPDAT :Programming Data FMC_MPDAT0~FMC_MPDAT3 : N/A
FLASH 64-bit Program	0x61	Valid address of flash memory organization	FMC_ISPDAT :N/A FMC_MPDAT0: LSB Programming Data FMC_MPDAT1: MSB Programming Data FMC_MPDAT2~FMC_MPDAT3: N/A
FLASH Multi-Word Program	0x27	Valid address of flash memory organization	FMC_ISPDAT :N/A FMC_MPDAT0: 1'st Programming Data FMC_MPDAT1: 2'nd Programming Data FMC_MPDAT2: 3'rd Programming Data FMC_MPDAT3: 4'th Programming Data
FLASH 32-bit Read	0x00	Valid address of flash memory organization	FMC_ISPDAT: Return Data FMC_MPDAT0~FMC_MPDAT3 : N/A
FLASH 64-bit Read	0x40	Valid address of flash memory organization	FMC_ISPDAT: LSB Return Data FMC_MPDAT0: LSB Return Data FMC_MPDAT1: MSB Return Data

			FMC_MPDAT2~FMC_MPDAT3: N/A
Read Company ID	0x0B	0x0000_0000	FMC_ISPDAT: 0x0000_00DA FMC_MPDAT0~FMC_MPDAT3 : N/A
Read Checksum	0x0D	Keep address of “Run Checksum Calculation”	FMC_ISPDAT: Return Checksum FMC_MPDAT0~FMC_MPDAT3 : N/A
Run Checksum Calculation	0x2D	Valid start address of memory organization It must be 2 KB page alignment	FMC_ISPDAT: Size It must be 2 KB alignment FMC_MPDAT0~FMC_MPDAT3 : N/A
Read Unique ID	0x04	0x0000_0000	FMC_ISPDAT: Unique ID Word 0 FMC_MPDAT0~FMC_MPDAT3 : N/A
		0x0000_0004	FMC_ISPDAT: Unique ID Word 1 FMC_MPDAT0~FMC_MPDAT3 : N/A
		0x0000_0008	FMC_ISPDAT: Unique ID Word 2 FMC_MPDAT0~FMC_MPDAT3 : N/A
Vector Remap	0x2E	Valid address in APROM,LDROM or boot loader It must be 512 bytes alignment	N/A

Table 6-10 ISP Command List

ISP Procedure

The FMC controller provides embedded flash memory read, erase and program operation. Several control bits of FMC control register are write-protected, thus it is necessary to unlock before setting.

After unlocking the protected register bits, user needs to set the FMC_ISPCTL control register to decide to update LDROM, APROM or user configuration block, and then set ISPEN (FMC_ISPCTL[0]) to enable ISP function.

Once the FMC_ISPCTL register is set properly, user can set FMC_ISPCMD (refer above ISP command list) for specify operation. Set FMC_ISPADDR for target flash memory based on flash memory organization. FMC_ISPDAT can be used to set the data to program or used to return the read data according to FMC_ISPCMD.

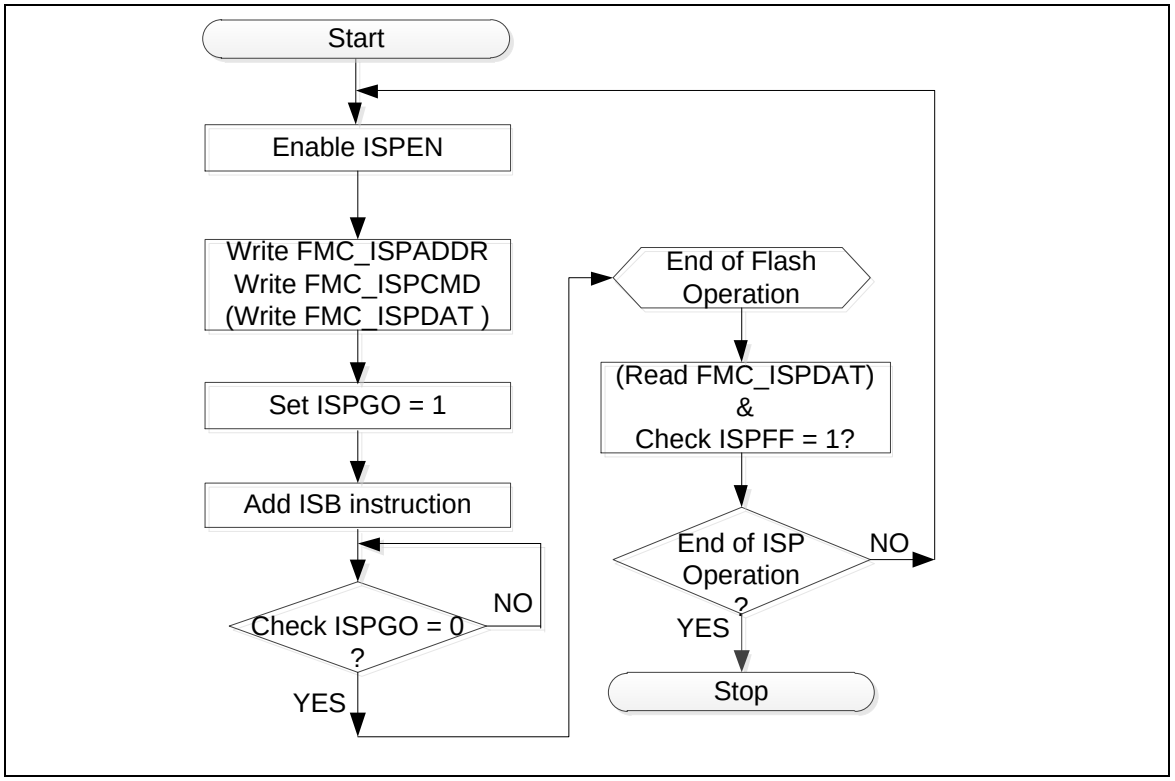


Figure 6.4-10 ISP Procedure Example

Finally, set the ISPGO (FMC_ISPTRG[0]) register to perform the relative ISP function. The ISPGO(FMC_ISPTRG[0]) bit is self-cleared when ISP function has been done. To make sure ISP function has been finished before CPU goes ahead, ISB (Instruction Synchronization Barrier) instruction is used right after ISPGO(FMC_ISPTRG[0]) setting.

Several error conditions will be checked after ISP is completed. If an error condition occurs, ISP operation is not started and the ISP fail flag will be set instead. ISPFF(FMC_ISPSTS[6]) flag can only be cleared by software. The next ISP procedure can be started even ISPFF(FMC_ISPSTS[6]) bit is kept as 1. Therefore, it is recommended to check the ISPFF(FMC_ISPSTS[6]) bit and clear it after each ISP operation if it is set to 1.

When the ISPGO(FMC_ISPTRG[0]) bit is set, CPU will wait for ISP operation to finish during this period; the peripheral still keeps working as usual. If any interrupt request occurs, CPU will not service it till ISP operation is finished. When ISP operation is finished, the ISPGO bit will be cleared by hardware automatically. User can check whether ISP operation is finished or not by the ISPGO(FMC_ISPTRG[0]) bit. User should add ISB (Instruction Synchronization Barrier) instruction next to the instruction in which ISPGO (FMC_ISPTRG[0]) bit is set 1 to ensure correct execution of the instructions following ISP operation.

6.4.4.5 Embedded Flash Memory Programming

The NuMicro® M4TK series provides 32-bit, 64-bit and multi-word flash memory programming function to speed up flash updated procedure. Table6.4-3 lists required FMC control registers in each embedded flash programming function.

Register	Description	32-Bit Programming	64-Bit Programming	Multi-Word Programming

FMC_ISPCTL	ISP Control Register	✓	✓	✓
FMC_ISPADDR	ISP Address Register	✓	✓	✓
FMC_ISPDAT	ISP Data Register	✓	N/A	N/A
FMC_ISPCMD	ISP CMD Register	0x21	0x61	0x27
FMC_ISPTRG	ISP Trigger Register	✓	✓	✓
FMC_ISPSTS	ISP Status Register	✓	✓	N/A
FMC_MPDAT0	ISP Data0 Register	N/A	✓	✓
FMC_MPDAT1	ISP Data1 Register	N/A	✓	✓
FMC_MPDAT2	ISP Data2 Register	N/A	N/A	✓
FMC_MPDAT3	ISP Data3 Register	N/A	N/A	✓
FMC_MPSTS	ISP Multi-Program status	N/A	N/A	✓
FMC_MPADDR	ISP Multi-Program Address	N/A	N/A	✓

Table 6.4-3 FMC control registers for Flash Programming

64-bit Programming

The NuMicro® M4TK series 64-bit programming function is faster than 32-bit programming. FMC_ISPDAT is used for 32-bit programming data register. In 64-bit programming, there are two programming data registers, one is FMC_MPDAT0 for LSB word, and the other is FMC_MPDAT1 for MSB word, and ISP command is 0x61, the other registers are the same as 32-bit programming. Figure 6.4-12 / Figure 6.4-13 shows ISP 32-bit / 64-bit programing procedure.

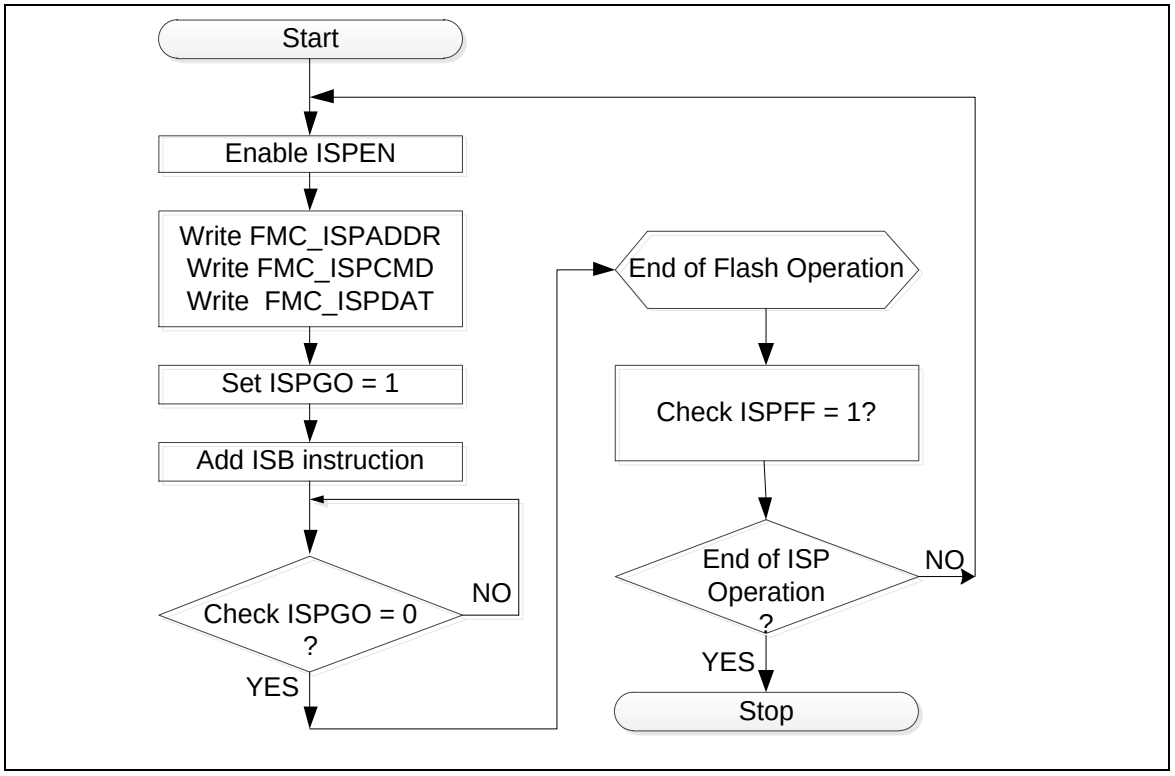


Figure 6.4-11 ISP 32-bit Programming Procedure

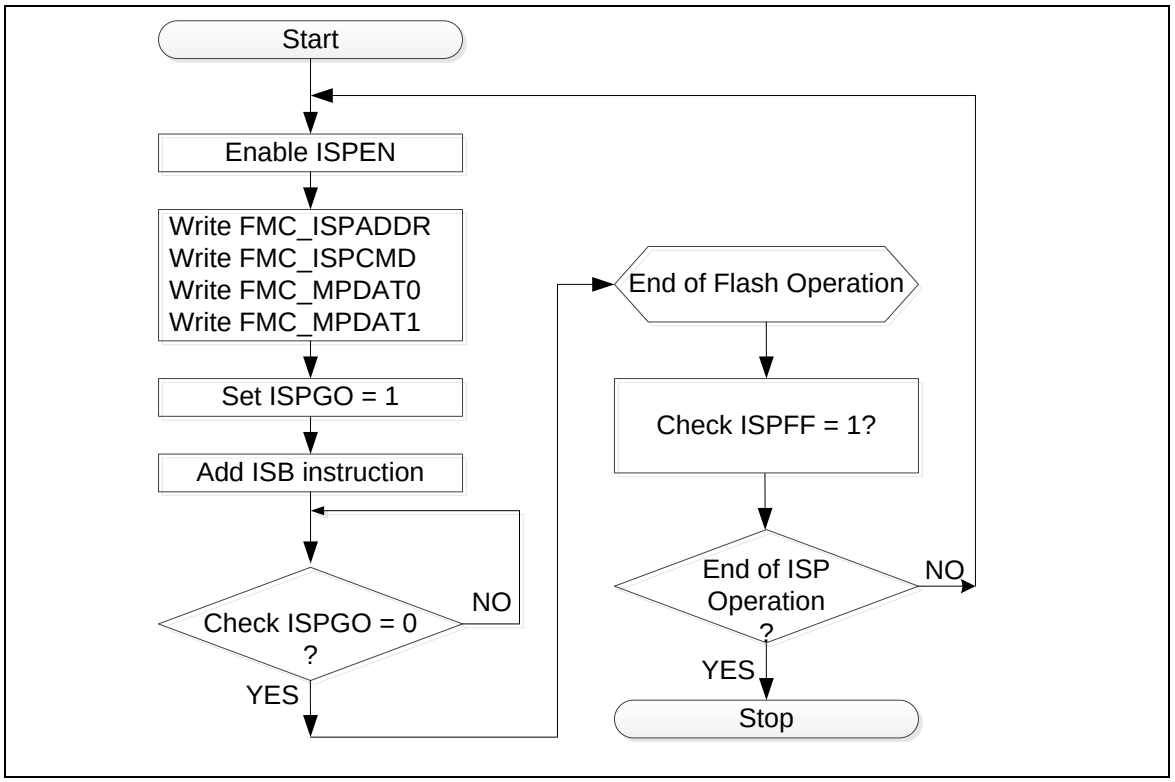


Figure 6.4-12 ISP 64-bit Programming Procedure

Multi-word Programming

The NuMicro® M4TK supports multi-word programming function to speed up flash updated procedure. The maximum programming length is up to 256 bytes, and the minimum programming length is 8 bytes (2 words). The multi-word programming is the fastest programming function if the programming words more than 8 bytes, because only one set of flash setup time and hold time needed for one time operation.

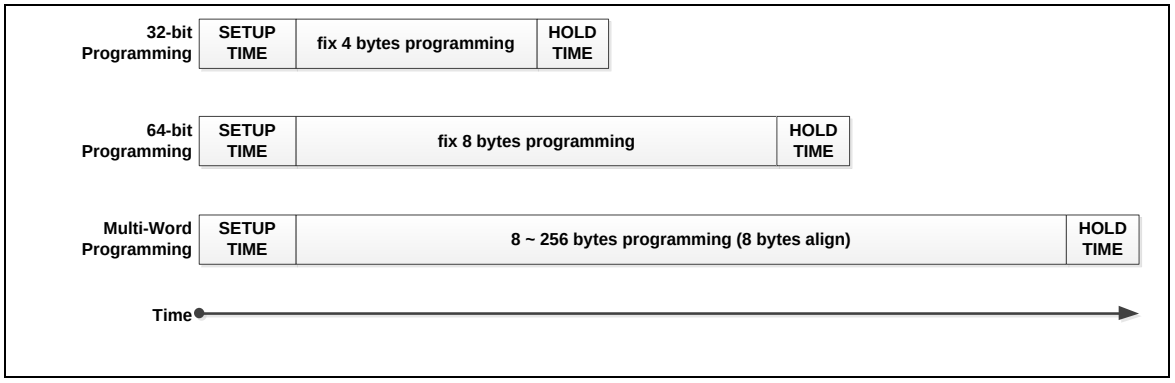


Figure 6.4-13 Multi-word Programming Time

In multi-word programming operation, Cortex®-M4 CPU has to monitor the empty status of the programming buffer. CPU has to prepare the next data for programming continuity. The multi-program firmware should not be located in APROM or LDROM, because CPU instruction fetch cannot be hold. The firmware has to be located in Boot loader or embedded SRAM of chip to avoid CPU hold.

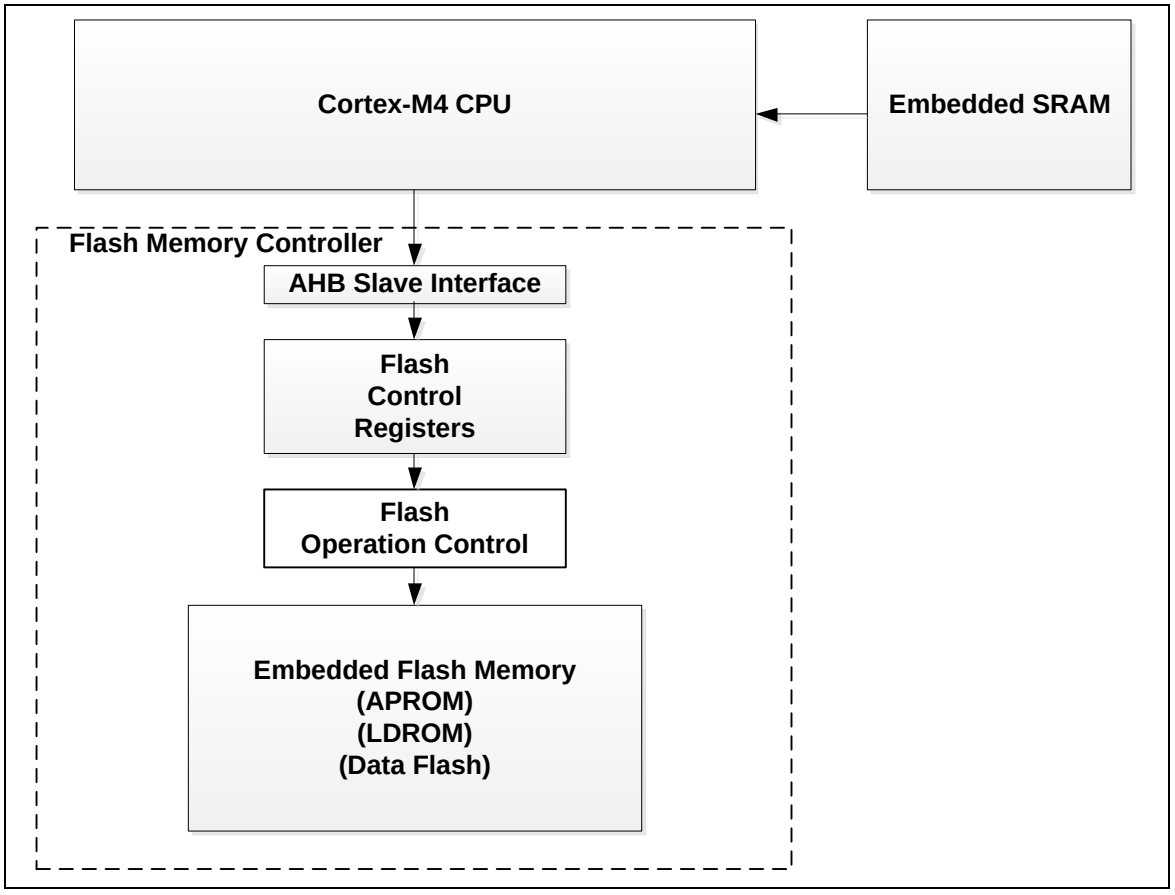


Figure 6.4-14 Firmware in SRAM for Multi-word Programming

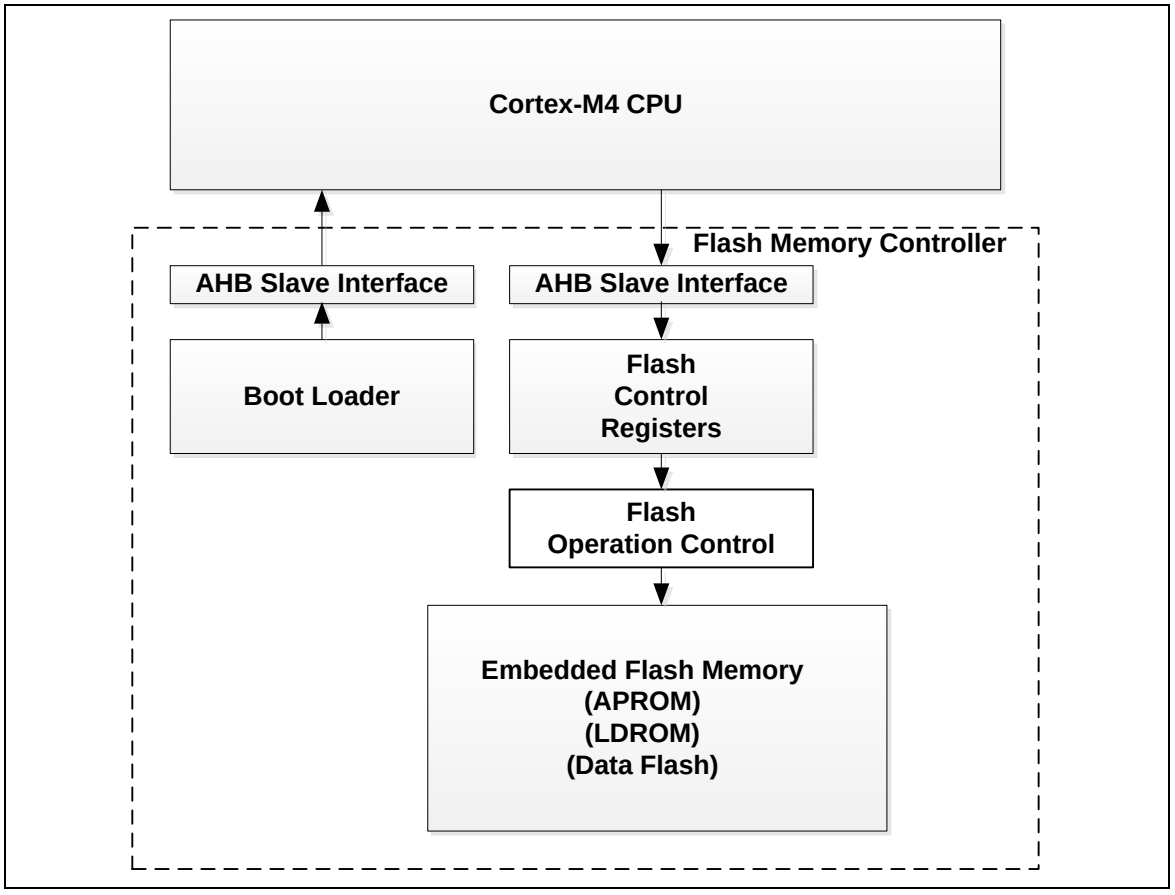


Figure 6.4-15 Firmware in Boot Loader for Multi-word Programming

The multi-word programming flow is shown below. The starting ISP address (FMC_ISPADDR) has to be 8-byte align, FMC_ISPADDR[2:0] should be 0. FMC_MPDAT0 is the data word of the offset 0x0, FMC_MPDAT1 is the second word (offset 0x4), FMC_MPDAT2 is the third word (offset 0x8), and FMC_MPDAT3 is forth word (offset 0xC). If the starting ISP address FMC_ISPADDR [3] is 0, the 1st data word should put on FMC_MPDAT0, and 2nd word is FMC_MPDAT1, 3rd word is FMC_MPDAT2, and 4th word is FMC_MPDAT3. If the starting ISP address FMC_ISPADDR [3] is 1, the 1st data word should put on FMC_MPDAT2, and 2nd word is FMC_MPDAT3, 3rd word is FMC_MPDAT0, and 4th word is FMC_MPDAT1. The maximum programming size is 256 bytes and align to 256-byte address. While FMC controller performs multi-word programming operation, CPU needs to monitor the buffer status D3~D0(FMC_MPSTS[7:4]) and MPBUSY (FMC_MPSTS[0]) to wait the buffer empty ((D1,D0)=00, or (D3,D2)=00), and then CPU needs to update the next programming data (FMC_MPDAT0, FMC_MPDAT1, FMC_MPDAT2 and FMC_MPDAT3) in time. Otherwise, FMC controller will exit multi-word programming operation (MPBUSY (FMC_MPSTS[0]) = 0). If CPU cannot update the data in time (MPBUSY (FMC_MPSTS[0]) =0), CPU needs restart a new multi-word programming procedure to continue, FMC_MPADDR provides the last program address information. At the end of operation, CPU has to check ISPFF (FMC_MPSTS[2]) to confirm the multi-word operation successful complete.

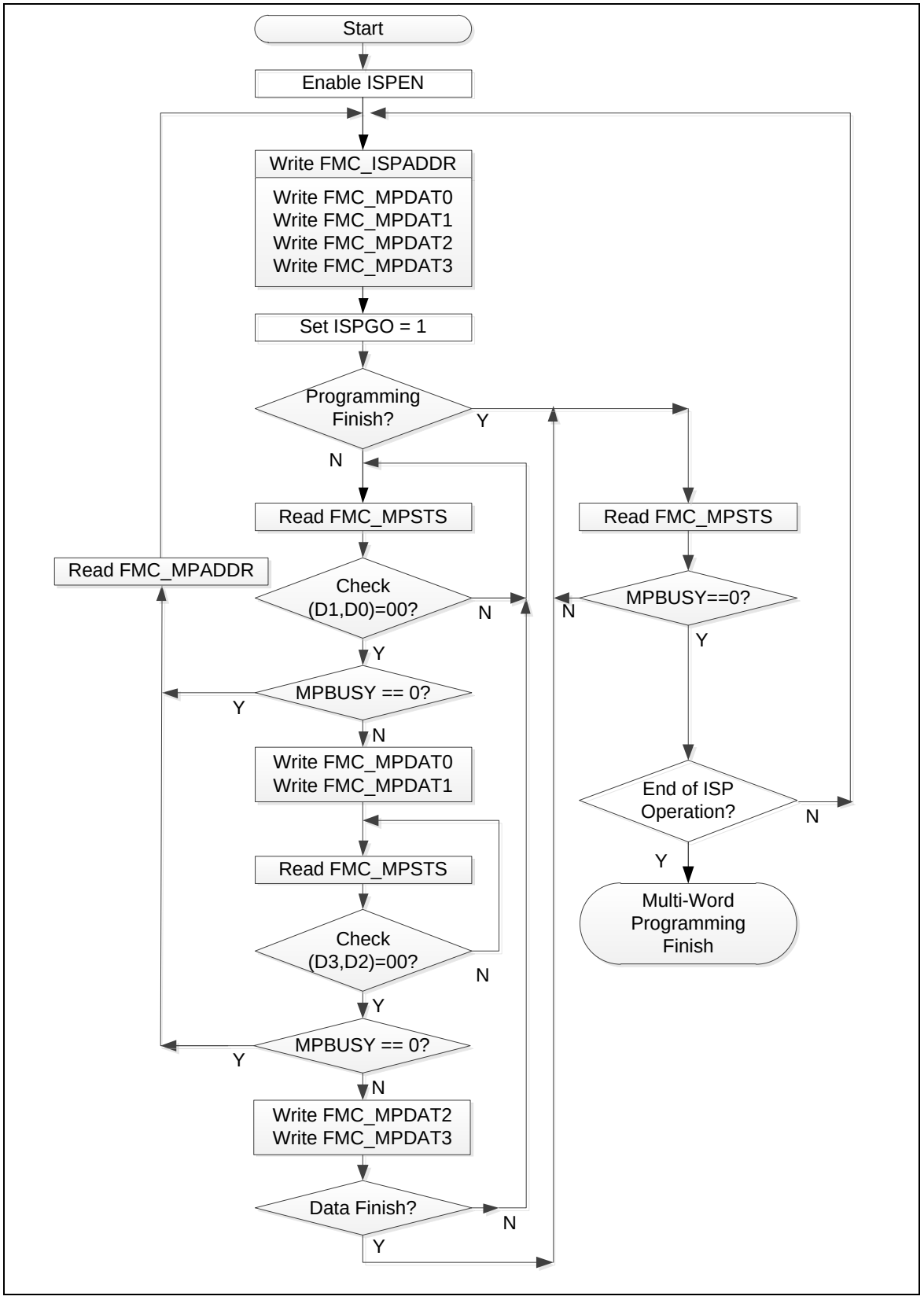


Figure 6.4-16 Multi-word Programming Flow

6.4.4.6 *Fast Flash Programming Verification*

In traditional flash programming operation, the controller receives the programming trigger event then control the timing to perform the programming embedded flash memory. as show in Figure 6.4-18.

The NuMicro® M4TK supports the fast flash programming verification function, which provides hardware verification for flash programming to save time of the CPU read back and comparison. When data is programmed to the embedded flash memory, the controller asserts the flash read operation to read data out, and performs data comparison with data in. Finally, the comparison result is saved in PGFF (FMC_ISPSTS[5]). The PGFF is set to 1 if output data is not the same as the input programming data. The flag is kept until clear by software or a new erase operation.

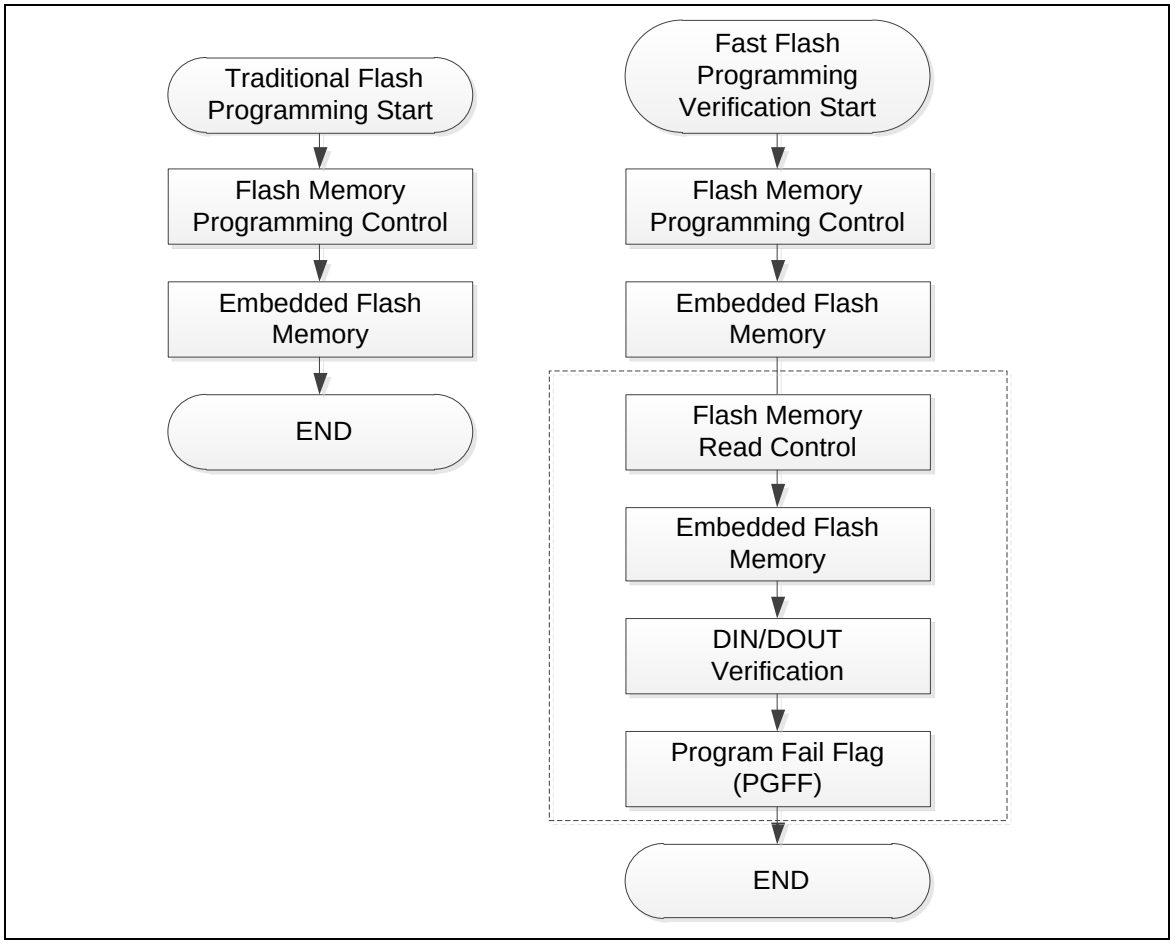


Figure 6.4-17 Fast Flash Programming Verification Flow

In traditional flash updated operation, the flash memory has to perform three steps to complete the flash memory updated procedure, (1) Flash ERASE (2) Flash PROGRAM (3) Flash READ back all of data to check the correction. In NuMicro® M4TK series, it is only read FMC_ISPSTS to check PGFF flag in step (3) without reading data back to confirm.

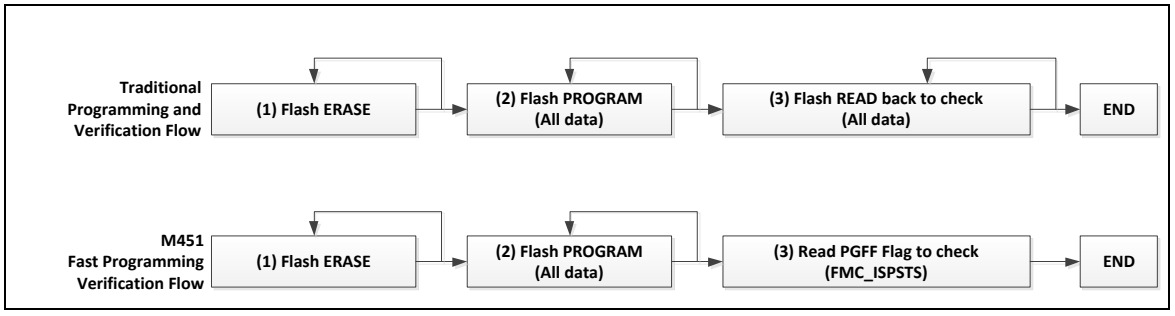


Figure 6.4-18 Verification Flow

The fast flash programming verification function is released for 32-bit programming and 64-bit programming operation, but multi-word programming operation is not suitable due to the embedded flash HV (High Voltage) of continue programming.

6.4.4.7 Checksum Calculation

The NuMicro® M4TK series supports the checksum calculation function to help user quickly check the memory content includes APROM, LDROM and Boot Loader. Figure 6.4-20 shows the checksum calculation. S_1 ~S_K are the KB data for checksum calculation the checksum (32-bit) is the summary of unsigned bytes.

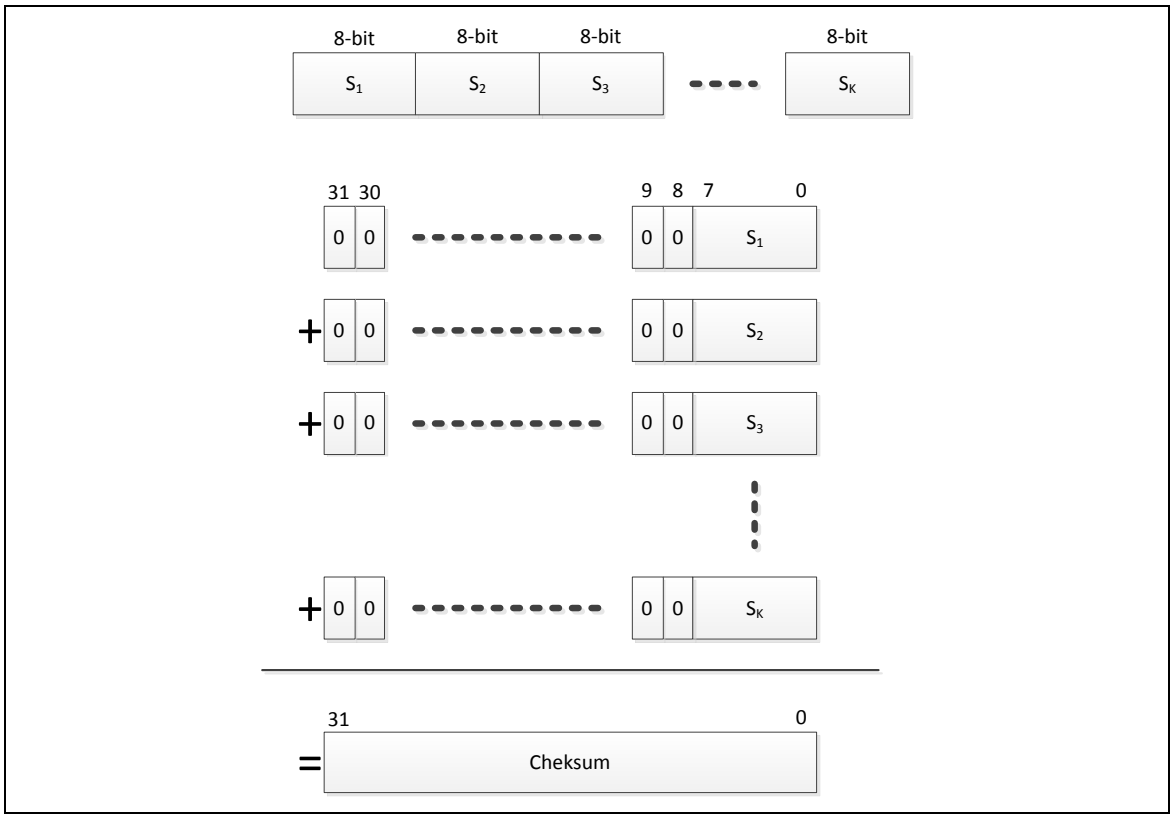


Figure 6.4-19 Checksum for KB Calculation

Three steps complete this checksum calculation.

Step 1: perform ISP “Run Memory Checksum” operation

Step 2: perform ISP “Read Memory Checksum” operation

Step 3: read FMC_ISPDAT to get checksum.

In step 1, user has to set the memory starting address (FMC_ISPADDR) and size (FMC_ISPDAT) to calculate. Both address and size have to be 2 KB alignment, the size should be ≥ 2 KB and the starting address includes APROM, LDROM and Boot Loader.

In step 2, the FMC_ISPADDR should be kept as the same as step 1.

In step 3, the checksum is read from FMC_ISPDAT. If the checksum is 0x0000_0000, there is one of three conditions (1) Checksum calculation is in-progress, (2) Address and size is over device limitation (3) All memory data are zero.

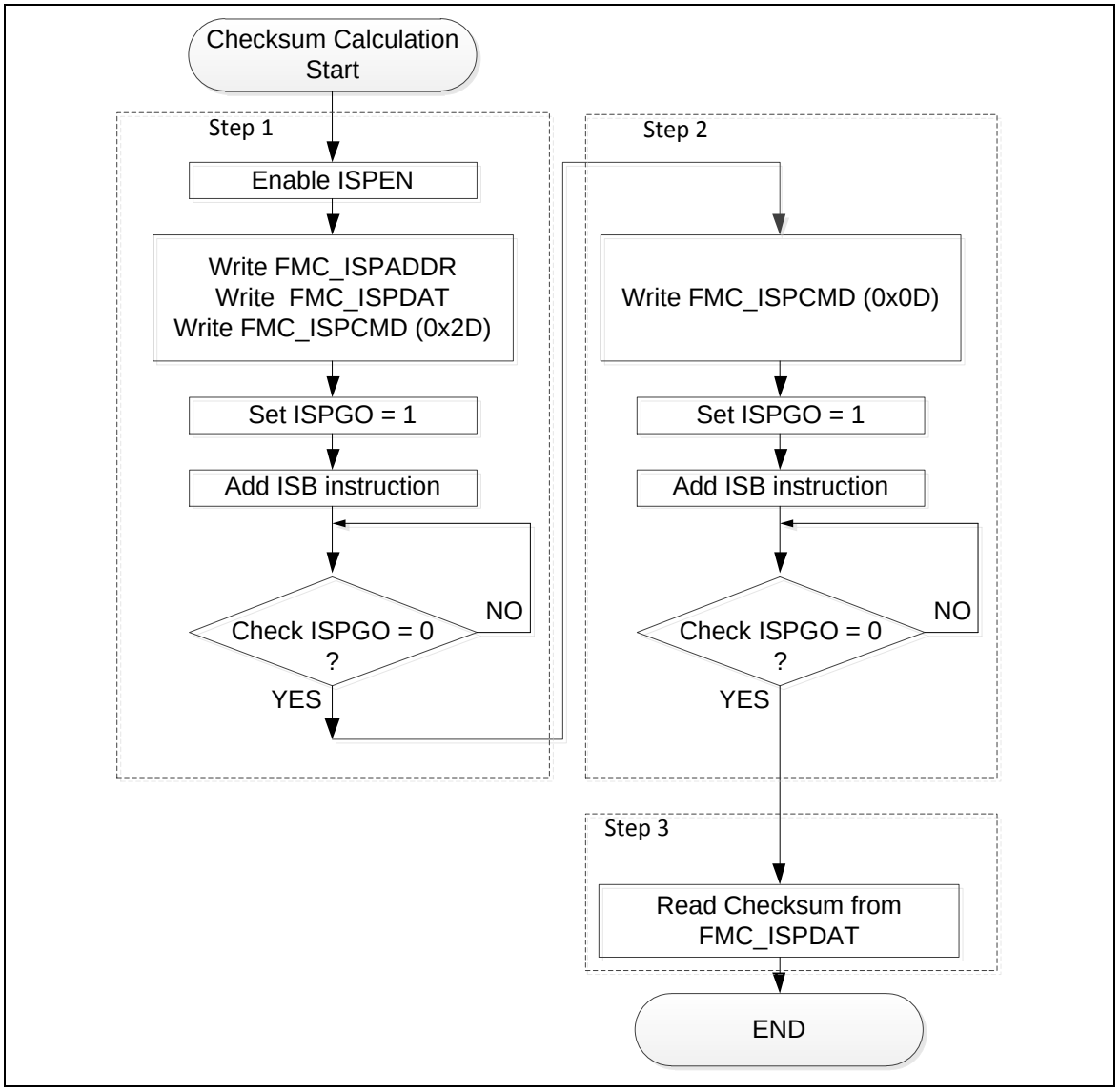


Figure 6.4-20 Checksum Calculation Flow

6.4.5 Register Map

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

Register	Offset	R/W	Description	Reset Value
FMC Base Address **FMC_BA = 0x4000_C000**				
FMC_ISPCTL	FMC_BA+0x00	R/W	ISP Control Register	0x0000_0000
FMC_ISPADDR	FMC_BA+0x04	R/W	ISP Address Register	0x0000_0000
FMC_ISPDAT	FMC_BA+0x08	R/W	ISP Data Register	0x0000_0000
FMC_ISPCMD	FMC_BA+0x0C	R/W	ISP CMD Register	0x0000_0000
FMC_ISPTRG	FMC_BA+0x10	R/W	ISP Trigger Control Register	0x0000_0000
FMC_DFBA	FMC_BA+0x14	R	Data Flash Base Address	0xXXXX_XXXX
FMC_FTCTL	FMC_BA+0x18	R/W	Flash Access Time Control Register	0x0000_0000
FMC_ISPSTS	FMC_BA+0x40	R/W	ISP Status Register	0x0000_0000
FMC_MPDAT0	FMC_BA+0x80	R/W	ISP Data0 Register	0x0000_0000
FMC_MPDAT1	FMC_BA+0x84	R/W	ISP Data1 Register	0x0000_0000
FMC_MPDAT2	FMC_BA+0x88	R/W	ISP Data2 Register	0x0000_0000
FMC_MPDAT3	FMC_BA+0x8C	R/W	ISP Data3 Register	0x0000_0000
FMC_MPSTS	FMC_BA+0xC0	R	ISP Multi-Program Status Register	0x0000_0000
FMC_MPADDR	FMC_BA+0xC4	R	ISP Multi-Program Address Register	0x0000_0000

6.4.6 Register Description

ISP Control Register (FMC_ISPCTL)

Register	Offset	R/W	Description	Reset Value
FMC_ISPCTL	FMC_BA+0x00	R/W	ISP Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							**BL**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**ISPFF**	**LDUEN**	**CFGUEN**	**APUEN**	**Reserved**	**BS**	**ISPEN**

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**BL**	**Boot Loader Booting (Write Protect)** This bit is initiated with the inversed value of MBS (CONFIG0[5]). Any reset, except CPU reset (CPU is 1) or system reset (SYS), BL will be reloaded. This bit is used to check chip boot from Boot Loader or not. User should keep original value of this bit when updating FMC_ISPCTL register. 0 = Booting from APROM or LDROM. 1 = Booting from Boot Loader. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[15]	**Reserved**	Reserved.
[14:12]	**Reserved**	Reserved.
[11]	**Reserved**	Reserved.
[10:8]	**Reserved**	Reserved.
[7]	**Reserved**	Reserved.
[6]	**ISPFF**	**ISP Fail Flag (Write Protect)** This bit is set by hardware when a triggered ISP meets any of the following conditions: This bit needs to be cleared by writing 1 to it. (1) APROM writes to itself if APUEN is set to 0. (2) LDROM writes to itself if LDUEN is set to 0. (3) CONFIG is erased/programmed if CFGUEN is set to 0. (4) Page Erase command at LOCK mode with ICE connection (5) Erase or Program command at brown-out detected (6) Destination address is illegal, such as over an available range. (7) Invalid ISP commands **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

[5]	**LDUEN**	**LDROM Update Enable Bit (Write Protect)** LDROM update enable bit. 0 = LDROM cannot be updated. 1 = LDROM can be updated. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[4]	**CFGUEN**	**CONFIG Update Enable Bit (Write Protect)** 0 = CONFIG cannot be updated. 1 = CONFIG can be updated. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[3]	**APUEN**	**APROM Update Enable Bit (Write Protect)** 0 = APROM cannot be updated when the chip runs in APROM. 1 = APROM can be updated when the chip runs in APROM. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[2]	**Reserved**	Reserved.
[1]	**BS**	**Boot Select (Write Protect)** When MBS in CONFIG0 is 1, set/clear this bit to select next booting from LDROM/APROM, respectively. This bit also functions as chip booting status flag, which can be used to check where chip booted from. This bit is initiated with the inversed value of CBS[1] (CONFIG0[7]) after any reset is happened except CPU reset (CPU is 1) or system reset (SYS) is happened 0 = Booting from APROM when MBS (CONFIG0[5]) is 1. 1 = Booting from LDROM when MBS (CONFIG0[5]) is 1. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[0]	**ISPEN**	**ISP Enable Bit (Write Protect)** ISP function enable bit. Set this bit to enable ISP function. 0 = ISP function Disabled. 1 = ISP function Enabled. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

ISP Address (FMC_ISPADDR)

Register	Offset	R/W	Description	Reset Value
FMC_ISPADDR	FMC_BA+0x04	R/W	ISP Address Register	0x0000_0000

31	30	29	28	27	26	25	24
ISPADDR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ISPADDR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ISPADDR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ISPADDR							

Bits	Description	
[31:0]	**ISPADDR**	**ISP Address** The NuMicro® M4TK series is equipped with embedded flash. ISPADDR[1:0] must be kept 00 for ISP 32-bit operation. ISPADDR[2:0] must be kept 000 for ISP 64-bit operation. For Checksum Calculation command, this field is the flash starting address for checksum calculation, 2 KB alignment is necessary for checksum calculation.

ISP Data Register (FMC_ISPDAT)

Register	Offset	R/W	Description	Reset Value
FMC_ISPDAT	FMC_BA+0x08	R/W	ISP Data Register	0x0000_0000

31	30	29	28	27	26	25	24
ISPDAT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ISPDAT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ISPDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ISPDAT							

Bits	Description	
[31:0]	**ISPDAT**	**ISP Data** Write data to this register before ISP program operation. Read data from this register after ISP read operation. For Run Checksum Calculation command, ISPDAT is the memory size (byte) and 2 KB alignment. For ISP Read Checksum command, ISPDAT is the checksum result. If ISPDAT = 0x0000_0000, it means that (1) the checksum calculation is in progress, (2) the memory range for checksum calculation is incorrect, or (3) all of data are 0.

ISP CMD (FMC_ISPCMD)

Register	Offset	R/W	Description	Reset Value
FMC_ISPCMD	FMC_BA+0x0C	R/W	ISP CMD Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	CMD						

Bits	Description	
[31:7]	Reserved	Reserved.
[6:0]	CMD	**ISP CMD** ISP command table is shown below: 0x00= FLASH 32-bit Read. 0x40= FLASH 64-bit Read. 0x04= Read Unique ID. 0x0B= Read Company ID. 0x0D= Read Checksum. 0x21= FLASH 32-bit Program. 0x22= FLASH Page Erase. 0x27= FLASH Multi-Word Program. 0x2D= Run Checksum Calculation. 0x2E= Vector Remap. 0x61= FLASH 64-bit Program. The other commands are invalid.

ISP Trigger Control Register (FMC_ISPTRG)

Register	Offset	R/W	Description	Reset Value
FMC_ISPTRG	FMC_BA+0x10	R/W	ISP Trigger Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**ISPGO**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**ISPGO**	**ISP Start Trigger (Write Protect)** Write 1 to start ISP operation and this bit will be cleared to 0 by hardware automatically when ISP operation is finished. 0 = ISP operation is finished. 1 = ISP is progressed. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.

Data Flash Base Address Register (FMC_DFBA)

Register	Offset	R/W	Description	Reset Value
FMC_DFBA	FMC_BA+0x14	R	Data Flash Base Address	0xXXXX_XXXX

31	30	29	28	27	26	25	24
DFBA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DFBA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DFBA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DFBA							

Bits	Description	
[31:0]	**DFBA**	**Data Flash Base Address** This register indicates Data Flash start address. It is a read only register. The Data Flash is shared with APROM. the content of this register is loaded from CONFIG1 This register is valid when DFEN (CONFIG0[0]) =0 .

Flash Access Time Control Register (FMC_FTCTL)

Register	Offset	R/W	Description	Reset Value
FMC_FTCTL	FMC_BA+0x18	R/W	Flash Access Time Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**FOM**			**Reserved**			

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6:4]	**FOM**	**Frequency Optimization Mode (Write Protect)** The NuMicro® M4TK series support adjustable flash access timing to optimize the flash access cycles in different working frequency. 001 = Frequency ≤ 12MHz. 010 = Frequency ≤ 36MHz. 100 = Frequency ≤ 60MHz. Others = Frequency ≤ 72MHz. **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[3:0]	**Reserved**	Reserved.

ISP Status Register (FMC_ISPSTS)

Register	Offset	R/W	Description	Reset Value
FMC_ISPSTS	FMC_BA+0x40	R/W	ISP Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
VECMAP							
15	14	13	12	11	10	9	8
VECMAP							Reserved
7	6	5	4	3	2	1	0
Reserved	ISPFF	PGFF	Reserved	MBS	CBS		ISPBUSY

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:9]	**VECMAP**	**Vector Page Mapping Address (Read Only)** All access to 0x0000_0000~0x0000_01FF is remapped to the flash memory address {VECMAP[14:0], 9'h000} ~ {VECMAP[14:0], 9'h1FF}
[8:7]	**Reserved**	Reserved.
[6]	**ISPFF**	**ISP Fail Flag (Write Protect)** This bit is the mirror of ISPFF (FMC_ISPCTL[6]), it needs to be cleared by writing 1 to FMC_ISPCTL[6] or FMC_ISPSTS[6]. This bit is set by hardware when a triggered ISP meets any of the following conditions: (1) APROM writes to itself if APUEN is set to 0. (2) LDROM writes to itself if LDUEN is set to 0. (3) CONFIG is erased/programmed if CFGUEN is set to 0. (4) Page Erase command at LOCK mode with ICE connection (5) Erase or Program command at brown-out detected (6) Destination address is illegal, such as over an available range. (7) Invalid ISP commands **Note:** This bit is write protected. Refer to the SYS_REGLCTL register.
[5]	**PGFF**	**Flash Program with Fast Verification Flag (Read Only)** This bit is set if data is mismatched at ISP programming verification. This bit is clear by performing ISP flash erase or ISP read CID operation 0 = Flash Program is success. 1 = Flash Program is fail. Program data is different with data in the flash memory
[4]	**Reserved**	Reserved.

[3]	**MBS**	**Boot From Boot Loader Selection Flag (Read Only)** This bit is initiated with the MBS (CONFIG0[5]) after any reset is happened except CPU reset (CPU is 1) or system reset (SYS) is happened 0 = Booting from Boot Loader. 1 = Booting from LDROM/APROM.(.see CBS bit setting)
[2:1]	**CBS**	**Boot Selection of CONFIG (Read Only)** This bit is initiated with the CBS (CONFIG0[7:6]) after any reset is happened except CPU reset (CPU is 1) or system reset (SYS) is happened. The following function is valid when MBS (FMC_ISPSTS[3])= 1. 00 = LDROM with IAP mode. 01 = LDROM without IAP mode. 10 = APROM with IAP mode. 11 = APROM without IAP mode.
[0]	**ISPBUSY**	**ISP Busy Flag (Read Only)** Write 1 to start ISP operation and this bit will be cleared to 0 by hardware automatically when ISP operation is finished. This bit is the mirror of ISPGO(FMC_ISPTRG[0]). 0 = ISP operation is finished. 1 = ISP is progressed.

ISP Data 0 Register (FMC_MPDAT0)

Register	Offset	R/W	Description	Reset Value
FMC_MPDAT0	FMC_BA+0x80	R/W	ISP Data0 Register	0x0000_0000

31	30	29	28	27	26	25	24
ISPDAT0							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ISPDAT0							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ISPDAT0							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ISPDAT0							

Bits	Description	
[31:0]	**ISPDAT0**	**ISP Data 0** This register is the first 32-bit data for 32-bit/64-bit/multi-word programming, and it is also the mirror of FMC_ISPDAT, both registers keep the same data.

ISP Data 1 Register (FMC_MPDAT1)

Register	Offset	R/W	Description	Reset Value
FMC_MPDAT1	FMC_BA+0x84	R/W	ISP Data1 Register	0x0000_0000

31	30	29	28	27	26	25	24
ISPDAT1							
23	22	21	20	19	18	17	16
ISPDAT1							
15	14	13	12	11	10	9	8
ISPDAT1							
7	6	5	4	3	2	1	0
ISPDAT1							

Bits	Description	
[31:0]	ISPDAT1	**ISP Data 1** This register is the second 32-bit data for 64-bit/multi-word programming.

ISP Data 2 Register (FMC_MPDAT2)

Register	Offset	R/W	Description	Reset Value
FMC_MPDAT2	FMC_BA+0x88	R/W	ISP Data2 Register	0x0000_0000

31	30	29	28	27	26	25	24
ISPDAT2							
23	22	21	20	19	18	17	16
ISPDAT2							
15	14	13	12	11	10	9	8
ISPDAT2							
7	6	5	4	3	2	1	0
ISPDAT2							

Bits	Description	
[31:0]	ISPDAT2	**ISP Data 2** This register is the third 32-bit data for multi-word programming.

ISP Data 3 Register (FMC_MPDAT3)

Register	Offset	R/W	Description	Reset Value
FMC_MPDAT3	FMC_BA+0x8C	R/W	ISP Data3 Register	0x0000_0000

31	30	29	28	27	26	25	24
ISPDAT3							
23	22	21	20	19	18	17	16
ISPDAT3							
15	14	13	12	11	10	9	8
ISPDAT3							
7	6	5	4	3	2	1	0
ISPDAT3							

Bits	Description	
[31:0]	ISPDAT3	**ISP Data 3** This register is the fourth 32-bit data for multi-word programming.

ISP Multi-Program Status Register (FMC_MPSTS)

Register	Offset	R/W	Description	Reset Value
FMC_MPSTS	FMC_BA+0xC0	R	ISP Multi-Program Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
D3	D2	D1	D0	Reserved	ISPFF	PPGO	MPBUSY

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**D3**	**ISP DATA 3 Flag (Read Only)** This bit is set when FMC_MPDAT3 is written and auto-clear to 0 when the FMC_MPDAT3 data is programmed to flash complete. 0 = FMC_MPDAT3 register is empty, or program to flash complete. 1 = FMC_MPDAT3 register has been written, and not program to flash complete.
[6]	**D2**	**ISP DATA 2 Flag (Read Only)** This bit is set when FMC_MPDAT2 is written and auto-clear to 0 when the FMC_MPDAT2 data is programmed to flash complete. 0 = FMC_MPDAT2 register is empty, or program to flash complete. 1 = FMC_MPDAT2 register has been written, and not program to flash complete.
[5]	**D1**	**ISP DATA 1 Flag (Read Only)** This bit is set when FMC_MPDAT1 is written and auto-clear to 0 when the FMC_MPDAT1 data is programmed to flash complete. 0 = FMC_MPDAT1 register is empty, or program to flash complete. 1 = FMC_MPDAT1 register has been written, and not program to flash complete.
[4]	**D0**	**ISP DATA 0 Flag (Read Only)** This bit is set when FMC_MPDAT0 is written and auto-clear to 0 when the FMC_MPDAT0 data is programmed to flash complete. 0 = FMC_MPDAT0 register is empty, or program to flash complete. 1 = FMC_MPDAT0 register has been written, and not program to flash complete.
[3]	**Reserved**	Reserved.

[2]	**ISPFF**	**ISP Fail Flag (Read Only)** This bit is the mirror of ISPFF (FMC_ISPCTL[6]), it needs to be cleared by writing 1 to FMC_ISPCTL[6] or FMC_ISPSTS[6]. This bit is set by hardware when a triggered ISP meets any of the following conditions: (1) APROM writes to itself if APUEN is set to 0. (2) LDROM writes to itself if LDUEN is set to 0. (3) CONFIG is erased/programmed if CFGUEN is set to 0. (4) Page Erase command at LOCK mode with ICE connection (5) Erase or Program command at brown-out detected (6) Destination address is illegal, such as over an available range. (7) Invalid ISP commands
[1]	**PPGO**	**ISP Multi-program Status (Read Only)** 0 = ISP multi-word program operation is not active. 1 = ISP multi-word program operation is in progress.
[0]	**MPBUSY**	**ISP Multi-word Program Busy Flag (Read Only)** Write 1 to start ISP Multi-Word program operation and this bit will be cleared to 0 by hardware automatically when ISP Multi-Word program operation is finished. This bit is the mirror of ISPGO(FMC_ISPTRG[0]). 0 = ISP Multi-Word program operation is finished. 1 = ISP Multi-Word program operation is progressed.

ISP Multi-Word Program Address Register (FMC_MPADDR)

Register	Offset	R/W	Description	Reset Value
FMC_MPADDR	FMC_BA+0xC4	R	ISP Multi-Program Address Register	0x0000_0000

31	30	29	28	27	26	25	24
MPADDR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
MPADDR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
MPADDR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MPADDR							

Bits	Description	
[31:0]	**MPADDR**	**ISP Multi-word Program Address** MPADDR is the address of ISP multi-word program operation when ISPGO flag is 1. MPADDR will keep the final ISP address when ISP multi-word program is complete.

6.5 External Bus Interface (EBI)

6.5.1 Overview

The NuMicro® M4TK is equipped with an external bus interface (EBI) for external device used. To save the connections between external device and the NuMicro® M4TK, EBI operating at address bus and data bus multiplex mode. The EBI supports two chip selects that can connect two external devices with different timing setting requirement.

6.5.2 Features

The External Bus Interface (EBI) has the following functions:

- Supports address bus and data bus multiplex mode to save the address pins
- Supports two chip selects with polarity control
- Supports external devices with max. 1 MB size for each chip select
- Supports variable external bus base clock (MCLK) which based on HCLK
- Supports 8-bit or 16-bit data width for each chip select
- Supports variable address latch enable time (tALE)
- Supports variable data access time (tACC) and data access hold time (tAHD) for each chip select
- Supports configurable idle cycle for different access condition: Idle of Write command finish (W2X) and Idle of Read-to-Read (R2R)

6.5.3 Block Diagram

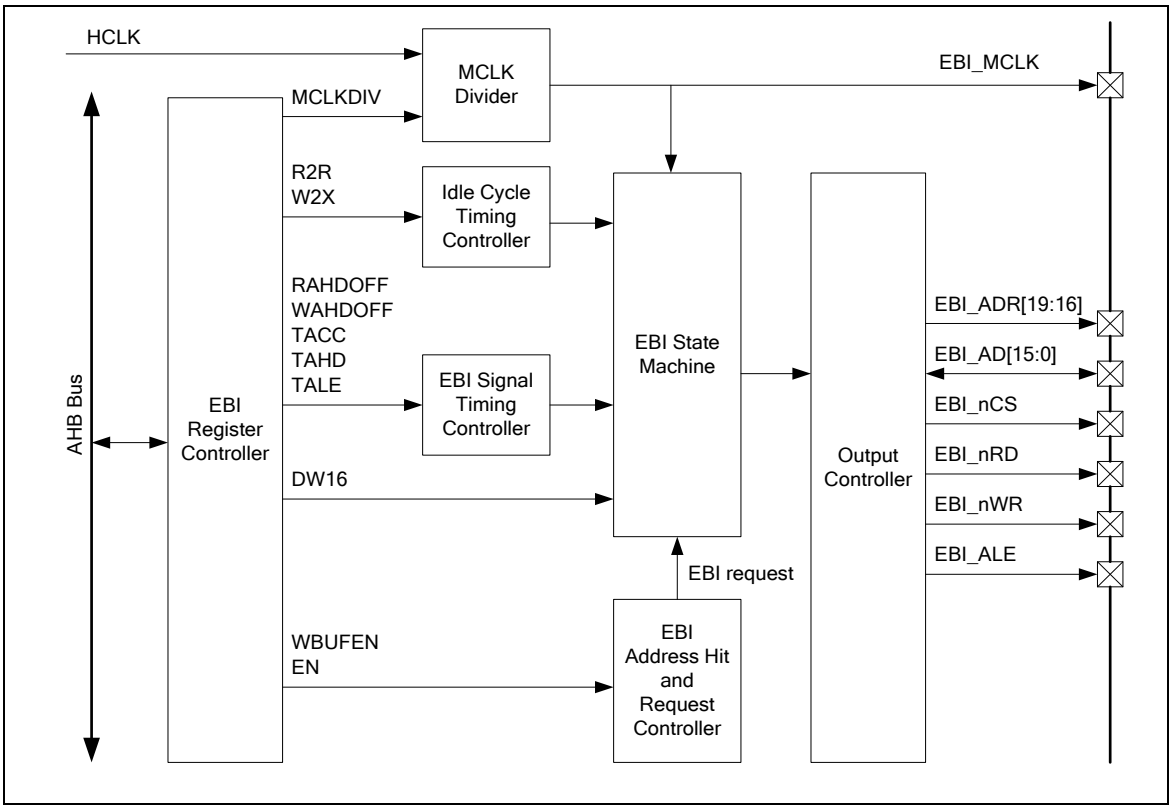


Figure 6.5-1 EBI Block Diagram

6.5.4 Basic Configuration

The EBI Controller function pins are configured in SYS_GPA_MFPL, SYS_GPB_MFPL, SYS_GPB_MFPH, SYS_GPC_MFPL, SYS_GPD_MFPL, SYS_GPD_MFPH and SYS_GPE_MFPL Multiple Function Registers.

The EBI Controller clock are enabled in EBICKEN (CLK_AHBCLK[3]).

6.5.5 Functional Description

6.5.5.1 EBI Area and Address Hit

The EBI mapping address is located at 0x6000_0000 ~ 0x601F_FFFF and the total memory space is 2 MB. When system request address hit EBI's memory space, the corresponding EBI chip select signal is assert and EBI state machine operates.

Chip Select	Address Mapping
EBI_nCS0	0x6000_0000 ~ 0x600F_FFFF
EBI_nCS1	0x6010_0000 ~ 0x601F_FFFF

To map the whole EBI memory space, it requires 20-bit address for 8-bit data width device and 19-bit address for 16-bit data width device. For package that output less than 20-bit address, EBI will map device to mirror space. For example, the package with 18-bit EBI address, EBI will

mapped external device (for Bank0/EBI_nCS0) to 0x6000_0000 ~ 0x6003_FFFF, 0x6004_0000 ~ 0x6007_FFFF, 0x6008_0000 ~ 0x600B_FFFF and 0x600C_0000 ~ 0x600F_FFFF simultaneously.

6.5.5.2 EBI Data Width Connection - Address Bus and Data Bus Multiplex Mode

The EBI supports device whose address bus and data bus are multiplexed. For the external device with separated address and data bus, the connection to device needs additional latch device to latch the address. In this case, pin EBI_ALE is connected to the latch device to latch the address value. Pin EBI_AD is the input of the latch device, and the output of the latch device is connected to the address of external device.

For 16-bit device, the EBI_AD [15:0] shared by address and 16-bit data, EBI_ADR [18:16] is dedicated for address and could be connected to 16-bit device directly. The EBI_ADR[19] will be ignore when EBI data width is set as 16-bit width. For 8-bit device, only EBI_AD [7:0] shared by address and 8-bit data, EBI_AD[15:8] and EBI_ADR[19:16] is dedicated for address and could be connected to 8-bit device directly. Figure 6.5-2 shows the connection of 16-bit data width device and Figure 6.5-3 show the connection of 8-bit data width device.

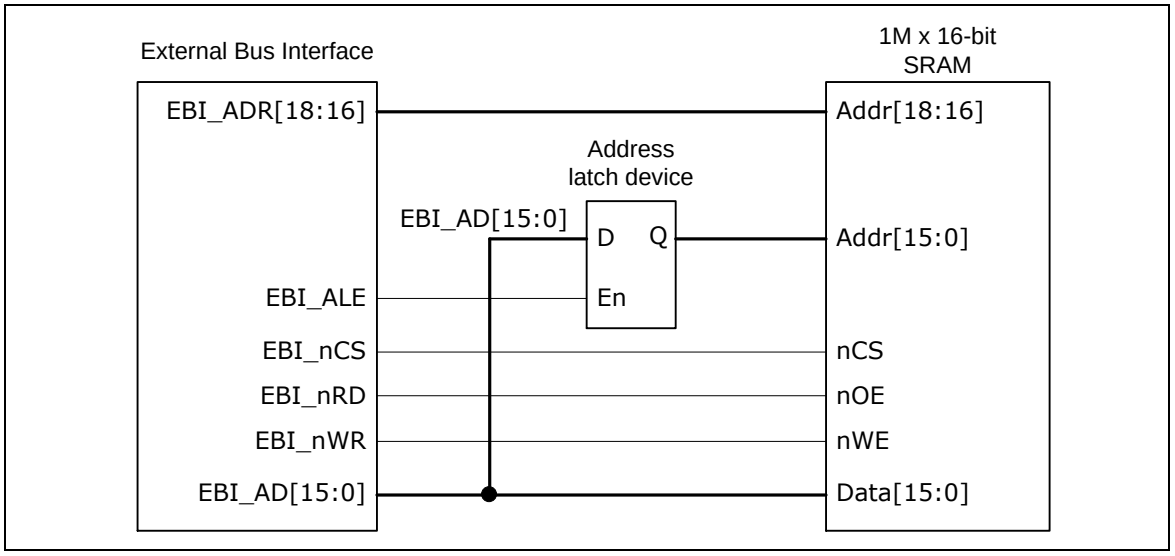


Figure 6.5-2 Connection of 16-bit EBI Data Width with 16-bit Device

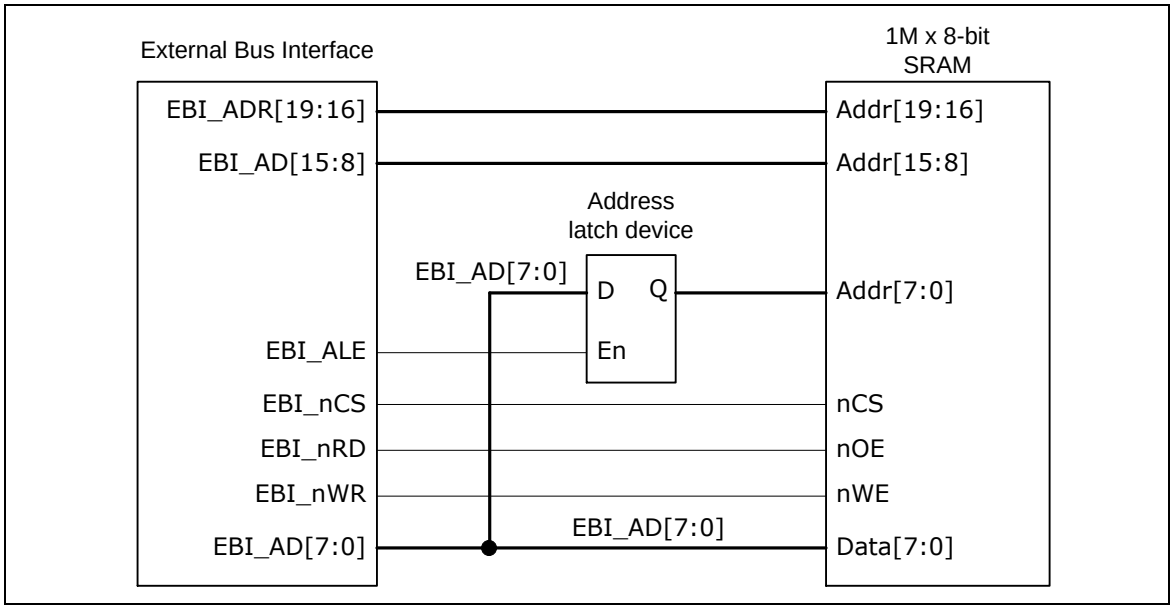


Figure 6.5-3 Connection of 8-bit EBI Data Width with 8-bit Device

When system access data width is larger than EBI data width, EBI controller will finish a system access command by operating EBI access more than once. For example, if system requests a 32-bit data through EBI device, EBI controller will operate accessing four times when setting EBI data width with 8-bit.

6.5.5.3 EBI Operating Control

MCLK Control

In the chip, all EBI signals will be synchronized by EBI_MCLK when EBI is operating. When chip connects to the external device with slower operating frequency, the EBI_MCLK can divide most to HCLK/32 by setting MCLKDIV (EBI_CTLx[10:8]). Therefore, chip can suitable for a wide frequency range of EBI device. If EBI_MCLK is set to HCLK/1, EBI signals are synchronized by positive edge of EBI_MCLK, else by negative edge of EBI_MCLK.

Operation and Access Timing Control

At the start of EBI access, chip select (EBI_nCS0 and EBI_nCS1) asserts to low and wait one EBI_MCLK for address setup time (tASU) for address stable. Then EBI_ALE asserts to high after address is stable and keeps for a period of time (tALE) for address latch. After latch address, EBI_ALE asserts to low and wait one EBI_MCLK for latch hold time (tLHD) and another one EBI_MCLK cycle (tA2D) that is inserted behind address hold time to be the bus turn-around time for address change to data. Then EBI_nRD asserts to low when read access or EBI_nWR asserts to low when write access. Then EBI_nRD or EBI_nWR asserts to high after keeps access time (tACC) for reading output stable or writing finish. After that, EBI signals keep for data access hold time (tAHD) and chip select asserts to high, address is released by current access control.

The EBI controller provides a flexible timing control for different external device. In EBI timing control, tASU, tLHD and tA2D are fixed to 1 EBI_MCLK cycle, tAHD can modulate to 1~8 EBI_MCLK cycles by setting TAHD (EBI_TCTLx[10:8]), tACC can modulate to 1~32 EBI_MCLK cycles by setting TACC (EBI_TCTLx[7:3]), and tALE can modulate to 1~8 EBI_MCLK cycles by setting TALE (EBI_CTL0[18:16]). Some external device can support zero data access hold time accessing, the EBI controller can skipped tAHD to increase access speed by setting WAHDOFF (EBI_TCTLx[23]) and RAHDOFF (EBI_TCTLx[22]).

For each chip select, the EBI provide individual register for timing control except tALE can only be controlled by EBI_CTL0.

Parameter	Value	Unit	Description
tASU	1	MCLK	Address Latch Setup Time.
tALE	1 ~ 8	MCLK	ALE High Period. Controlled by TALE (EBI_CTL0[18:16]).
tLHD	1	MCLK	Address Latch Hold Time.
tA2D	1	MCLK	Address To Data Delay (Bus Turn-Around Time).
tACC	1 ~ 32	MCLK	Data Access Time. Controlled by TACC (EBI_TCTLx[7:3]).
tAHD	1 ~ 8	MCLK	Data Access Hold Time. Controlled by TAHD (EBI_TCTLx[10:8]).
IDLE	0 ~ 15	MCLK	Idle Cycle. Controlled by R2R (EBI_TCTLx[27:24]) and W2X (EBI_TCTLx[15:12]).

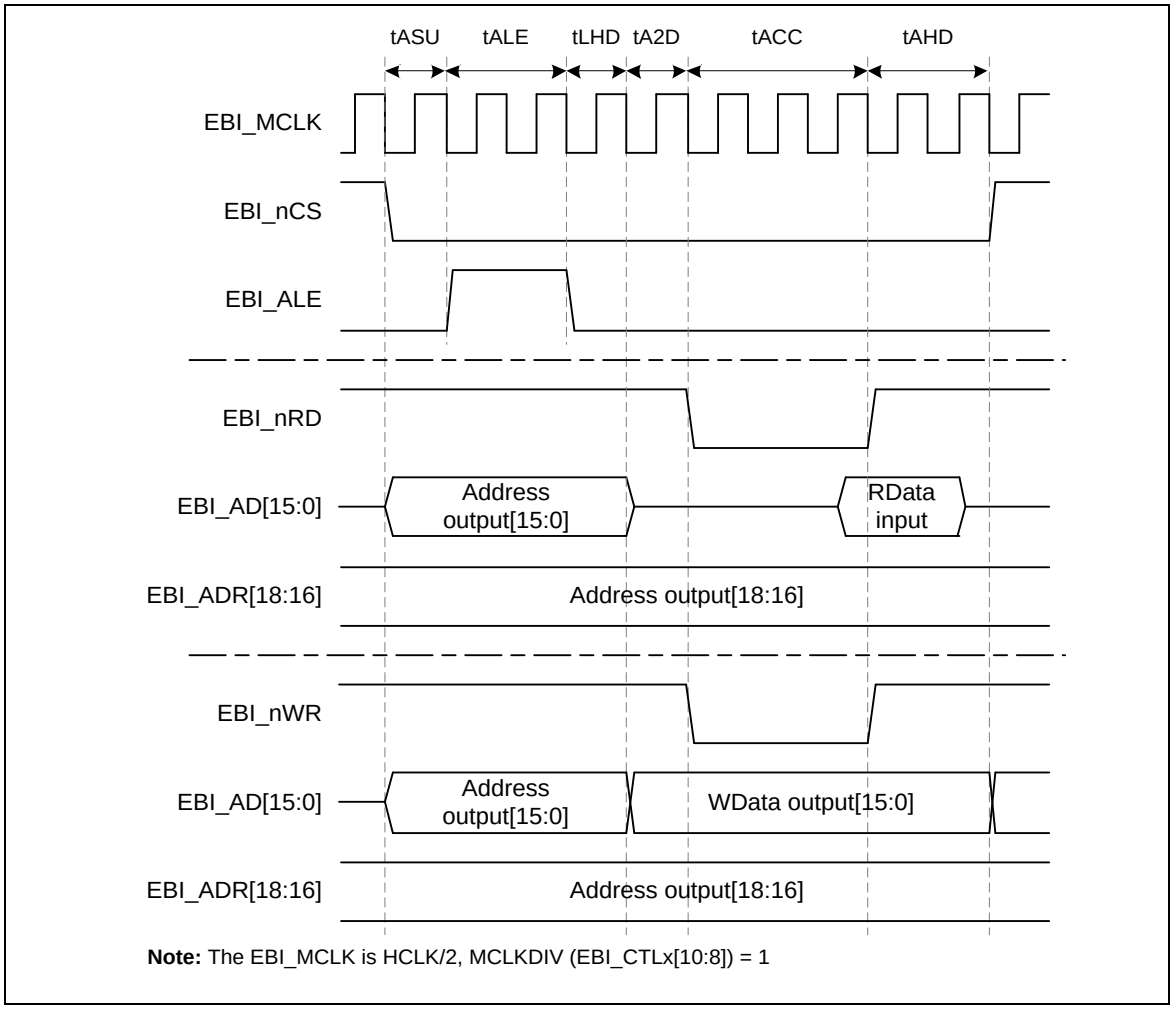


Figure 6.5-4 Timing Control Waveform for 16-bit Data Width

Figure 6.5-4 shows an example of setting 16-bit data width. In this example, EBI_AD bus is used for being address [15:0] and data [15:0]. When EBI_ALE assert to high, EBI_AD is address output. After address is latched, EBI_ALE asserts to low and the EBI_AD bus change to high impedance to wait device output data in read access operation, or it is used for being write data output.

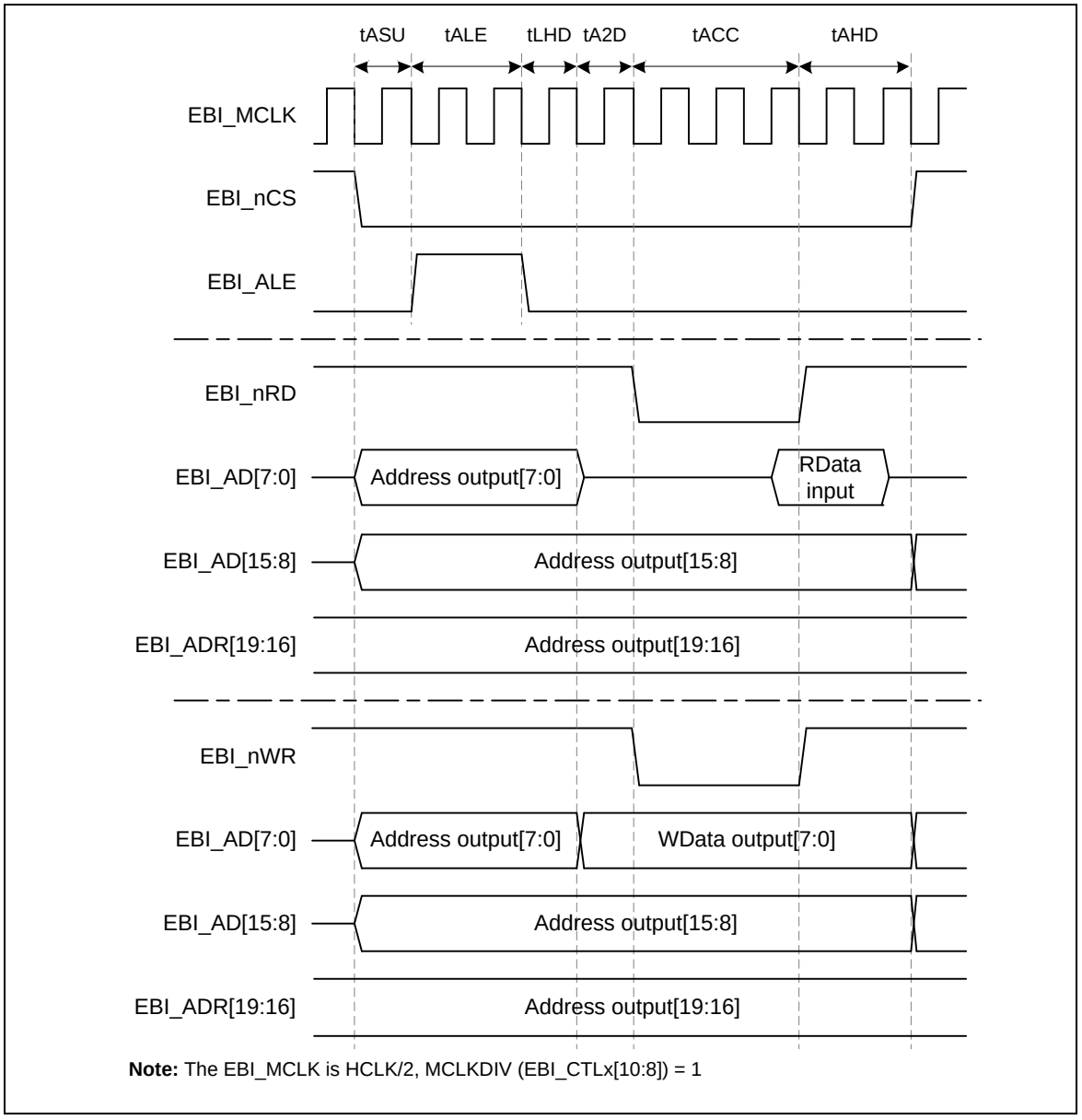


Figure 6.5-5 Timing Control Waveform for 8-bit Data Width

Figure 6.5-5 figure above shows an example of setting 8-bit data width. The difference between 8-bit and 16-bit data width is EBI_AD[15:8]. In 8-bit data width setting, EBI_AD[15:8] always be Address [15:8] output so that external latch need only 8-bit width.

Insert Idle Cycle

When EBI accessing continuously, there may occur bus conflict if the device access time is much slow with system operating. EBI controller supply additional idle cycle to solve this problem. During idle cycle, all control signals of EBI are inactive. Figure 6.5-6 shows idle cycle.

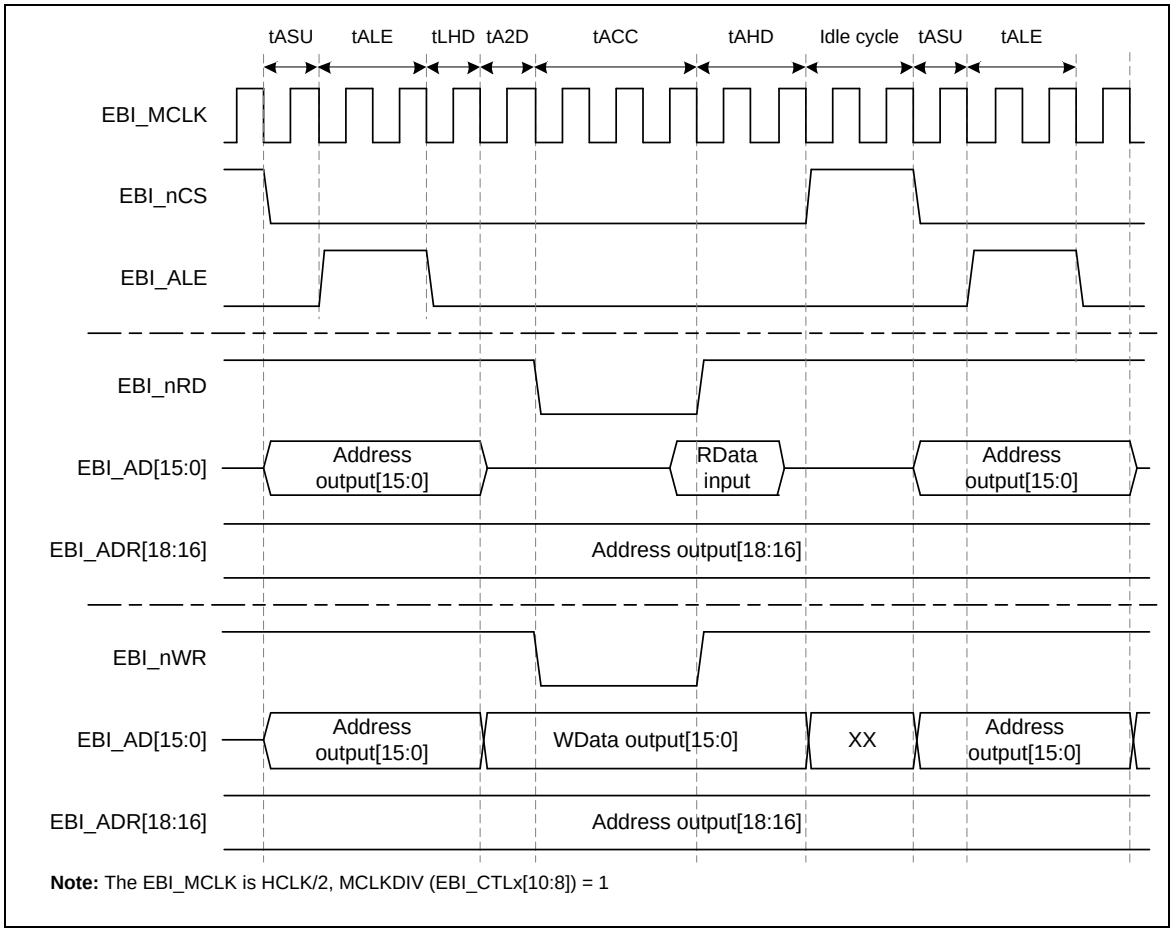


Figure 6.5-6 Timing Control Waveform for Insert Idle Cycle

There are two conditions that EBI can insert idle cycle by timing control:

1. After write access
2. After read access and before next read access (R2R idle cycle)

By setting W2X (EBI_TCTLx[15:12]), and R2R (EBI_TCTLx[27:24]), the time of idle cycle can be specified from 0~15 EBI_MCLK.

Write Buffer

When user write a data to external device through EBI bus, the EBI controller will start processing the write action immediately and the CPU is held until current EBI write action finish. User can enable write buffer function to improve CPU and EBI access performance. When EBI write buffer function is enabled, the CPU can continuous execute other instruction during EBI controller process the write action to external device. There is one exception condition for this case, if CPU executes another data access through EBI bus when EBI process write action, the CPU will be held.

6.5.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
EBI Base Address: **EBI_BA = 0x4001_0000**				
EBI_CTL0	EBI_BA+0x00	R/W	External Bus Interface Bank0 Control Register	0x0000_0000
EBI_TCTL0	EBI_BA+0x04	R/W	External Bus Interface Bank0 Timing Control Register	0x0000_0000
EBI_CTL1	EBI_BA+0x10	R/W	External Bus Interface Bank1 Control Register	0x0000_0000
EBI_TCTL1	EBI_BA+0x14	R/W	External Bus Interface Bank1 Timing Control Register	0x0000_0000

6.5.7 Register Description

External Bus Interface Control Register (EBI_CTLx)

Register	Offset	R/W	Description	Reset Value
EBI_CTL0	EBI_BA+0x00	R/W	External Bus Interface Bank0 Control Register	0x0000_0000
EBI_CTL1	EBI_BA+0x10	R/W	External Bus Interface Bank1 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reversed							WBUFEN
23	22	21	20	19	18	17	16
Reversed					TALE		
15	14	13	12	11	10	9	8
Reversed					MCLKDIV		
7	6	5	4	3	2	1	0
Reversed					CSPOLINV	DW16	EN

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**WBUFEN**	**EBI Write Buffer Enable Bit** 0 = EBI write buffer Disabled. 1 = EBI write buffer Enabled. **Note:** This bit only available in EBI_CTL0 register
[23:19]	**Reserved**	Reserved.
[18:16]	**TALE**	**Extend Time of ALE** The EBI_ALE high pulse period (tALE) to latch the address can be controlled by TALE. tALE = (TALE+1)*EBI_MCLK. **Note:** This field only available in EBI_CTL0 register
[15:11]	**Reserved**	Reserved.
[10:8]	**MCLKDIV**	**External Output Clock Divider** The frequency of EBI output clock (MCLK) is controlled by MCLKDIV as follow: 000 = HCLK/1. 001 = HCLK/2. 010 = HCLK/4. 011 = HCLK/8. 100 = HCLK/16. 101 = HCLK/32. 110 = Reserved. 111 = Reserved.
[7:3]	**Reserved**	Reserved.
[2]	**CSPOLINV**	**Chip Select Pin Polar Inverse**

		This bit defines the active level of EBI chip select pin (EBI_nCS). 0 = Chip select pin (EBI_nCS) is active low. 1 = Chip select pin (EBI_nCS) is active high.
[1]	**DW16**	**EBI Data Width 16-bit Select** This bit defines if the EBI data width is 8-bit or 16-bit. 0 = EBI data width is 8-bit. 1 = EBI data width is 16-bit.
[0]	**EN**	**EBI Enable Bit** This bit is the functional enable bit for EBI. 0 = EBI function Disabled. 1 = EBI function Enabled.

External Bus Interface Timing Control Register (EBI_TCTLx)

Register	Offset	R/W	Description	Reset Value
EBI_TCTL0	EBI_BA+0x04	R/W	External Bus Interface Bank0 Timing Control Register	0x0000_0000
EBI_TCTL1	EBI_BA+0x14	R/W	External Bus Interface Bank1 Timing Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				R2R			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
WAHDOFF	RAHDOFF	Reserved					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
W2X				Reversed	TAHD		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TACC					Reserved		

Bits	Description	
[31:30]	**Reserved**	Reserved.
[27:24]	**R2R**	**Idle Cycle Between Read-to-read** This field defines the number of R2R idle cycle. R2R idle cycle = (R2R * EBI_MCLK). When read action is finish and next action is going to read, R2R idle cycle is inserted and EBI_nCS return to idle state.
[23]	**WAHDOFF**	**Access Hold Time Disable Control When Write** 0 = The Data Access Hold Time (tAHD) during EBI writing is Enabled. 1 = The Data Access Hold Time (tAHD) during EBI writing is Disabled.
[22]	**RAHDOFF**	**Access Hold Time Disable Control When Read** 0 = The Data Access Hold Time (tAHD) during EBI reading is Enabled. 1 = The Data Access Hold Time (tAHD) during EBI reading is Disabled.
[21:16]	**Reserved**	Reserved.
[15:12]	**W2X**	**Idle Cycle After Write** This field defines the number of W2X idle cycle. W2X idle cycle = (W2X * EBI_MCLK). When write action is finish, W2X idle cycle is inserted and EBI_nCS return to idle state.
[11]	**Reserved**	Reserved.
[10:8]	**TAHD**	**EBI Data Access Hold Time** TAHD define data access hold time (tAHD). tAHD = (TAHD +1) * EBI_MCLK.
[7:3]	**TACC**	**EBI Data Access Time** TACC define data access time (tACC). tACC = (TACC +1) * EBI_MCLK.

[2:0]	**Reserved**	Reserved.

6.6 General Purpose I/O (GPIO)

6.6.1 Overview

The NuMicro® M4TK series has up to 87 General Purpose I/O pins to be shared with other function pins depending on the chip configuration. These 87 pins are arranged in 6 ports named as PA, PB, PC, PD, PE and PF. PA, PB, PC, and PD has 16 pins on port. PE has 15 pins on port.PF has 8 pins on port. Each of the 87 pins is independent and has the corresponding register bits to control the pin mode function and data.

The I/O type of each of I/O pins 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 are depending on CIOIN (CONFIG0[10]). Each I/O pin has a very weakly individual pull-up resistor which is about 110 kΩ ~ 300 kΩ for V_{DD} is from 5.0 V to 2.5 V.

6.6.2 Features

- Four I/O modes:
  - Quasi-bidirectional mode
  - Push-Pull Output mode
  - Open-Drain Output mode
  - Input only with high impendence mode
- TTL/Schmitt trigger input selectable
- I/O pin can be configured as interrupt source with edge/level setting
- Supports High Drive and High Slew Rate I/O mode
- Configurable default I/O mode of all pins after reset by CIOINI (CONFIG0[10]) setting
  - CIOIN = 0, all GPIO pins in Quasi-bidirectional mode after chip reset
  - CIOIN = 1, all GPIO pins in input mode after chip reset
- I/O pin internal pull-up resistor enabled only in Quasi-bidirectional I/O mode
- Enabling the pin interrupt function will also enable the wake-up function
- Supports 5V-tolerance function for following pins
  - PA.0 ~ PA.15, PC.0 ~ PC.7, PC.9 ~ PC.15, PD.5 ~ PD.8, PD.10 ~ PD.15, PE.0 ~ PE.14, PF.2, PF.5 ~ PF.6
  - PA.0 ~ PA.15, PB.14 ~ PB.14, PC.0 ~ PC.8, PC.10 ~ PC.13, PD.4 ~ PD.7, PD.12 ~ PD.15, PE.0 ~ PE.1, PE.3 ~ PE.5, PE.8 ~ PE.13, PF.2.

6.6.3 Block Diagram

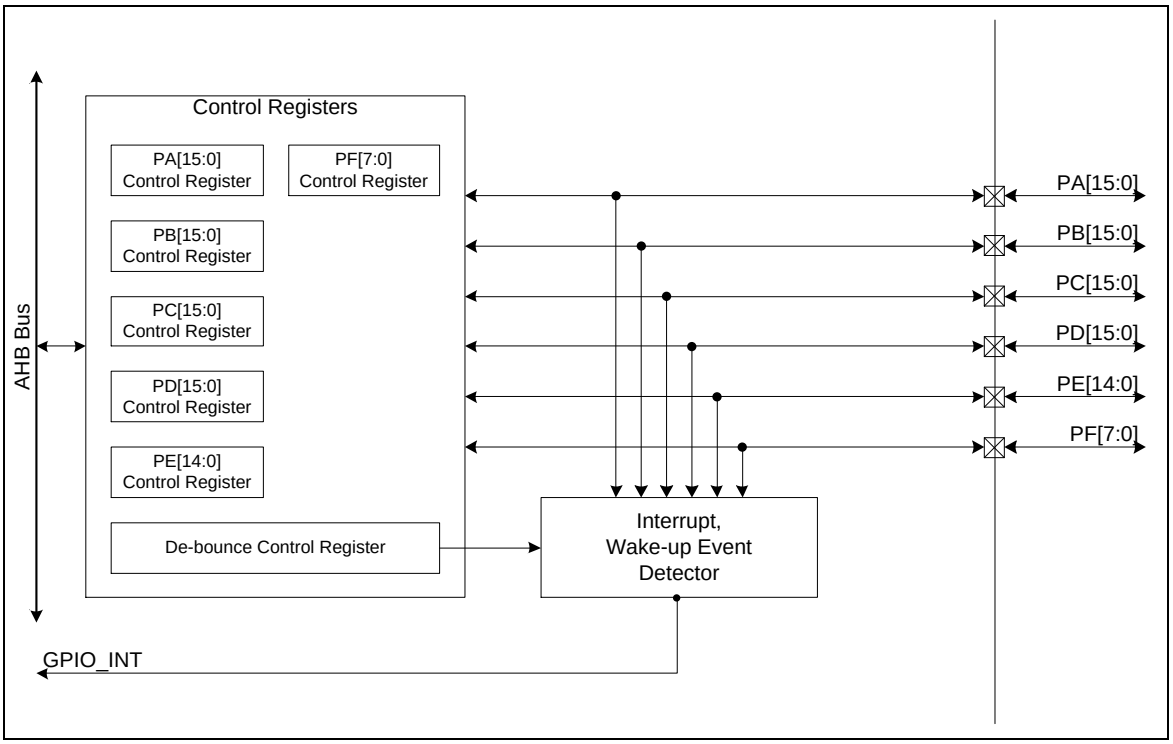


Figure 6.6-1 GPIO Controller Block Diagram

6.6.4 Basic Configuration

The GPIO pin functions are configured in SYS_GPA_MFPL, SYS_GPA_MFPH, SYS_GPB_MFPL, SYS_GPB_MFPH, SYS_GPC_MFPL, SYS_GPC_MFPH, SYS_GPD_MFPL, SYS_GPD_MFPH, SYS_GPE_MFPL, SYS_GPE_MFPH and SYS_GPF_MFPL registers.

6.6.5 Functional Description

6.6.5.1 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.

6.6.5.2 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.

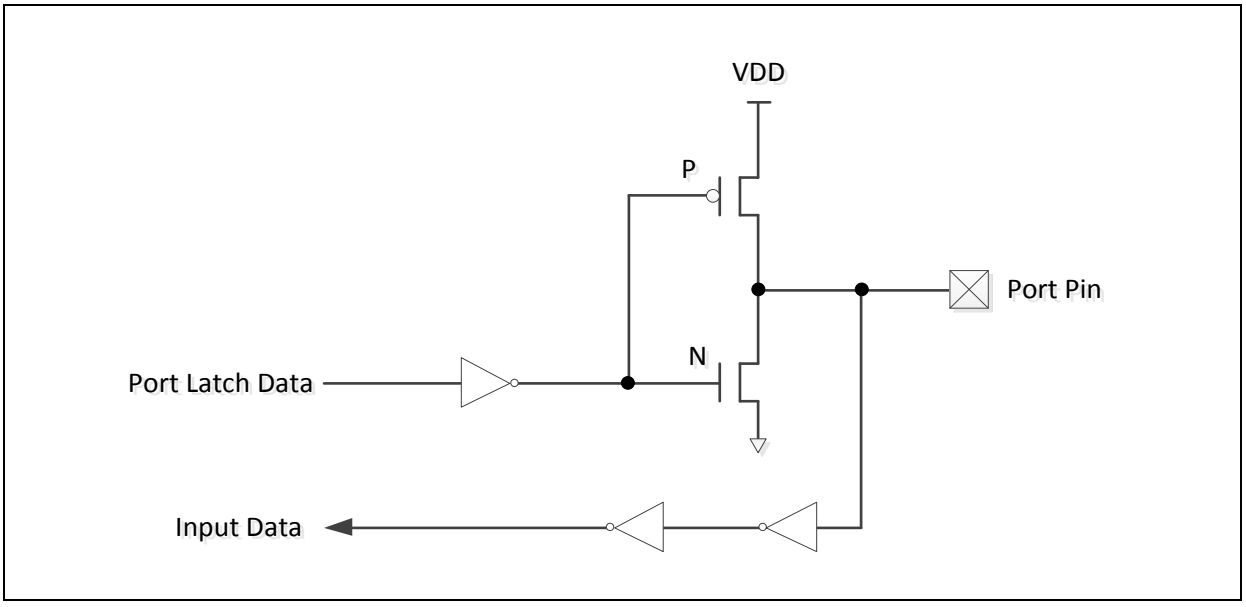


Figure 6.6-2 Push-Pull Output

6.6.5.3 Open-drain Mode

Set MODEn (Px_MODE[2n+1:2n]) to 10 the 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.

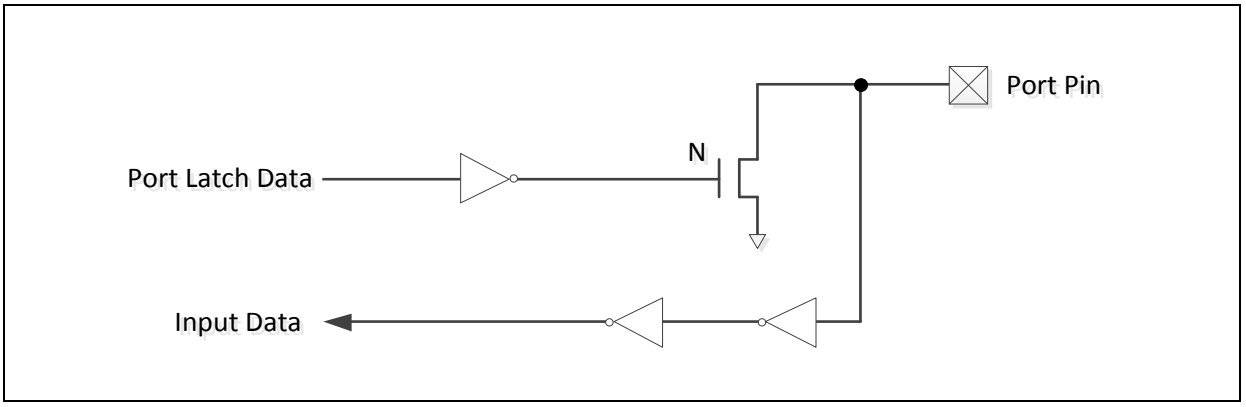


Figure 6.6-3 Open-Drain Output

6.6.5.4 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. The quasi-bidirectional output is common on the 80C51 and most of its derivatives. 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 will check the pin value. If pin value is high, no action takes. If pin state is low, the pin will drive strong high with 2 clock cycles on the pin and then disable the strong output drive. Meanwhile, the pin status is controlled by internal pull-up resistor. Note that the source current capability in quasi-bidirectional mode is only about 200 uA to 30 uA for V_{DD} is form 5.0 V to 2.5 V.

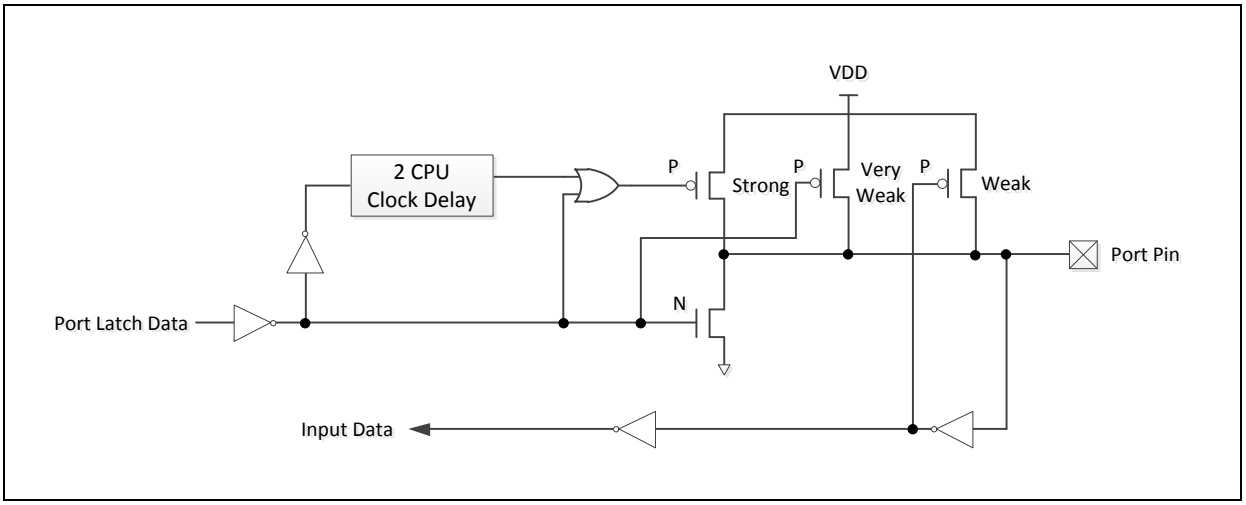


Figure 6.6-4 Quasi-Bidirectional I/O Mode

6.6.5.5 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.

6.6.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
GPIO Base Address: **GPIO_BA = 0x4000_4000**				
PA_MODE	GPIO_BA+0x000	R/W	PA I/O Mode Control	0xXXXX_XXXX
PA_DINOFF	GPIO_BA+0x004	R/W	PA Digital Input Path Disable Control	0x0000_0000
PA_DOUT	GPIO_BA+0x008	R/W	PA Data Output Value	0x0000_FFFF
PA_DATMSK	GPIO_BA+0x00C	R/W	PA Data Output Write Mask	0x0000_0000
PA_PIN	GPIO_BA+0x010	R	PA Pin Value	0x0000_XXXX
PA_DBEN	GPIO_BA+0x014	R/W	PA De-Bounce Enable Control Register	0x0000_0000
PA_INTTYPE	GPIO_BA+0x018	R/W	PA Interrupt Trigger Type Control	0x0000_0000
PA_INTEN	GPIO_BA+0x01C	R/W	PA Interrupt Enable Control Register	0x0000_0000
PA_INTSRC	GPIO_BA+0x020	R/W	PA Interrupt Source Flag	0x0000_XXXX
PA_SMTEN	GPIO_BA+0x024	R/W	PA Input Schmitt Trigger Enable Register	0x0000_0000
PA_SLEWCTL	GPIO_BA+0x028	R/W	PA High Slew Rate Control Register	0x0000_0000
PB_MODE	GPIO_BA+0x040	R/W	PB I/O Mode Control	0xXXXX_XXXX
PB_DINOFF	GPIO_BA+0x044	R/W	PB Digital Input Path Disable Control	0x0000_0000
PB_DOUT	GPIO_BA+0x048	R/W	PB Data Output Value	0x0000_FFFF
PB_DATMSK	GPIO_BA+0x04C	R/W	PB Data Output Write Mask	0x0000_0000
PB_PIN	GPIO_BA+0x050	R	PB Pin Value	0x0000_XXXX
PB_DBEN	GPIO_BA+0x054	R/W	PB De-Bounce Enable Control Register	0x0000_0000
PB_INTTYPE	GPIO_BA+0x058	R/W	PB Interrupt Trigger Type Control	0x0000_0000
PB_INTEN	GPIO_BA+0x05C	R/W	PB Interrupt Enable Control Register	0x0000_0000
PB_INTSRC	GPIO_BA+0x060	R/W	PB Interrupt Source Flag	0x0000_XXXX
PB_SMTEN	GPIO_BA+0x064	R/W	PB Input Schmitt Trigger Enable Register	0x0000_0000
PB_SLEWCTL	GPIO_BA+0x068	R/W	PB High Slew Rate Control Register	0x0000_0000
PC_MODE	GPIO_BA+0x080	R/W	PC I/O Mode Control	0xXXXX_XXXX
PC_DINOFF	GPIO_BA+0x084	R/W	PC Digital Input Path Disable Control	0x0000_0000
PC_DOUT	GPIO_BA+0x088	R/W	PC Data Output Value	0x0000_FFFF
PC_DATMSK	GPIO_BA+0x08C	R/W	PC Data Output Write Mask	0x0000_0000
PC_PIN	GPIO_BA+0x090	R	PC Pin Value	0x0000_XXXX

PC_DBEN	GPIO_BA+0x094	R/W	PC De-Bounce Enable Control Register	0x0000_0000
PC_INTTYPE	GPIO_BA+0x098	R/W	PC Interrupt Trigger Type Control	0x0000_0000
PC_INTEN	GPIO_BA+0x09C	R/W	PC Interrupt Enable Control Register	0x0000_0000
PC_INTSRC	GPIO_BA+0x0A0	R/W	PC Interrupt Source Flag	0x0000_XXXX
PC_SMTEN	GPIO_BA+0x0A4	R/W	PC Input Schmitt Trigger Enable Register	0x0000_0000
PC_SLEWCTL	GPIO_BA+0x0A8	R/W	PC High Slew Rate Control Register	0x0000_0000
PD_MODE	GPIO_BA+0x0C0	R/W	PD I/O Mode Control	0xXXXX_XXXX
PD_DINOFF	GPIO_BA+0x0C4	R/W	PD Digital Input Path Disable Control	0x0000_0000
PD_DOUT	GPIO_BA+0x0C8	R/W	PD Data Output Value	0x0000_FFFF
PD_DATMSK	GPIO_BA+0x0CC	R/W	PD Data Output Write Mask	0x0000_0000
PD_PIN	GPIO_BA+0x0D0	R	PD Pin Value	0x0000_XXXX
PD_DBEN	GPIO_BA+0x0D4	R/W	PD De-Bounce Enable Control Register	0x0000_0000
PD_INTTYPE	GPIO_BA+0x0D8	R/W	PD Interrupt Trigger Type Control	0x0000_0000
PD_INTEN	GPIO_BA+0x0DC	R/W	PD Interrupt Enable Control Register	0x0000_0000
PD_INTSRC	GPIO_BA+0x0E0	R/W	PD Interrupt Source Flag	0x0000_XXXX
PD_SMTEN	GPIO_BA+0x0E4	R/W	PD Input Schmitt Trigger Enable Register	0x0000_0000
PD_SLEWCTL	GPIO_BA+0x0E8	R/W	PD High Slew Rate Control Register	0x0000_0000
PE_MODE	GPIO_BA+0x100	R/W	PE I/O Mode Control	0xXXXX_XXXX
PE_DINOFF	GPIO_BA+0x104	R/W	PE Digital Input Path Disable Control	0x0000_0000
PE_DOUT	GPIO_BA+0x108	R/W	PE Data Output Value	0x0000_FFFF
PE_DATMSK	GPIO_BA+0x10C	R/W	PE Data Output Write Mask	0x0000_0000
PE_PIN	GPIO_BA+0x110	R	PE Pin Value	0x0000_XXXX
PE_DBEN	GPIO_BA+0x114	R/W	PE De-Bounce Enable Control Register	0x0000_0000
PE_INTTYPE	GPIO_BA+0x118	R/W	PE Interrupt Trigger Type Control	0x0000_0000
PE_INTEN	GPIO_BA+0x11C	R/W	PE Interrupt Enable Control Register	0x0000_0000
PE_INTSRC	GPIO_BA+0x120	R/W	PE Interrupt Source Flag	0x0000_XXXX
PE_SMTEN	GPIO_BA+0x124	R/W	PE Input Schmitt Trigger Enable Register	0x0000_0000
PE_SLEWCTL	GPIO_BA+0x128	R/W	PE High Slew Rate Control Register	0x0000_0000
PE_DRVCTL	GPIO_BA+0x12C	R/W	PE High Drive Strength Control Register	0x0000_0000
PF_MODE	GPIO_BA+0x140	R/W	PF I/O Mode Control	0x0000_XXXX
PF_DINOFF	GPIO_BA+0x144	R/W	PF Digital Input Path Disable Control	0x0000_0000

PF_DOUT	GPIO_BA+0x148	R/W	PF Data Output Value	0x0000_00FF
PF_DATMSK	GPIO_BA+0x14C	R/W	PF Data Output Write Mask	0x0000_0000
PF_PIN	GPIO_BA+0x150	R	PF Pin Value	0x0000_00XX
PF_DBEN	GPIO_BA+0x154	R/W	PF De-Bounce Enable Control Register	0x0000_0000
PF_INTTYPE	GPIO_BA+0x158	R/W	PF Interrupt Trigger Type Control	0x0000_0000
PF_INTEN	GPIO_BA+0x15C	R/W	PF Interrupt Enable Control Register	0x0000_0000
PF_INTSRC	GPIO_BA+0x160	R/W	PF Interrupt Source Flag	0x0000_00XX
PF_SMTEN	GPIO_BA+0x164	R/W	PF Input Schmitt Trigger Enable Register	0x0000_0000
PF_SLEWCTL	GPIO_BA+0x168	R/W	PF High Slew Rate Control Register	0x0000_0000
GPIO_DBCTL	GPIO_BA+0x440	R/W	Interrupt De-bounce Control Register	0x0000_0020
PAn_PDIO n=0,1..15	GPIO_BA+0x800+(0x04 * n)	R/W	GPIO PA.n Pin Data Input/Output Register	0x0000_000X
PBn_PDIO n=0,1..15	GPIO_BA+0x840+(0x04 * n)	R/W	GPIO PB.n Pin Data Input/Output Register	0x0000_000X
PCn_PDIO n=0,1..15	GPIO_BA+0x880+(0x04 * n)	R/W	GPIO PC.n Pin Data Input/Output Register	0x0000_000X
PDn_PDIO n=0,1..15	GPIO_BA+0x8C0+(0x04 * n)	R/W	GPIO PD.n Pin Data Input/Output Register	0x0000_000X
PEn_PDIO n=0,1..14	GPIO_BA+0x900+(0x04 * n)	R/W	GPIO PE.n Pin Data Input/Output Register	0x0000_000X
PFn_PDIO n=0,1..7	GPIO_BA+0x940+(0x04 * n)	R/W	GPIO PF.n Pin Data Input/Output Register	0x0000_000X

6.6.7 Register Description

Port A-F I/O Mode Control (Px_MODE)

Register	Offset	R/W	Description	Reset Value
PA_MODE	GPIO_BA+0x000	R/W	PA I/O Mode Control	0xXXXX_XXXX
PB_MODE	GPIO_BA+0x040	R/W	PB I/O Mode Control	0xXXXX_XXXX
PC_MODE	GPIO_BA+0x080	R/W	PC I/O Mode Control	0xXXXX_XXXX
PD_MODE	GPIO_BA+0x0C0	R/W	PD I/O Mode Control	0xXXXX_XXXX
PE_MODE	GPIO_BA+0x100	R/W	PE I/O Mode Control	0xXXXX_XXXX
PF_MODE	GPIO_BA+0x140	R/W	PF I/O Mode Control	0x0000_XXXX

31	30	29	28	27	26	25	24
MODE15		MODE14		MODE13		MODE12	
23	22	21	20	19	18	17	16
MODE11		MODE10		MODE9		MODE8	
15	14	13	12	11	10	9	8
MODE7		MODE6		MODE5		MODE4	
7	6	5	4	3	2	1	0
MODE3		MODE2		MODE1		MODE0	

Bits	Description	
[2n+1:2n] n=0,1..15	**MODEn**	**Port A-f 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. **Note1:** The initial value of this field is defined by CIOINI (CONFIG0 [10]). If CIOINI is set to 0, the default value is 0xFFFF_FFFF and all pins will be quasi-bidirectional mode after chip powered on. If CIOINI is set to 1, the default value is 0x0000_0000 and all pins will be input mode after chip powered on. **Note2:** Max. n=15 for port A/B/C/D. Max. n=14 for port E. Max. n=7 for port F.

Port A-F Digital Input Path Disable Control (Px_DINOFF)

Register	Offset	R/W	Description	Reset Value
PA_DINOFF	GPIO_BA+0x004	R/W	PA Digital Input Path Disable Control	0x0000_0000
PB_DINOFF	GPIO_BA+0x044	R/W	PB Digital Input Path Disable Control	0x0000_0000
PC_DINOFF	GPIO_BA+0x084	R/W	PC Digital Input Path Disable Control	0x0000_0000
PD_DINOFF	GPIO_BA+0x0C4	R/W	PD Digital Input Path Disable Control	0x0000_0000
PE_DINOFF	GPIO_BA+0x104	R/W	PE Digital Input Path Disable Control	0x0000_0000
PF_DINOFF	GPIO_BA+0x144	R/W	PF Digital Input Path Disable Control	0x0000_0000

31	30	29	28	27	26	25	24
DINOFF							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DINOFF							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[n+16] n=0,1..15	**DINOFF**	**Port A-f Pin[N] Digital Input Path Disable Control** Each of these bits is used to control if the digital input path of corresponding Px.n pin is disabled. If input is analog signal, users can disable Px.n digital input path to avoid input current leakage. 0 = Px.n digital input path Enabled. 1 = Px.n digital input path Disabled (digital input tied to low). **Note1:** Max. n=15 for port A/B/C/D. Max. n=14 for port E. Max. n=7 for port F.
[15:0]	**Reserved**	Reserved.

Port A-F Data Output Value (Px_DOUT)

Register	Offset	R/W	Description	Reset Value
PA_DOUT	GPIO_BA+0x008	R/W	PA Data Output Value	0x0000_FFFF
PB_DOUT	GPIO_BA+0x048	R/W	PB Data Output Value	0x0000_FFFF
PC_DOUT	GPIO_BA+0x088	R/W	PC Data Output Value	0x0000_FFFF
PD_DOUT	GPIO_BA+0x0C8	R/W	PD Data Output Value	0x0000_FFFF
PE_DOUT	GPIO_BA+0x108	R/W	PE Data Output Value	0x0000_FFFF
PF_DOUT	GPIO_BA+0x148	R/W	PF Data Output Value	0x0000_00FF

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DOUT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DOUT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**DOUT**	**Port A-f 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. **Note1:** Max. n=15 for port A/B/C/D. Max. n=14 for port E. Max. n=7 for port F.

Port A-F Data Output Write Mask (Px_DATMSK)

Register	Offset	R/W	Description	Reset Value
PA_DATMSK	GPIO_BA+0x00C	R/W	PA Data Output Write Mask	0x0000_0000
PB_DATMSK	GPIO_BA+0x04C	R/W	PB Data Output Write Mask	0x0000_0000
PC_DATMSK	GPIO_BA+0x08C	R/W	PC Data Output Write Mask	0x0000_0000
PD_DATMSK	GPIO_BA+0x0CC	R/W	PD Data Output Write Mask	0x0000_0000
PE_DATMSK	GPIO_BA+0x10C	R/W	PE Data Output Write Mask	0x0000_0000
PF_DATMSK	GPIO_BA+0x14C	R/W	PF Data Output Write Mask	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DATMSK							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATMSK							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[n] n=0,1..15	**DATMSK**	**Port A-f 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 ignored. 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=15 for port A/B/C/D. Max. n=14 for port E. Max. n=7 for port F.

Port A-F Pin Value (Px_PIN)

Register	Offset	R/W	Description	Reset Value
PA_PIN	GPIO_BA+0x010	R	PA Pin Value	0x0000_XXXX
PB_PIN	GPIO_BA+0x050	R	PB Pin Value	0x0000_XXXX
PC_PIN	GPIO_BA+0x090	R	PC Pin Value	0x0000_XXXX
PD_PIN	GPIO_BA+0x0D0	R	PD Pin Value	0x0000_XXXX
PE_PIN	GPIO_BA+0x110	R	PE Pin Value	0x0000_XXXX
PF_PIN	GPIO_BA+0x150	R	PF Pin Value	0x0000_00XX

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PIN							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PIN							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**PIN**	**Port A-f 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. **Note1:** Max. n=15 for port A/B/C/D. Max. n=14 for port E. Max. n=7 for port F.

Port A-F De-Bounce Enable Control Register (Px_DBEN)

Register	Offset	R/W	Description	Reset Value
PA_DBEN	GPIO_BA+0x014	R/W	PA De-Bounce Enable Control Register	0x0000_0000
PB_DBEN	GPIO_BA+0x054	R/W	PB De-Bounce Enable Control Register	0x0000_0000
PC_DBEN	GPIO_BA+0x094	R/W	PC De-Bounce Enable Control Register	0x0000_0000
PD_DBEN	GPIO_BA+0x0D4	R/W	PD De-Bounce Enable Control Register	0x0000_0000
PE_DBEN	GPIO_BA+0x114	R/W	PE De-Bounce Enable Control Register	0x0000_0000
PF_DBEN	GPIO_BA+0x154	R/W	PF De-Bounce Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DBEN							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DBEN							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**DBEN**	**Port A-f 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 ignored. **Note1:** Max. n=15 for port A/B/C/D. Max. n=14 for port E. Max. n=7 for port F.

Port A-F Interrupt Type Control (Px_INTTYPE)

Register	Offset	R/W	Description	Reset Value
PA_INTTYPE	GPIO_BA+0x018	R/W	PA Interrupt Trigger Type Control	0x0000_0000
PB_INTTYPE	GPIO_BA+0x058	R/W	PB Interrupt Trigger Type Control	0x0000_0000
PC_INTTYPE	GPIO_BA+0x098	R/W	PC Interrupt Trigger Type Control	0x0000_0000
PD_INTTYPE	GPIO_BA+0x0D8	R/W	PD Interrupt Trigger Type Control	0x0000_0000
PE_INTTYPE	GPIO_BA+0x118	R/W	PE Interrupt Trigger Type Control	0x0000_0000
PF_INTTYPE	GPIO_BA+0x158	R/W	PF Interrupt Trigger Type Control	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TYPE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TYPE							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**TYPE**	**Port A-f 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 ignored. **Note1:** Max. n=15 for port A/B/C/D. Max. n=14 for port E. Max. n=7 for port F.

Port A-F Interrupt Enable Control Register (Px_INTEN)

Register	Offset	R/W	Description	Reset Value
PA_INTEN	GPIO_BA+0x01C	R/W	PA Interrupt Enable Control Register	0x0000_0000
PB_INTEN	GPIO_BA+0x05C	R/W	PB Interrupt Enable Control Register	0x0000_0000
PC_INTEN	GPIO_BA+0x09C	R/W	PC Interrupt Enable Control Register	0x0000_0000
PD_INTEN	GPIO_BA+0x0DC	R/W	PD Interrupt Enable Control Register	0x0000_0000
PE_INTEN	GPIO_BA+0x11C	R/W	PE Interrupt Enable Control Register	0x0000_0000
PF_INTEN	GPIO_BA+0x15C	R/W	PF Interrupt Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
RHIEN							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RHIEN							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FLIEN							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FLIEN							

Bits	Description	
[n+16] n=0,1..15	**RHIEN**	**Port A-f 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. **Note1:** Max. n=15 for port A/B/C/D. Max. n=14 for port E. Max. n=7 for port F.
[n] n=0,1..15	**FLIEN**	**Port A-f 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. **Note1:** Max. n=15 for port A/B/C/D. Max. n=14 for port E. Max. n=7 for port F.

Port A-F Interrupt Source Flag (Px_INTSRC)

Register	Offset	R/W	Description	Reset Value
PA_INTSRC	GPIO_BA+0x020	R/W	PA Interrupt Source Flag	0x0000_XXXX
PB_INTSRC	GPIO_BA+0x060	R/W	PB Interrupt Source Flag	0x0000_XXXX
PC_INTSRC	GPIO_BA+0x0A0	R/W	PC Interrupt Source Flag	0x0000_XXXX
PD_INTSRC	GPIO_BA+0x0E0	R/W	PD Interrupt Source Flag	0x0000_XXXX
PE_INTSRC	GPIO_BA+0x120	R/W	PE Interrupt Source Flag	0x0000_XXXX
PF_INTSRC	GPIO_BA+0x160	R/W	PF Interrupt Source Flag	0x0000_00XX

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
INTSRC							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INTSRC							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**INTSRC**	**Port A-f 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. **Note1:** Max. n=15 for port A/B/C/D. Max. n=14 for port E. Max. n=7 for port F.

Port A-F Input Schmitt Trigger Enable Register (Px_SMTEN)

Register	Offset	R/W	Description	Reset Value
PA_SMTEN	GPIO_BA+0x024	R/W	PA Input Schmitt Trigger Enable Register	0x0000_0000
PB_SMTEN	GPIO_BA+0x064	R/W	PB Input Schmitt Trigger Enable Register	0x0000_0000
PC_SMTEN	GPIO_BA+0x0A4	R/W	PC Input Schmitt Trigger Enable Register	0x0000_0000
PD_SMTEN	GPIO_BA+0x0E4	R/W	PD Input Schmitt Trigger Enable Register	0x0000_0000
PE_SMTEN	GPIO_BA+0x124	R/W	PE Input Schmitt Trigger Enable Register	0x0000_0000
PF_SMTEN	GPIO_BA+0x164	R/W	PF Input Schmitt Trigger Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SMTEN							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SMTEN							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**SMTEN**	**Port A-f Pin[N] Input Schmitt Trigger Enable Bit** 0 = Px.n input schmitt trigger function Disabled. 1 = Px.n input schmitt trigger function Enabled. **Note1:** Max. n=15 for port A/B/C/D. Max. n=14 for port E. Max. n=7 for port F.

Port A-F High Slew Rate Control Register (Px_SLEWCTL)

Register	Offset	R/W	Description	Reset Value
PA_SLEWCTL	GPIO_BA+0x028	R/W	PA High Slew Rate Control Register	0x0000_0000
PB_SLEWCTL	GPIO_BA+0x068	R/W	PB High Slew Rate Control Register	0x0000_0000
PC_SLEWCTL	GPIO_BA+0x0A8	R/W	PC High Slew Rate Control Register	0x0000_0000
PD_SLEWCTL	GPIO_BA+0x0E8	R/W	PD High Slew Rate Control Register	0x0000_0000
PE_SLEWCTL	GPIO_BA+0x128	R/W	PE High Slew Rate Control Register	0x0000_0000
PF_SLEWCTL	GPIO_BA+0x168	R/W	PF High Slew Rate Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
HSREN							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
HSREN							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[n] n=0,1..15	**HSREN**	**Port A-f Pin[N] High Slew Rate Control** 0 = Px.n output with basic slew rate. 1 = Px.n output with higher slew rate. **Note1:** Max. n=15 for port A/B/C/D. Max. n=14 for port E. Max. n=7 for port F.

Port E High Drive Strength Control Register (PE_DRVCTL)

Register	Offset	R/W	Description	Reset Value
PE_DRVCTL	GPIO_BA+0x12C	R/W	PE High Drive Strength Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		HDRVEN[13:8]					
7	6	5	4	3	2	1	0
Reserved							

Bits	Description	
[31:14]	**Reserved**	Reserved.
[n] n=8,9..13	**HDRVEN[13:8]**	**Port E Pin[N] Driving Strength Control** 0 = Px.n output with basic driving strength. 1 = Px.n output with high driving strength. **Note:** n=8,9..13 for port E.
[7:0]	**Reserved**	Reserved.

Interrupt De-bounce Control Register (GPIO_DBCTL)

Register	Offset	R/W	Description	Reset Value
GPIO_DBCTL	GPIO_BA+0x440	R/W	Interrupt De-bounce Control Register	0x0000_0020

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		ICLKON	DBCLKSRC	DBCLKSEL			

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**ICLKON**	**Interrupt Clock on Mode** 0 = Edge detection circuit is active only if I/O pin corresponding RHIEN (Px_INTEN[n+16])/FLIEN (Px_INTEN[n]) bit is set to 1. 1 = All I/O pins edge detection circuit is always active after reset. **Note:** It is recommended to disable this bit to save system power if no special application concern.
[4]	**DBCLKSRC**	**De-bounce Counter Clock Source Selection** 0 = De-bounce counter clock source is the HCLK. 1 = De-bounce counter clock source is the 10 kHz internal low speed RC oscillator (LIRC).
[3:0]	**DBCLKSEL**	**De-bounce Sampling Cycle Selection** 0000 = Sample interrupt input once per 1 clocks. 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.

GPIO Px.n Pin Data Input/Outut Register (Pxn_PDIO)

Register	Offset	R/W	Description	Reset Value
PAn_PDIO n=0,1..15	GPIO_BA+0x800+(0x04 * n)	R/W	GPIO PA.n Pin Data Input/Output Register	0x0000_000X
PBn_PDIO n=0,1..15	GPIO_BA+0x840+(0x04 * n)	R/W	GPIO PB.n Pin Data Input/Output Register	0x0000_000X
PCn_PDIO n=0,1..15	GPIO_BA+0x880+(0x04 * n)	R/W	GPIO PC.n Pin Data Input/Output Register	0x0000_000X
PDn_PDIO n=0,1..15	GPIO_BA+0x8C0+(0x04 * n)	R/W	GPIO PD.n Pin Data Input/Output Register	0x0000_000X
PEn_PDIO n=0,1..14	GPIO_BA+0x900+(0x04 * n)	R/W	GPIO PE.n Pin Data Input/Output Register	0x0000_000X
PFn_PDIO n=0,1..7	GPIO_BA+0x940+(0x04 * n)	R/W	GPIO PF.n Pin Data Input/Output Register	0x0000_000X

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**PDIO**

Bits	Description	
[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 PA0_PDIO will reflect the written value to bit DOUT (Px_DOUT[0]), reading PA0_PDIO will return the value of PIN (PA_PIN[0]). **Note1:** The writing operation will not be affected by register DATMSK (Px_DATMSK[n]). **Note2:** Max. n=15 for port A/B/C/D. Max. n=14 for port E. Max. n=7 for port F.

6.7 PDMA Controller (PDMA)

6.7.1 Overview

The peripheral direct memory access (PDMA) controller is used to provide high-speed data transfer. The PDMA controller can transfer data from one address to another without CPU intervention. This has the benefit of reducing the workload of CPU and keeps CPU resources free for other applications. The PDMA controller has a total of 12 channels and each channel can perform transfer between memory and peripherals or between memory and memory.

6.7.2 Features

- Supports 12 independently configurable channels
- Supports selectable 2 level of priority (fixed priority or round-robin priority)
- Supports transfer data width of 8, 16, and 32 bits
- Supports source and destination address increment size can be byte, half-word, word or no increment
- Supports software and SPI, UART, DAC, ADC and PWM request
- Supports Scatter-Gather mode to perform sophisticated transfer through the use of the descriptor link list table
- Supports single and burst transfer type

6.7.3 Block Diagram

The block diagram about PDMA controller is shown as follows.

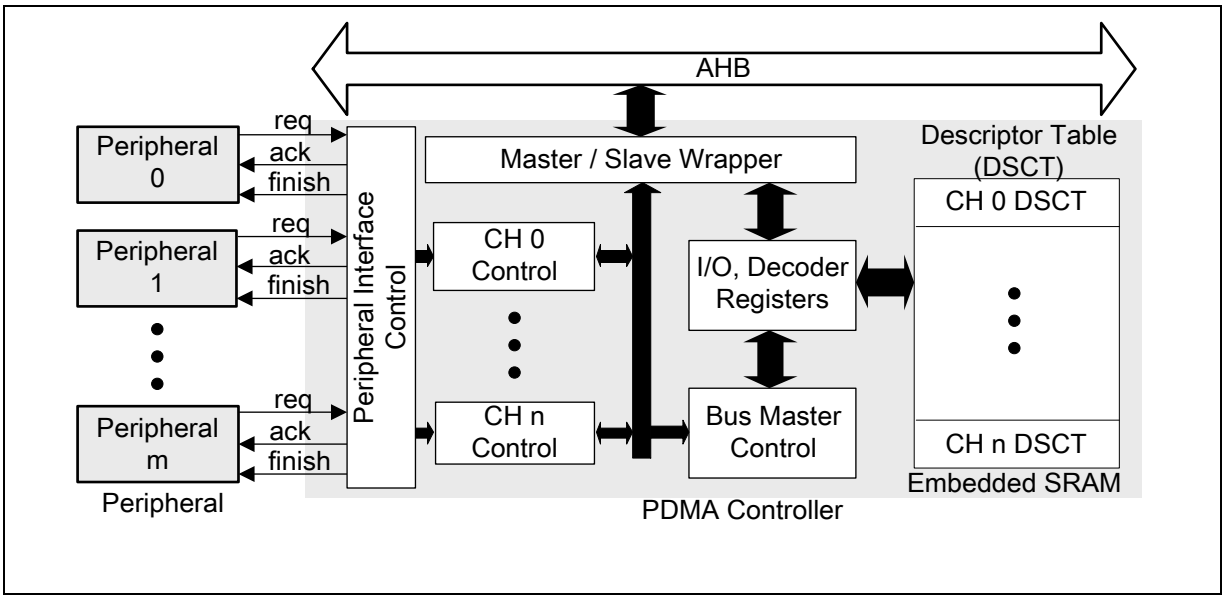


Figure 6.7-1 PDMA Controller Block Diagram

6.7.4 Basic Configuration

The peripheral direct memory access (PDMA) controller peripheral clock is enabled in PDMACKEN (CLK_AHBCLK[1]).

6.7.5 Functional Description

The PDMA controller transfers data from one address to another without CPU intervention. For M4TK, the PDMA controller supports 12 independent channels, as the result, PDMA controller supports two level channel priorities: fixed and round-robin priority, PDMA controller serves channel in order from highest to lowest priority channel. The PDMA controller supports two operation modes: Basic mode and Scatter-gather mode. Basic mode is used to perform one descriptor table transfer. Scatter-gather mode has more entries for each PDMA channel, and thus the PDMA controller supports sophisticated transfer through the entries. The descriptor table entry data structure contains many transfer information including the transfer source address, transfer destination address, transfer count, burst size, transfer type and operation mode. Figure 6.7-2 shows the diagram of descriptor table (DSCT) data structure.

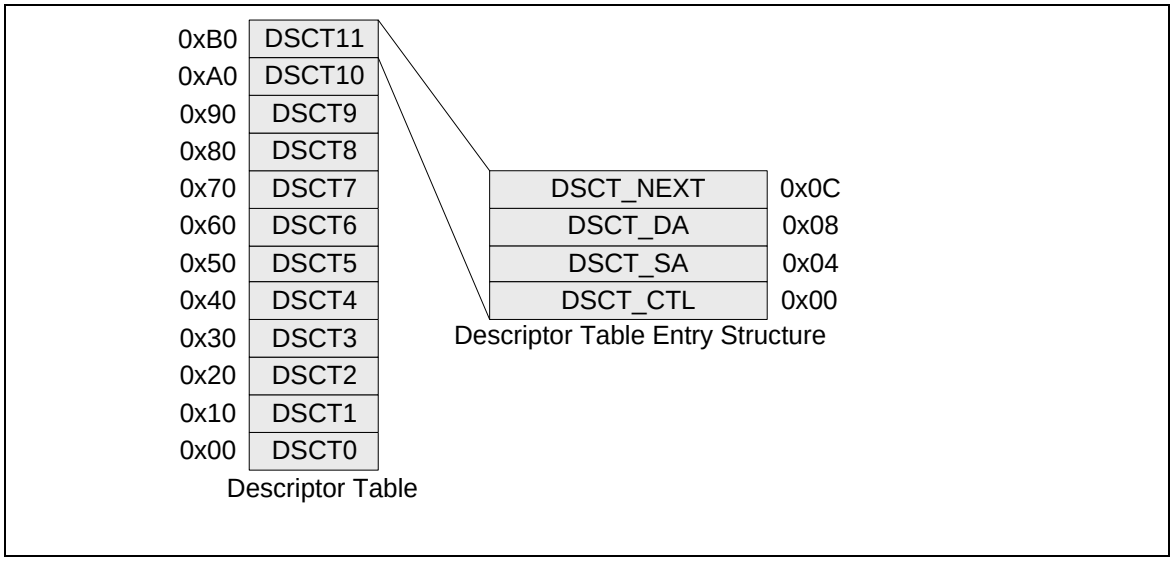


Figure 6.7-2 Descriptor Table Entry Structure

PDMA controller also supports single and burst transfer type and the request source can be from software or peripheral request, transfer between memory to memory using software request. A single transfer means that software or peripheral is ready to transfer one data (every data needs one request), and the burst transfer means that software or peripherals will transfer multiple data (multiple data only need one request).

6.7.5.1 Channel Priority

The PDMA controller supports two level channel priorities including fixed and round-robin priority. The fixed priority channel has higher priority than round-robin priority channel. If multiple channels are set as fixed or round-robin priority, the higher channel will have higher priority. The priority order is listed in Table 6-11.

PDMA_PRISET	Channel Number	Priority Setting	Arbitration Priority In Descending Order
1	11	Channel11, Fixed Priority	Highest
1	10	Channel10, Fixed Priority	---
---	---	---	---
1	0	Channel0, Fixed Priority	---
0	11	Channel11, Round-Robin Priority	---
0	10	Channel10, Round-Robin Priority	---
---	---	---	---
0	0	Channel0, Round-Robin Priority	Lowest

Table 6-11 Channel Priority Table

6.7.5.2 PDMA Operation Mode

The PDMA controller supports two operation modes including Basic mode and Scatter-Gather mode.

Basic Mode

Basic mode is used to perform one descriptor table transfer mode. This mode can be used to transfer data between memory and memory or peripherals and memory. PDMA controller operation mode can be set from OPMODE (PDMA_DSCTn_CTL[1:0], n denotes PDMA channel), default setting is in idle state (OPMODE (PDMA_DSCTn_CTL[1:0]) = 0x0) and recommend user configure the descriptor table in idle state. If operation mode is not in idle state, user re-configure channel setting may make some operation error.

User must fill the transfer count TXCNT (PDMA_DSCTn_CTL[29:16]) register and select transfer width TXWIDTH (PDMA_DSCTn_CTL[13:12]), destination address increment size DAINC (PDMA_DSCTn_CTL[11:10]), source address increment size SAINC (PDMA_DSCTn_CTL[9:8]), burst size BURSIZE (PDMA_DSCTn_CTL[6:4]) and transfer type TXTYPE (PDMA_DSCTn_CTL[2]), then the PDMA controller will perform transfer operation in transfer state after receiving request signal. Finishing this task will generate an interrupt to CPU if corresponding PDMA interrupt bit INTENn is enabled and the operation mode will be updated to idle state as shown in Figure 6.7-3. If software configures the operation mode to idle state, the PDMA controller will not perform any transfer and then clear this operation request. Finishing this task will also generate an interrupt to CPU if corresponding PDMA interrupt bit is enabled.

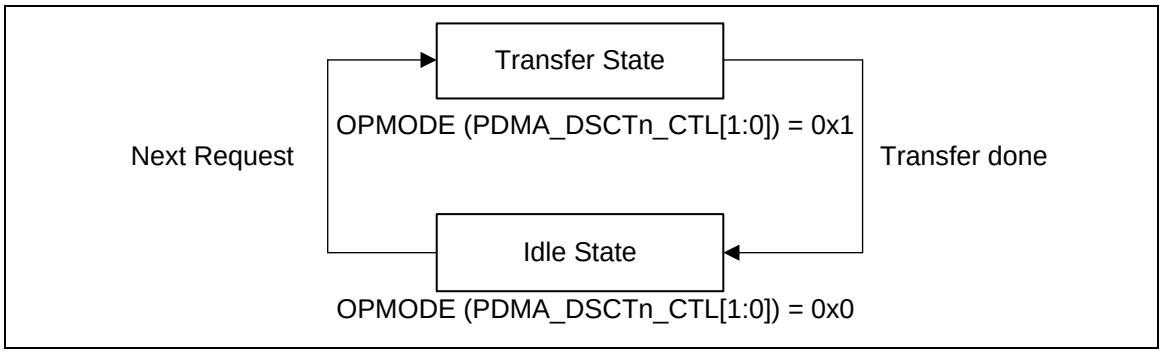


Figure 6.7-3 Basic Mode Finite State Machine

Scatter-Gather Mode

Scatter-Gather mode is a complex mode and can perform sophisticated transfer through the use of the description link list table as shown in Figure 6.7-4. Through operation mode user can perform peripheral wrapper-around, multiple PDMA task or can be used for data transfer between varied locations in system memory instead of a set of contiguous locations.

In Scatter-Gather mode, the table is just used for jumping to the next table entry. The first task will not perform any operation transfer. Finishing each task will generate an interrupt to CPU if corresponding PDMA interrupt bit is enabled and TBINTDIS (PDMA_DSCTn_CTL[7]) bit is “0” (when finishing task and TBINTDIS bit is “0”, corresponding TDIFn flag will be asserted and if this bit is “1” TDIFn will not be active).

If channel n has been triggered, and the operation mode is in Scatter-Gather mode (OPMODE (PDMA_DSCTn_CTL[1:0]) = 0x2), the hardware will load the real PDMA information task from the address generated by adding PDMA_DSCTn_NEXT (link address) and PDMA_SCATBA (base address) registers. For example, base address is 0x2000_0000 (only MSB 16bits valid in PDMA_SCATBA), current link address is 0x0000_0100 (only LSB 16bits without last two bits [1:0] valid in PDMA_DSCTn_NEXT), then next DSCT entry start address is 0x2000_0100.

Note that after each task of description link list table has been finished, the content of transfer count and operation mode in table will be cleared to 0 by hardware. To use the same link list table for transfer, user must reconfigure transfer count and operation mode.

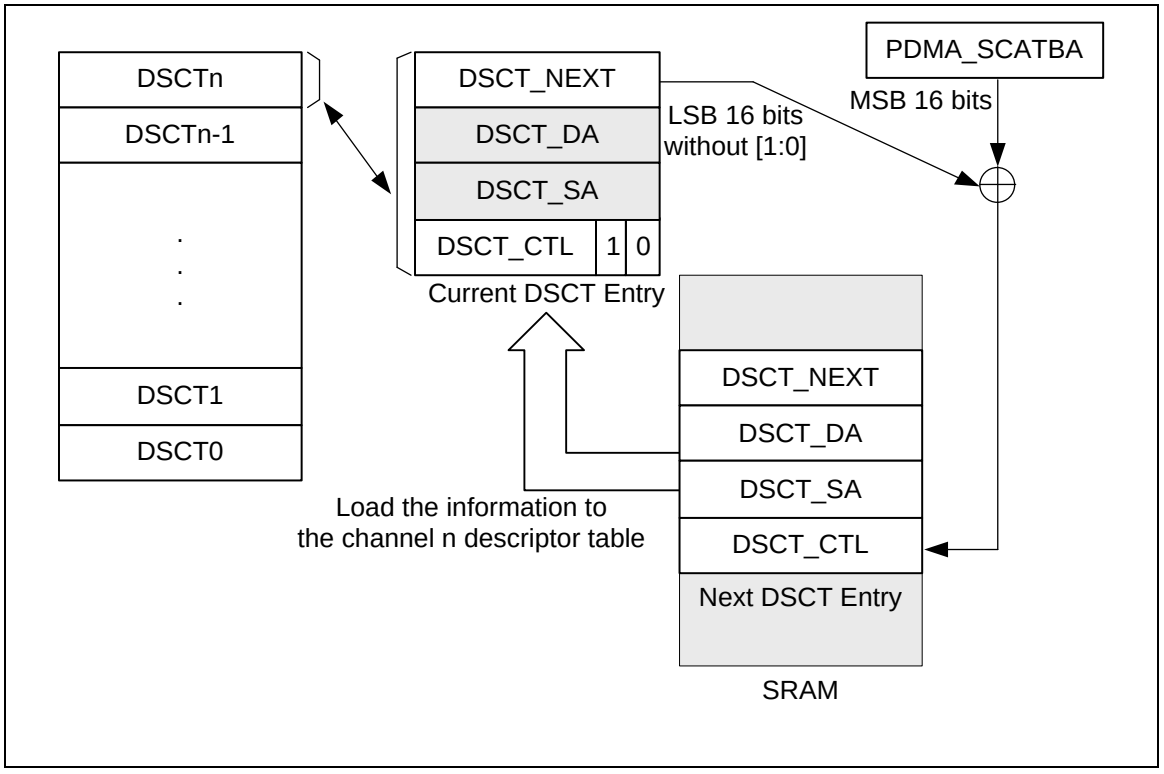


Figure 6.7-4 Descriptor Table Link List Structure

The above link list table operation is DSCT state in Scatter-Gather Mode as shown in Figure 6.7-5. When loading the information is finished, it will go to transfer state and start transfer by this

information automatically. However, if the next PDMA information is also in the Scatter-Gather mode, the hardware will catch the next PDMA information block when the current task is finished. The Scatter-Gather mode stops until the PDMA controller operation mode switch to basic mode and transfer once or directly switch to idle state.

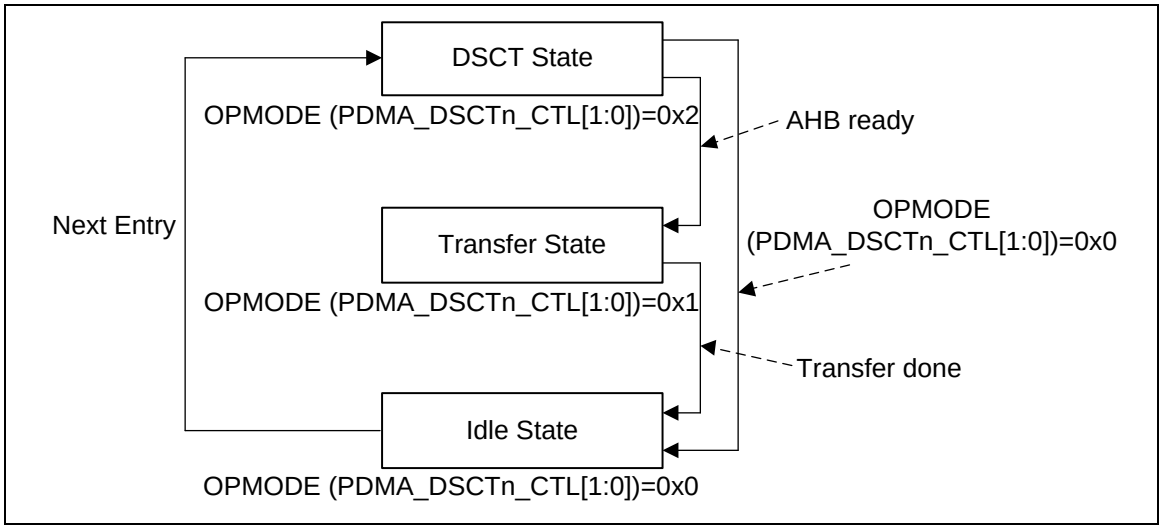


Figure 6.7-5 Scatter-Gather Mode Finite State Machine

6.7.5.3 Transfer Type

The PDMA controller supports two transfer types: single transfer type and burst transfer type, configure by setting TXTYPE (PDMA_DSCTn_CTL[2]).

When PDMA controller operated in single transfer type, each transfer data needs one request signal for one transfer, after transferred data, TXCNT (PDMA_DSCTn_CTL[29:16]) will decrease 1. Transfer will finish until the TXCNT (PDMA_DSCTn_CTL[29:16]) decrease to 0. In this mode, the BURSIZE (PDMA_DSCTn_CTL[6:4]) is not useful to control the transfer size. The BURSIZE (PDMA_DSCTn_CTL[6:4]) will be fixed as one.

For the burst transfer type, PDMA controller transfers TXCNT (PDMA_DSCTn_CTL[29:16]) of data and need only one request signal. After transferred BURSIZE (PDMA_DSCTn_CTL[6:4]) of data, TXCNT (PDMA_DSCTn_CTL[29:16]) will decrease BURSIZE number. Transfer will done until the transfer count TXCNT (PDMA_DSCTn_CTL[29:16]) decrease to 0.

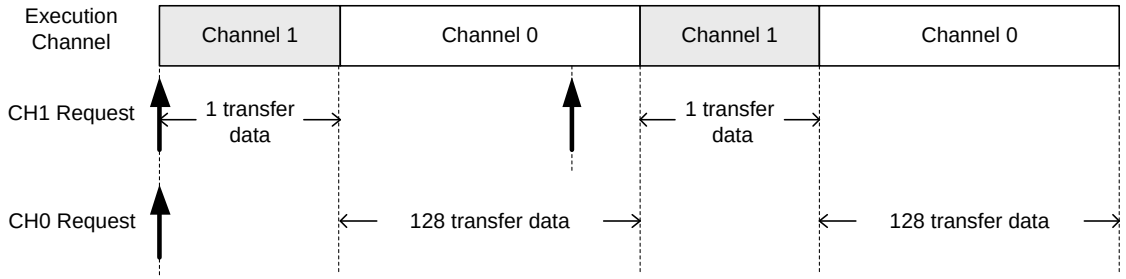
Note that burst transfer type can only be used for PDMA controller to do burst transfer between memory and memory. User must use single request type for memory-to-peripheral and peripheral-to-memory transfers.

Figure 6.7-6 shows an example about single and burst transfer type in basic mode. In this example, channel 1 uses single transfer type and TXCNT (PDMA_DSCTn_CTL[29:16]) = 128. Channel 0 uses burst transfer type, BURSIZE (PDMA_DSCTn_CTL[6:4]) = 128 and TXCNT (PDMA_DSCTn_CTL[29:16]) = 256. The operation sequence is described below:

1. Channel 0 and channel 1 get the trigger signal at the same time.
2. Channel 1 has higher priority than channel 0 by default; the PDMA controller will load the channel 1 descriptor table first and executing. But channel 1 is single transfer type, so PDMA controller will only transfer one transfer data.
3. Then, PDMA controller turns to the channel 0 and loads channel 0's descriptor table. The channel 0 is burst transfer type and the burst size selected to 128. Therefore, PDMA controller will transfer 128 transfer data.
4. When channel 0 transfers 128 data, channel 1 gets another request signal, then after

channel 0 finishes 128 transfer data, the PDMA controller will turn to channel 1 and transfer next one data.

5. After channel 1 transfers data, PDMA controller switches to low priority channel 0 to continuous next 128 data transfer. If no channel 1 request receives, PDMA will start next channel 0, 128 data transfer.
6. PDMA controller will complete transfer when channel 0 finishes data transfer 256 times, and channel 1 finishes transferring 128 times.



	CH1	CH0
TXCNT (PDMA_DSCTn_CTL[29:16])	128	256
BURSIZE (PDMA_DSCTn_CTL[6:4])	Non-useful	128
TXTYPE (PDMA_DSCTn_CTL[2])	Single Transfer	Burst Transfer

Figure 6.7-6 Example of Single Transfer Type and Burst Transfer Type in Basic Mode

Register Map

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

Register	Offset	R/W	Description	Reset Value
PDMA Base Address: **PDMA_BA = 0x4000_8000** **DSCT_CTL_BA = 0x4000_8000** **DSCT_SA_BA = 0x4000_8004** **DSCT_DA_BA = 0x4000_8008** **DSCT_NEXT_BA = 0x4000_800c** **CURSCAT_BA = 0x4000_80C0**				
PDMA_DSCTn_CTL **n = 0~11**	DSCT_CTL_BA + (0x10*n)	R/W	Descriptor Table Control Register of PDMA Channel n	0xXXXX_XXXX
PDMA_DSCTn_SA **n = 0~11**	DSCT_SA_BA + (0x10*n)	R/W	Source Address Register of PDMA Channel n	0xXXXX_XXXX
PDMA_DSCTn_DA **n = 0~11**	DSCT_DA_BA + (0x10*n)	R/W	Destination Address Register of PDMA Channel n	0xXXXX_XXXX
PDMA_DSCTn_NEXT **n = 0~11**	DSCT_NEXT_BA + (0x10*n)	R/W	First Scatter-Gather Descriptor Table Offset Address of PDMA Channel n	0xXXXX_XXXX
PDMA_CURSCATn **n = 0~11**	CURSCAT_BA + (0x04*n)	R	Current Scatter-Gather Descriptor Table Address of PDMA Channel n	0xXXXX_XXXX
PDMA_CHCTL	PDMA_BA + 0x400	R/W	PDMA Channel Control Register	0x0000_0000
PDMA_STOP	PDMA_BA + 0x404	W	PDMA Transfer Stop Control Register	0x0000_0000
PDMA_SWREQ	PDMA_BA + 0x408	W	PDMA Software Request Register	0x0000_0000
PDMA_TRGSTS	PDMA_BA + 0x40C	R	PDMA Channel Request Status Register	0x0000_0000
PDMA_PRISET	PDMA_BA + 0x410	R/W	PDMA Fixed Priority Setting Register	0x0000_0000
PDMA_PRICLR	PDMA_BA + 0x414	W	PDMA Fixed Priority Clear Register	0x0000_0000
PDMA_INTEN	PDMA_BA + 0x418	R/W	PDMA Interrupt Enable Register	0x0000_0000
PDMA_INTSTS	PDMA_BA + 0x41C	R/W	PDMA Interrupt Status Register	0x0000_0000
PDMA_ABTSTS	PDMA_BA + 0x420	R/W	PDMA Channel Read/Write Target Abort Flag Register	0x0000_0000
PDMA_TDSTS	PDMA_BA + 0x424	R/W	PDMA Channel Transfer Done Flag Register	0x0000_0000
PDMA_SCATSTS	PDMA_BA + 0x428	R/W	PDMA Scatter-Gather Table Empty Status Register	0x0000_0000
PDMA_TACTSTS	PDMA_BA + 0x42C	R	PDMA Transfer Active Flag Register	0x0000_0000
PDMA_SCATBA	PDMA_BA + 0x43C	R/W	PDMA Scatter-Gather Descriptor Table Base Address Register	0x2000_0000
PDMA_REQSEL0_3	PDMA_BA + 0x480	R/W	PDMA Request Source Select Register 0	0x1F1F_1F1F
PDMA_REQSEL4_7	PDMA_BA + 0x484	R/W	PDMA Request Source Select Register 1	0x1F1F_1F1F
PDMA_REQSEL8_11	PDMA_BA + 0x488	R/W	PDMA Request Source Select Register 2	0x1F1F_1F1F

6.7.6 Register Description

Descriptor Table Control Register (PDMA_DSCTn_CTL)

Register	Offset	R/W	Description	Reset Value
PDMA_DSCTn_CTL **n = 0~11**	DSCT_CTL_BA + (0x10*n)	R/W	Descriptor Table Control Register of PDMA Channel n	0xXXXX_XXXX

31	30	29	28	27	26	25	24
Reserved		TXCNT					
23	22	21	20	19	18	17	16
TXCNT							
15	14	13	12	11	10	9	8
Reserved		TXWIDTH		DAINC		SAINC	
7	6	5	4	3	2	1	0
TBINTDIS	BURSIZE			Reserved	TXTYPE	OPMODE	

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:16]	**TXCNT**	**Transfer Count** The TXCNT represents the required number of PDMA transfer, the real transfer count is (TXCNT + 1); The maximum transfer count is 16384 , every transfer may be byte, half-word or word that is dependent on TXWIDTH field. **Note:** When PDMA finish each transfer data, this field will be decrease immediately.
[15:14]	**Reserved**	Reserved.
[13:12]	**TXWIDTH**	**Transfer Width Selection** This field is used for transfer width. 00 = One byte (8 bit) is transferred for every operation. 01= One half-word (16 bit) is transferred for every operation. 10 = One word (32-bit) is transferred for every operation. 11 = Reserved. **Note:** The PDMA transfer source address (PDMA_DSCT_SA) and PDMA transfer destination address (PDMA_DSCT_DA) should be alignment under the TXWIDTH selection
[11:10]	**DAINC**	**Destination Address Increment** This field is used to set the destination address increment size. 11 = No increment (fixed address). Others = Increment and size is depended on TXWIDTH selection.
[9:8]	**SAINC**	**Source Address Increment** This field is used to set the source address increment size. 11 = No increment (fixed address). Others = Increment and size is depended on TXWIDTH selection.
[7]	**TBINTDIS**	**Table Interrupt Disable Bit** This field can be used to decide whether to enable table interrupt or not. If the TBINTDIS

Bits	Description	
		bit is enabled when PDMA controller finishes transfer task, it will not generates interrupt. 0 = Table interrupt Enabled. 1 = Table interrupt Disabled. **Note:** If this bit set to '1', the TEMPTYF will not be set.
[6:4]	**BURSIZE**	**Burst Size** This field is used for peripheral to determine the burst size or used for determine the re-arbitration size. 000 = 128 Transfers. 001 = 64 Transfers. 010 = 32 Transfers. 011 = 16 Transfers. 100 = 8 Transfers. 101 = 4 Transfers. 110 = 2 Transfers. 111 = 1 Transfers. **Note:** This field is only useful in burst transfer type.
[3]	**Reserved**	Reserved.
[2]	**TXTYPE**	**Transfer Type** 0 = Burst transfer type. 1 = Single transfer type.
[1:0]	**OPMODE**	**PDMA Operation Mode Selection** 00 = Idle state: Channel is stopped or this table is complete, when PDMA finish channel table task, OPMODE will be cleared to idle state automatically. 01 = Basic mode: The descriptor table only has one task. When this task is finished, the PDMA_INTSTS[n] will be asserted. 10 = Scatter-Gather mode: When operating in this mode, user must give the next descriptor table address in PDMA_DSCT_NEXT register; PDMA controller will ignore this task, then load the next task to execute. 11 = Reserved. **Note:** Before filling transfer task in the Descriptor Table, user must check if the descriptor table is complete.

Note: The n in the descriptor table represents the PDMA channel.

Start Source Address Register (PDMA_DSCTn_SA)

Register	Offset	R/W	Description	Reset Value
PDMA_DSCTn_SA **n = 0~11**	DSCT_SA_BA + (0x10*n)	R/W	Source Address Register of PDMA Channel n	0xXXXX_XXXX

31	30	29	28	27	26	25	24
SA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SA							

Bits	Description	
[31:0]	**SA**	**PDMA Transfer Source Address Register** This field indicates a 32-bit source address of PDMA controller.

Destination Address Register (PDMA_DSCTn_DA)

Register	Offset	R/W	Description	Reset Value
PDMA_DSCTn_DA **n = 0~11**	DSCT_DA_BA + (0x10*n)	R/W	Destination Address Register of PDMA Channel n	0xXXXX_XXXX

31	30	29	28	27	26	25	24
DA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DA							

Bits	Description	
[31:0]	**DA**	**PDMA Transfer Destination Address Register** This field indicates a 32-bit destination address of PDMA controller.

First Scatter-Gather Descriptor Table Offset Address (PDMA_DSCTn_NEXT)

Register	Offset	R/W	Description	Reset Value
PDMA_DSCTn_NEXT **n = 0~11**	DSCT_NEXT_BA + (0x10*n)	R/W	First Scatter-Gather Descriptor Table Offset Address of PDMA Channel n	0xXXXX_XXXX

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NEXT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NEXT						**Reserved**	

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:2]	**NEXT**	**PDMA Next Descriptor Table Offset Address Register** This field indicates the offset of next descriptor table address in system memory. The system memory based address is 0x2000_0000 (PDMA_SCATBA), if the next descriptor table is 0x2000_0100, then this field must fill in 0x0100. **Note1:** The next descriptor table address must be word boundary. **Note2:** Before filled transfer task in the descriptor table, user must check if the descriptor table is complete.
[1:0]	**Reserved**	Reserved.

Current Scatter-Gather Descriptor Table Address (PDMA_CURSCAT)

Register	Offset	R/W	Description	Reset Value
PDMA_CURSCATn **n = 0~11**	CURSCAT_BA + (0x04*n)	R	Current Scatter-Gather Descriptor Table Address of PDMA Channel n	0xXXXX_XXXX

31	30	29	28	27	26	25	24
CURADDR							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CURADDR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CURADDR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CURADDR							

Bits	Description	
[31:0]	**CURADDR**	**PDMA Current Description Address Register (Read Only)** This field indicates a 32-bit current external description address of PDMA controller. **Note:** This field is read only and only used for Scatter-Gather mode to indicate the current external description address.

Channel Control Register (PDMA_CHCTL)

Register	Offset	R/W	Description	Reset Value
PDMA_CHCTL	PDMA_BA + 0x400	R/W	PDMA Channel Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**CHEN11**	**CHEN10**	**CHEN9**	**CHEN8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CHEN7	**CHEN6**	**CHEN5**	**CHEN4**	**CHEN3**	**CHEN2**	**CHEN1**	**CHEN0**

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**CHENn**	**PDMA Channel Enable Bit** Set this bit to 1 to enable PDMAn operation. Channel cannot be active if it is not set as enabled. 0 = PDMA channel [n] Disabled. 1 = PDMA channel [n] Enabled. **Note1:** If software stops corresponding PDMA transfer by setting PDMA_STOP register, this bit will be cleared automatically after finishing current transfer. **Note2:** Software reset (writing 0xFFFF_FFFF to PDMA_STOP register) will also clear this bit.

Note: The n in the descriptor table represents the PDMA channel.

PDMA Transfer Stop Control Register (PDMA_STOP)

Register	Offset	R/W	Description	Reset Value
PDMA_STOP	PDMA_BA + 0x404	W	PDMA Transfer Stop Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**STOP11**	**STOP10**	**STOP9**	**STOP8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
STOP7	**STOP6**	**STOP5**	**STOP4**	**STOP3**	**STOP2**	**STOP1**	**STOP0**

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**STOPn**	**PDMA Transfer Stop Control Register (Write Only)** User can stop the PDMA transfer by STOPn bit field or by software reset (writing '0xFFFF_FFFF' to PDMA_STOP register). **By bit field:** 0 = No effect. 1 = Stop PDMA transfer[n]. When software set PDMA_STOP bit, the operation will finish the on-going transfer channel and then clear the channel enable bit (PDMA_CHCTL [CHEN]) and request active flag. **By write 0xFFFF_FFFF to PDMA_STOP:** Setting all PDMA_STOP bit to "1" will generate software reset to reset internal state machine (the DSCT will not be reset). When software reset, the operation will be stopped imminently that include the on-going transfer and the channel enable bit (PDMA_CHCTL [CHEN]) and request active flag will be cleared to '0'. **Note1:** User can read channel enable bit to know if the on-going transfer is finished.

Note: The n in the descriptor table represents the PDMA channel.

PDMA Software Request Register (PDMA_SWREQ)

Register	Offset	R/W	Description	Reset Value
PDMA_SWREQ	PDMA_BA + 0x408	W	PDMA Software Request Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**SWREQ11**	**SWREQ10**	**SWREQ9**	**SWREQ8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SWREQ7	**SWREQ6**	**SWREQ5**	**SWREQ4**	**SWREQ3**	**SWREQ2**	**SWREQ1**	**SWREQ0**

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**SWREQn**	**PDMA Software Request Register (Write Only)** Set this bit to 1 to generate a software request to PDMA [n]. 0 = No effect. 1 = Generate a software request. **Note1:** User can read PDMA_TRGSTS register to know which channel is on active. Active flag may be triggered by software request or peripheral request. **Note2:** If user does not enable corresponding PDMA channel, the software request will be ignored.

Note: The n in the descriptor table represents the PDMA channel.

PDMA Channel Request Status Register (PDMA_TRGSTS)

Register	Offset	R/W	Description	Reset Value
PDMA_TRGSTS	PDMA_BA + 0x40C	R	PDMA Channel Request Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				REQSTS11	REQSTS10	REQSTS9	REQSTS8
7	6	5	4	3	2	1	0
REQSTS7	REQSTS6	REQSTS5	REQSTS4	REQSTS3	REQSTS2	REQSTS1	REQSTS0

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**REQSTSn**	**PDMA Channel Request Status (Read Only)** This flag indicates whether channel[n] have a request or not, no matter request from software or peripheral. When PDMA controller finishes channel transfer, this bit will be cleared automatically. 0 = PDMA Channel n has no request. 1 = PDMA Channel n has a request. **Note1:** If software stops corresponding PDMA transfer by setting PDMA_STOP register, this bit will be cleared automatically after finishing current transfer. **Note2:** Software reset (writing 0xFFFF_FFFF to PDMA_STOP register) will also clear this bit.

Note: The n in the descriptor table represents the PDMA channel.

PDMA Fixed Priority Setting Register (PDMA_PRISET)

Register	Offset	R/W	Description	Reset Value
PDMA_PRISET	PDMA_BA + 0x410	R/W	PDMA Fixed Priority Setting Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**FPRISET11**	**FPRISET10**	**FPRISET9**	**FPRISET8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FPRISET7	**FPRISET6**	**FPRISET5**	**FPRISET4**	**FPRISET3**	**FPRISET2**	**FPRISET1**	**FPRISET0**

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**FPRISETn**	**PDMA Fixed Priority Setting Register** Set this bit to 1 to enable fixed priority level. **Write Operation:** 0 = No effect. 1 = Set PDMA channel [n] to fixed priority channel. **Read Operation:** 0 = Corresponding PDMA channel is round-robin priority. 1 = Corresponding PDMA channel is fixed priority. **Note1:** This field only set to fixed priority, clear fixed priority use PDMA_PRICLR register.

Note: The n in the descriptor table represents the PDMA channel.

PDMA Fix Priority Clear Register (PDMA_PRICLR)

Register	Offset	R/W	Description	Reset Value
PDMA_PRICLR	PDMA_BA + 0x414	W	PDMA Fixed Priority Clear Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**FPRICLR11**	**FPRICLR10**	**FPRICLR9**	**FPRICLR8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FPRICLR7	**FPRICLR6**	**FPRICLR5**	**FPRICLR4**	**FPRICLR3**	**FPRICLR2**	**FPRICLR1**	**FPRICLR0**

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**FPRICLRn**	**PDMA Fixed Priority Clear Register (Write Only)** Set this bit to 1 to clear fixed priority level. 0 = No effect. 1 = Clear PDMA channel [n] fixed priority setting. **Note1:** User can read PDMA_PRISET register to know the channel priority.

Note: The n in the descriptor table represents the PDMA channel.

PDMA Interrupt Enable Register (PDMA_INTEN)

Register	Offset	R/W	Description	Reset Value
PDMA_INTEN	PDMA_BA + 0x418	R/W	PDMA Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**INTEN11**	**INTEN10**	**INTEN9**	**INTEN8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INTEN7	**INTEN6**	**INTEN5**	**INTEN4**	**INTEN3**	**INTEN2**	**INTEN1**	**INTEN0**

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**INTENn**	**PDMA Interrupt Enable Register** This field is used for enabling PDMA channel[n] interrupt. 0 = PDMA channel n interrupt Disabled. 1 = PDMA channel n interrupt Enabled.

Note: The n in the descriptor table represents the PDMA channel.

PDMA Interrupt Status Register (PDMA_INTSTS)

Register	Offset	R/W	Description	Reset Value
PDMA_INTSTS	PDMA_BA + 0x41C	R/W	PDMA Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
REQTOF7	REQTOF6	REQTOF5	REQTOF4	REQTOF3	REQTOF2	REQTOF1	REQTOF0
7	6	5	4	3	2	1	0
Reserved					TEIF	TDIF	ABTIF

Bits	Description	
[31:8]	**Reserved**	Reserved.
[29:3]	**Reserved**	Reserved.
[2]	**TEIF**	**Table Empty Interrupt Flag (Read Only)** This bit indicates that PDMA controller has finished each table transmission and the operation is Stop mode. User can read TEIF register to indicate which channel finished transfer. 0 = PDMA channel transfer is not finished. 1 = PDMA channel transfer is finished and the operation is in idle state.
[1]	**TDIF**	**Transfer Done Interrupt Flag (Read Only)** This bit indicates that PDMA controller has finished transmission; User can read PDMA_TDSTS register to indicate which channel finished transfer. 0 = Not finished yet. 1 = PDMA channel has finished transmission.
[0]	**ABTIF**	**PDMA Read/Write Target Abort Interrupt Flag (Read-only)** This bit indicates that PDMA has target abort error; Software can read PDMA_ABTSTS register to find which channel has target abort error. 0 = No AHB bus ERROR response received. 1 = AHB bus ERROR response received.

PDMA Channel Read/Write Target Abort Flag Register (PDMA_ABTSTS)

Register	Offset	R/W	Description	Reset Value
PDMA_ABTSTS	PDMA_BA + 0x420	R/W	PDMA Channel Read/Write Target Abort Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ABTIF7	**ABTIF6**	**ABTIF5**	**ABTIF4**	**ABTIF3**	**ABTIF2**	**ABTIF1**	**ABTIF0**

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**ABTIFn**	**PDMA Read/Write Target Abort Interrupt Status Flag** This bit indicates which PDMA controller has target abort error; User can write 1 to clear these bits. 0 = No AHB bus ERROR response received when channel n transfer. 1 = AHB bus ERROR response received when channel n transfer.

Note: The n in the descriptor table represents the PDMA channel.

PDMA Channel Transfer Done Flag Register (PDMA_TDSTS)

Register	Offset	R/W	Description	Reset Value
PDMA_TDSTS	PDMA_BA + 0x424	R/W	PDMA Channel Transfer Done Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**TDIF11**	**TDIF10**	**TDIF9**	**TDIF8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TDIF7	**TDIF6**	**TDIF5**	**TDIF4**	**TDIF3**	**TDIF2**	**TDIF1**	**TDIF0**

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**TDIFn**	**Transfer Done Flag Register** This bit indicates whether PDMA controller channel transfer has been finished or not, user can write 1 to clear these bits. 0 = PDMA channel transfer has not finished. 1 = PDMA channel has finished transmission.

Note: The n in the descriptor table represents the PDMA channel.

PDMA Scatter-Gather Table Empty Status Register (PDMA_SCATSTS)

Register	Offset	R/W	Description	Reset Value
PDMA_SCATSTS	PDMA_BA + 0x428	R/W	PDMA Scatter-Gather Table Empty Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				TEMPTYF11	TEMPTYF10	TEMPTYF9	TEMPTYF8
7	6	5	4	3	2	1	0
TEMPTYF7	TEMPTYF6	TEMPTYF5	TEMPTYF4	TEMPTYF3	TEMPTYF2	TEMPTYF1	TEMPTYF0

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**TEMPTYFn**	**Scatter-gather Table Empty Flag Register** This bit indicates which PDMA channel n Scatter Gather table is empty when SWREQn (PDMA_SWREQ[11:0]) set to high or channel has finished transmission and the operation mode is Stop mode. User can write 1 to clear these bits. 0 = PDMA channel scatter-gather table is not empty. 1 = PDMA channel scatter-gather table is empty and PDMA SWREQ has be set.

Note: The n in the descriptor table represents the PDMA channel.

PDMA Transfer Active Flag Register (PDMA_TACTSTS)

Register	Offset	R/W	Description	Reset Value
PDMA_TACTSTS	PDMA_BA + 0x42C	R	PDMA Transfer Active Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				TXACTF11	TXACTF10	TXACTF9	TXACTF8
7	6	5	4	3	2	1	0
TXACTF7	TXACTF6	TXACTF5	TXACTF4	TXACTF3	TXACTF2	TXACTF1	TXACTF0

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**TXACTFn**	**Transfer on Active Flag Register (Read Only)** This bit indicates which PDMA channel is in active. 0 = PDMA channel is not finished. 1 = PDMA channel is active.

Note: The n in the descriptor table represents the PDMA channel.

PDMA Scatter-Gather Descriptor Table Base Address Register (PDMA_SCATBA)

Register	Offset	R/W	Description	Reset Value
PDMA_SCATBA	PDMA_BA + 0x43C	R/W	PDMA Scatter-Gather Descriptor Table Base Address Register	0x2000_0000

31	30	29	28	27	26	25	24
SCATBA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SCATBA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:16]	**SCATBA**	**PDMA Scatter-gather Descriptor Table Address Register** In Scatter-Gather mode, this is the base address for calculating the next link - list address. The next link address equation is Next Link Address = PDMA_SCATBA + PDMA_DSCT_NEXT. **Note:** Only useful in Scatter-Gather mode.
[15:0]	**Reserved**	Reserved.

PDMA Request Source Select Register 0 (PDMA_REQSEL0_3)

Register	Offset	R/W	Description	Reset Value
PDMA_REQSEL0_3	PDMA_BA + 0x480	R/W	PDMA Request Source Select Register 0	0x1F1F_1F1F

31	30	29	28	27	26	25	24
Reserved			REQSRC3				
23	22	21	20	19	18	17	16
Reserved			REQSRC2				
15	14	13	12	11	10	9	8
Reserved			REQSRC1				
7	6	5	4	3	2	1	0
Reserved			REQSRC0				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**REQSRC3**	**Channel 3 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 3. User can configure the peripheral setting by REQSRC3. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[23:21]	**Reserved**	Reserved.
[20:16]	**REQSRC2**	**Channel 2 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 2. User can configure the peripheral setting by REQSRC2. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[15:13]	**Reserved**	Reserved.
[12:8]	**REQSRC1**	**Channel 1 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 1. User can configure the peripheral setting by REQSRC1. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[7:5]	**Reserved**	Reserved.
[4:0]	**REQSRC0**	**Channel 0 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 0. User can configure the peripheral by setting REQSRC0. 1 = Channel connects to SPI0_TX. 2 = Channel connects to SPI1_TX. 3 = Channel connects to SPI2_TX. 4 = Channel connects to UART0_TX. 5 = Channel connects to UART1_TX. 6 = Channel connects to UART2_TX.

Bits	Description	
		7 = Channel connects to UART3_TX. 8 = Channel connects to DAC_TX. 9 = Channel connects to ADC_RX. 11 = Channel connects to PWM0_P1_RX. 12 = Channel connects to PWM0_P2_RX. 13 = Channel connects to PWM0_P3_RX. 14 = Channel connects to PWM1_P1_RX. 15 = Channel connects to PWM1_P2_RX. 16 = Channel connects to PWM1_P3_RX. 17 = Channel connects to SPI0_RX. 18 = Channel connects to SPI1_RX. 19 = Channel connects to SPI2_RX. 20 = Channel connects to UART0_RX. 21 = Channel connects to UART1_RX. 22 = Channel connects to UART2_RX. 23 = Channel connects to UART3_RX. 31 = Disable PDMA. Others = Reserved. **Note 1:** A peripheral can't assign to two channels at the same time. **Note 2:** This field is useless when transfer between memory and memory.

PDMA Request Source Select Register 1 (PDMA_REQSEL4_7)

Register	Offset	R/W	Description	Reset Value
PDMA_REQSEL4_7	PDMA_BA + 0x484	R/W	PDMA Request Source Select Register 1	0x1F1F_1F1F

31	30	29	28	27	26	25	24
Reserved			REQSRC7				
23	22	21	20	19	18	17	16
Reserved			REQSRC6				
15	14	13	12	11	10	9	8
Reserved			REQSRC5				
7	6	5	4	3	2	1	0
Reserved			REQSRC4				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**REQSRC7**	**Channel 7 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 7. User can configure the peripheral setting by REQSRC7. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[23:21]	**Reserved**	Reserved.
[20:16]	**REQSRC6**	**Channel 6 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 6. User can configure the peripheral setting by REQSRC6. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[15:13]	**Reserved**	Reserved.
[12:8]	**REQSRC5**	**Channel 5 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 5. User can configure the peripheral setting by REQSRC5. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[7:5]	**Reserved**	Reserved.
[4:0]	**REQSRC4**	**Channel 4 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 4. User can configure the peripheral setting by REQSRC4. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.

PDMA Request Source Select Register 2 (PDMA_REQSEL8_11)

Register	Offset	R/W	Description	Reset Value
PDMA_REQSEL8_11	PDMA_BA + 0x488	R/W	PDMA Request Source Select Register 2	0x1F1F_1F1F

31	30	29	28	27	26	25	24
Reserved			REQSRC11				
23	22	21	20	19	18	17	16
Reserved			REQSRC10				
15	14	13	12	11	10	9	8
Reserved			REQSRC9				
7	6	5	4	3	2	1	0
Reserved			REQSRC8				

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28:24]	**REQSRC11**	**Channel 11 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 11. User can configure the peripheral setting by REQSRC11. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[23:21]	**Reserved**	Reserved.
[20:16]	**REQSRC10**	**Channel 10 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 10. User can configure the peripheral setting by REQSRC10. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[15:13]	**Reserved**	Reserved.
[12:8]	**REQSRC9**	**Channel 9 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 9. User can configure the peripheral setting by REQSRC9. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.
[7:5]	**Reserved**	Reserved.
[4:0]	**REQSRC8**	**Channel 8 Request Source Selection** This filed defines which peripheral is connected to PDMA channel 8. User can configure the peripheral setting by REQSRC8. **Note:** The channel configuration is the same as REQSRC0 field. Please refer to the explanation of REQSRC0.

6.8 Timer Controller (TMR)

6.8.1 Overview

The Timer controller includes four 32-bit timers, Timer0 ~ Timer3, allowing user to easily implement a timer control for applications. The timer can perform functions, such as frequency measurement, delay timing, clock generation, and event counting by external input pins, and interval measurement by external capture pins.

6.8.2 Features

- Four sets of 32-bit timers 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 (TIMERx_CNT[23:0])
- Supports event counting function
- 24-bit capture value is readable through CAPDAT (TIMERx_CAP[23:0])
- Supports external capture pin event for interval measurement
- Supports external capture pin event to reset 24-bit up counter
- Supports chip wake-up from Idle/Power-down mode if a timer interrupt signal is generated
- Support Timer0 time-out interrupt signal to trigger Touch-Key scan
- Support Timer0 ~ Timer3 time-out interrupt signal or capture interrupt signal to trigger PWM, EADC and DAC function

6.8.3 Block Diagram

The Timer Controller block diagram and clock control are shown as follows.

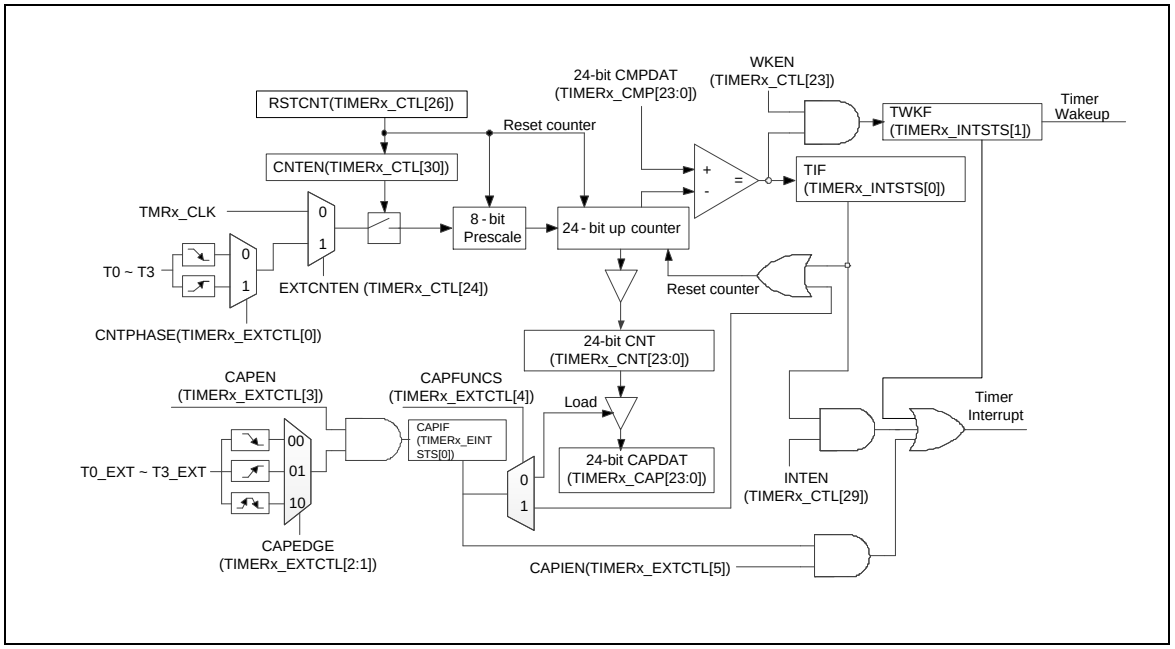


Figure 6.8-1 Timer Controller Block Diagram

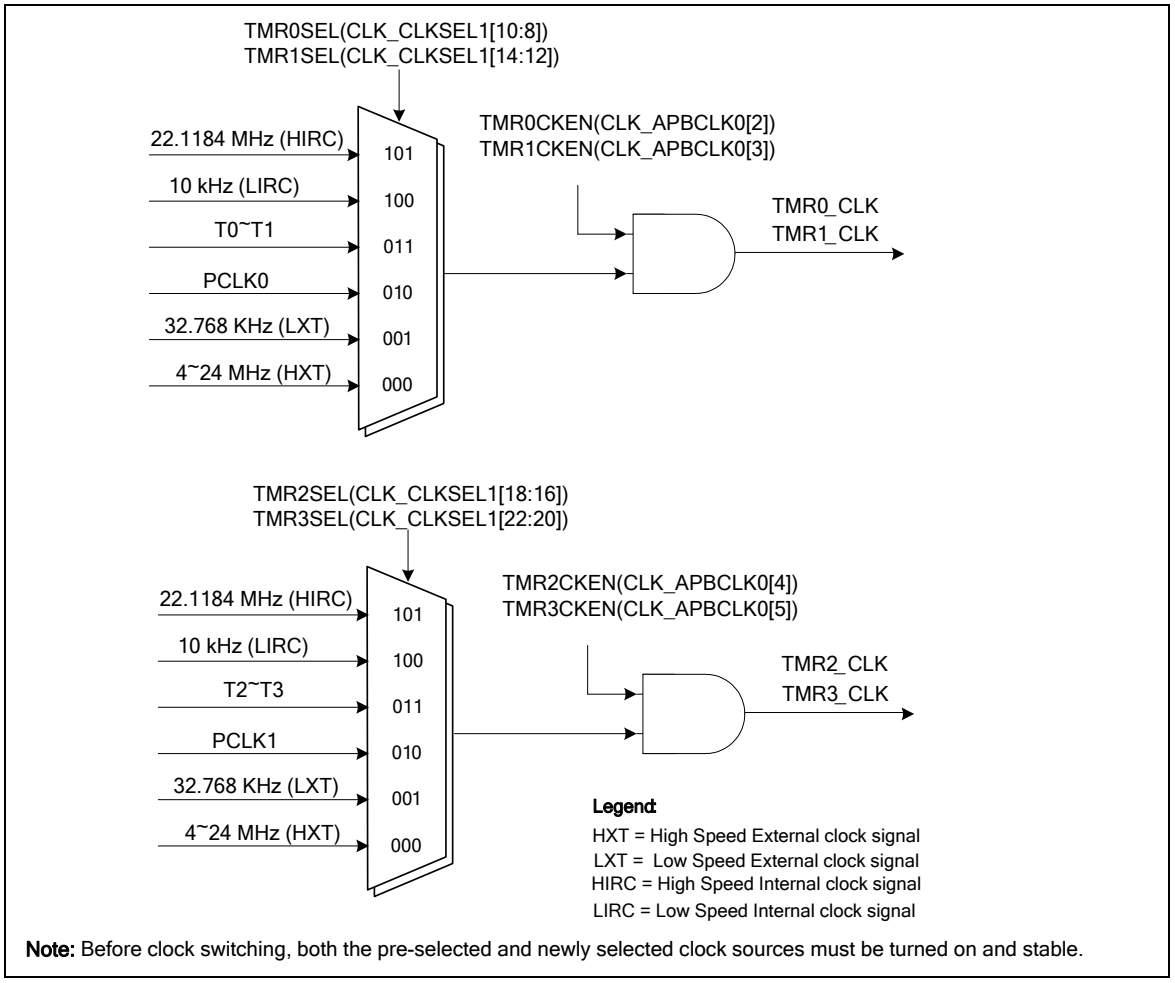


Figure 6.8-2 Clock Source of Timer Controller

6.8.4 Basic Configuration

The peripheral clock source of Tiimer0 ~ Timer3 can be enabled in TMRxCKEN (CLK_APBCLK0[5:2]) and selected as different frequency in TMR0SEL (CLK_CLKSEL1[10:8]) for Timer0, TMR1SEL (CLK_CLKSEL1[14:12]) for Timer1, TMR2SEL (CLK_CLKSEL1[18:16]) for Timer2 and TMR3SEL (CLK_CLKSEL1[22:20]) for Timer3.

6.8.5 Functional Description

6.8.5.1 Timer Interrupt Flag

Timer controller supports two interrupt flags; one is TIF (TIMERx_INTSTS[0]) and its set while timer counter value CNT (TIMERx_CNT[23:0]) matches the timer compared value CMPDAT (TIMERx_CMP[23:0]), the other is CAPIF (TIMERx_EINTSTS[0]) and its set when the transition on the Tx_EXT pin associated CAPEDGE (TIMERx_EXTCTL[2:1]) setting.

6.8.5.2 Timer Counting Mode

Timer controller provides four timer counting modes: one-shot, periodic, toggle-output and continuous counting operation modes:

6.8.5.3 One–shot Mode

If timer controller is configured at one-shot mode (TIMERx_CTL[28:27] is 00) and CNTEN (TIMERx_CTL[30]) is set, the timer counter starts up counting. Once the CNT

(TIMERx_CNT[23:0]) value reaches CMPDAT (TIMERx_CMP[23:0]) value, the TIF (TIMERx_INTSTS[0]) 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 (TIMERx_CTL[29]) is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU also.

6.8.5.4 Periodic Mode

If timer controller is configured at periodic mode (TIMERx_CTL[28:27] is 01) and CNTEN (TIMERx_CTL[30]) is set, the timer counter starts up counting. Once the CNT (TIMERx_CNT[23:0]) value reaches CMPDAT (TIMERx_CMP[23:0]) value, the TIF (TIMERx_INTSTS[0]) 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 (TIMERx_CTL[29]) bit is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU also. In this mode, timer controller operates counting and compares with CMPDAT value periodically until the CNTEN bit is cleared by user.

6.8.5.5 Toggle-Output Mode

If timer controller is configured at toggle-output mode (TIMERx_CTL[28:27] is 10) and CNTEN (TIMERx_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 T0 ~ T3 or T0_EXT ~ T3_EXT pin depending on TGLPINSEL (TIMERx_CTL[22]) setting to output signal while specify TIF (TIMERx_INTSTS[0]) is set. Thus, the toggle-output signal on T0 ~ T3 or T0_EXT ~ T3_EXT pin is high and changing back and forth with 50% duty cycle.

6.8.5.6 Continuous Counting Mode

If timer controller is configured at continuous counting mode (TIMERx_CTL[28:27] is 11) and CNTEN (TIMERx_CTL[30]) is set, the timer counter starts up counting. Once the CNT (TIMERx_CNT[23:0]) value reaches CMPDAT (TIMERx_CMP[23:0]) value, the TIF (TIMERx_INTSTS[0]) will be set to 1 and CNT value keeps up counting. In the meantime, if the INTEN (TIMERx_CTL[29]) is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU also. 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 TIF will set to 1 when CNT value is equal to 80, timer counter is kept counting and CNT value will not goes back to 0, it continues to count 81, 82, 83,⋯ to 2^{24} -1, 0, 1, 2, 3, ⋯ to 2^{24} -1 again and again. Next, if user programs CMPDAT value as 200 and clears TIF, the TIF will set to 1 again when CNT value reaches to 200. At last, user programs CMPDAT as 500 and clears TIF, the TIF will set to 1 again when CNT value reaches to 500.

In this mode, the timer counting is continuous. So, this operation mode is called as continuous counting mode.

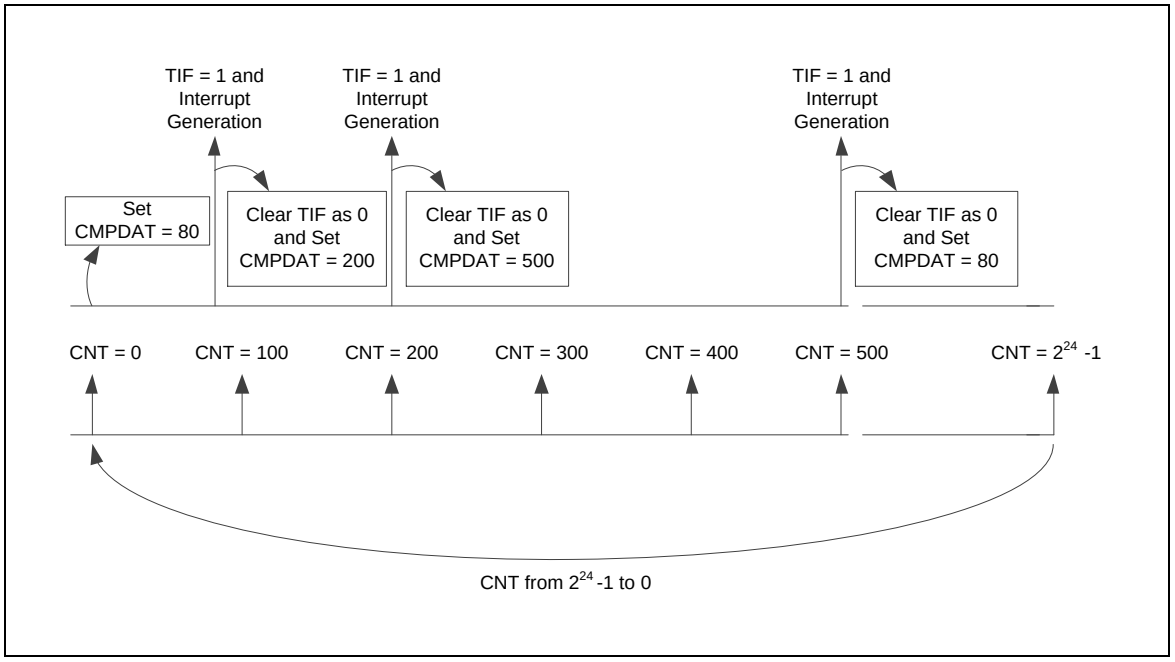


Figure 6.8-3 Continuous Counting Mode

6.8.5.7 Event Counting Mode

Timer controller also provides an application which can count the input event from Tx (x= 0~3) pin and the number of event will reflect to CNT (TIMERx_CNT[23:0]) value. It is also called as event counting function. In this function, EXTCNTEN (TIMERx_CTL[24]) should be set and the timer peripheral clock source should be set as PCLKx (x= 0~1).

User can enable or disable Tx pin de-bounce circuit by setting CNTDBEN (TIMERx_EXTCTL[7]). The input event frequency should be less than 1/3 PCLKx if Tx pin de-bounce disabled or less than 1/8 PCLKx if Tx pin de-bounce enabled to assure the returned CNT value is correct, and user can also select edge detection phase of Tx pin by setting CNTPHASE (TIMERx_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 (TIMERx_CNT[23:0]) for Tx pin.

6.8.5.8 External Capture Mode

The event capture function is used to load CNT (TIMERx_CNT[23:0]) value to CAPDAT (TIMERx_CAP[23:0]) value while edge transition detected on Tx_EXT (x= 0~3) pin. In this mode, CAPFUNCS (TIMERx_EXTCTL[4]) should be as 0 for select Tx_EXT transition is using to trigger event capture function and the timer peripheral clock source should be set as PCLKx (x= 0~1).

User can enable or disable Tx_EXT pin de-bounce circuit by setting CAPDBEN (TIMERx_EXTCTL[6]). The transition frequency of Tx_EXT pin should be less than 1/3 PCLKx if Tx_EXT pin de-bounce disabled or less than 1/8 PCLKx if Tx_EXT pin de-bounce enabled to assure the capture function can be work normally, and user can also select edge transition detection of Tx_EXT pin by setting CAPEDGE (TIMERx_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 Tx_EXT pin is detected.

Users must consider the Timer will keep register TIMERx_CAP unchanged and drop the new capture value, if the CPU does not clear the CAPIF status.

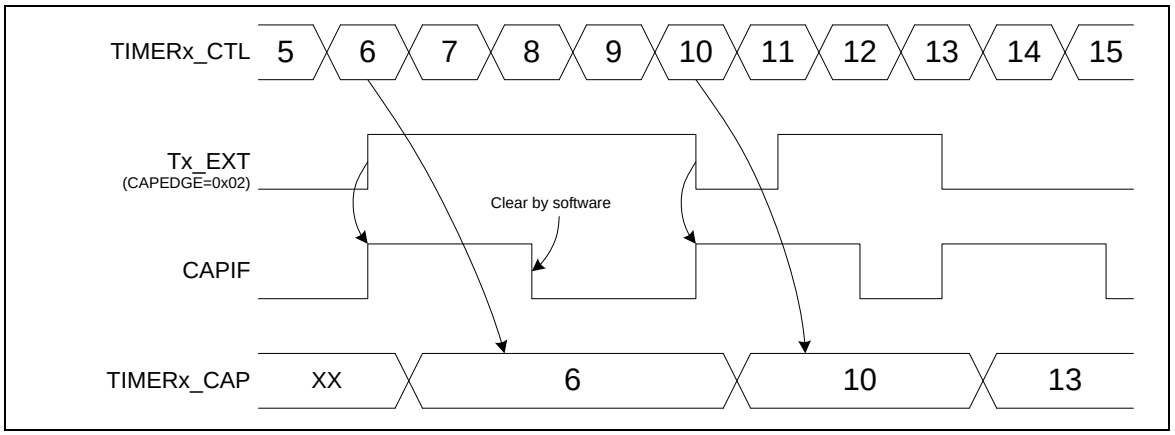


Figure 5.20-4 External Capture Mode

6.8.5.9 External Reset Counter Mode

Timer controller also provides reset counter function to reset CNT (TIMERx_CNT[23:0]) value while edge transition detected on Tx_EXT (x= 0~3). In this mode, most the settings are the same as event capture mode except CAPFUNCS (TIMERx_EXTCTL[4]) should be as 1 for select Tx_EXT transition is using to trigger reset counter value.

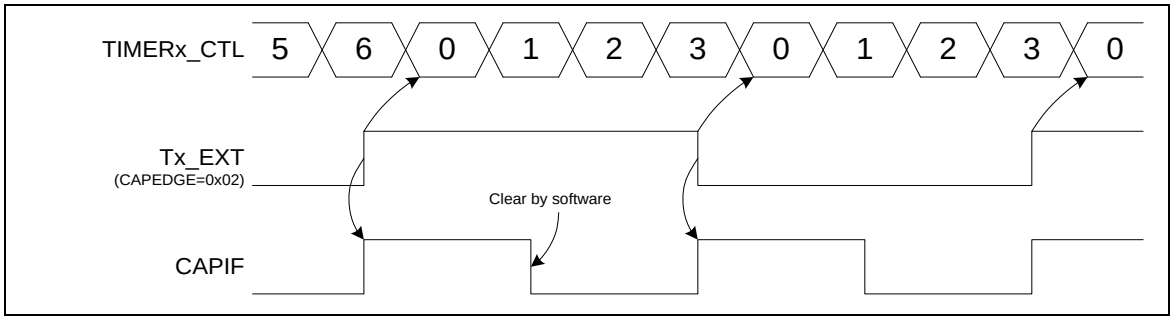


Figure 5.20-5 External Reset Counter Mode

6.8.5.10 Timer Trigger Function

Timer controller provides timer time-out interrupt or capture interrupt to trigger PWM, DAC and EADC. If TRGSSEL (TIMERx_CTL[18]) is 0, time-out interrupt signal is used to trigger PWM, EADC and DAC. If TRGSSEL (TIMERx_CTL[18]) is 1, capture interrupt signal is used to trigger PWM, EADC and DAC.

When the TRGPWM (TIMERx_CTL[19]) is set, if the timer interrupt signal is generated, the timer controller will generate a trigger pulse as PWM external clock source.

When the TRGDAC (TIMERx_CTL[20]) is set, if the timer interrupt signal is generated, the timer controller will trigger DAC to start converter.

When the TRGEADC (TIMERx_CTL[21]) is set, if the timer interrupt signal is generated, the timer controller will trigger EADC to start converter.

Timer controller also provides timer time-out interrupt to trigger Touch-Key scan only in Timer0. When the WKTKEN (TIMER0_CTL[17]) is set and chip in Power-down mode, if the Timer0 time-out interrupt signal is generated, the Timer0 controller will trigger Touck-Key scan also. In this function, the Timer0 module clock source should be set as 10 kHz internal low speed RC oscillator (LIRC) or 32.768 kHz external low speed crystal oscillator (LXT).

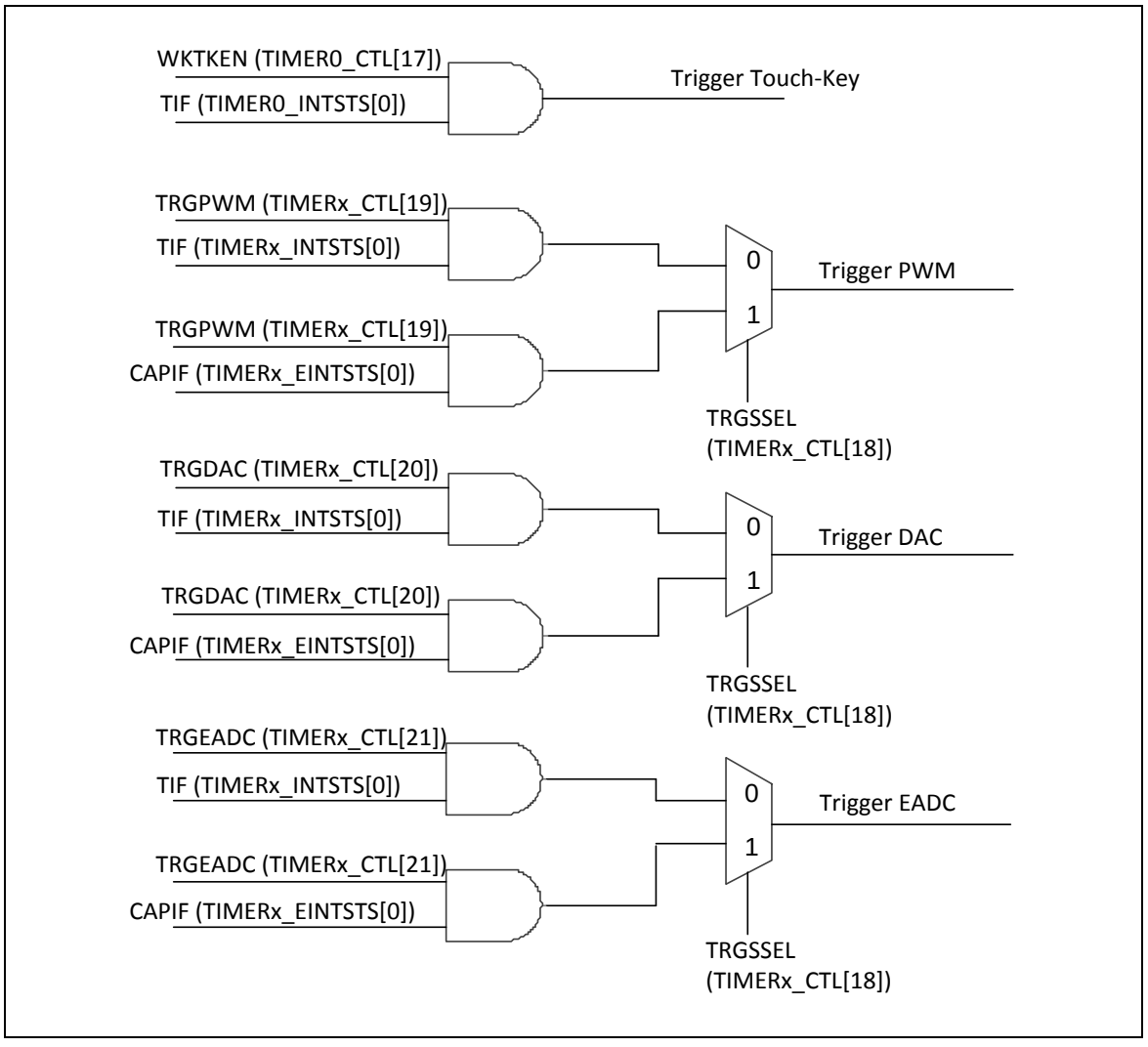


Figure 6.8-6 Internal Timer Trigger

6.8.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
TMR Base Address: **TMR_BA01 = 0x4005_0000** **TMR_BA23 = 0x4005_1000**				
TIMER0_CTL	TMR_BA01+0x00	R/W	Timer0 Control and Status Register	0x0000_0005
TIMER0_CMP	TMR_BA01+0x04	R/W	Timer0 Compare Register	0x0000_0000
TIMER0_INTSTS	TMR_BA01+0x08	R/W	Timer0 Interrupt Status Register	0x0000_0000
TIMER0_CNT	TMR_BA01+0x0C	R	Timer0 Data Register	0x0000_0000
TIMER0_CAP	TMR_BA01+0x10	R	Timer0 Capture Data Register	0x0000_0000
TIMER0_EXTCTL	TMR_BA01+0x14	R/W	Timer0 External Control Register	0x0000_0000
TIMER0_EINTSTS	TMR_BA01+0x18	R/W	Timer0 External Interrupt Status Register	0x0000_0000
TIMER1_CTL	TMR_BA01+0x20	R/W	Timer1 Control and Status Register	0x0000_0005
TIMER1_CMP	TMR_BA01+0x24	R/W	Timer1 Compare Register	0x0000_0000
TIMER1_INTSTS	TMR_BA01+0x28	R/W	Timer1 Interrupt Status Register	0x0000_0000
TIMER1_CNT	TMR_BA01+0x2C	R	Timer1 Data Register	0x0000_0000
TIMER1_CAP	TMR_BA01+0x30	R	Timer1 Capture Data Register	0x0000_0000
TIMER1_EXTCTL	TMR_BA01+0x34	R/W	Timer1 External Control Register	0x0000_0000
TIMER1_EINTSTS	TMR_BA01+0x38	R/W	Timer1 External Interrupt Status Register	0x0000_0000
TIMER2_CTL	TMR_BA23+0x00	R/W	Timer2 Control and Status Register	0x0000_0005
TIMER2_CMP	TMR_BA23+0x04	R/W	Timer2 Compare Register	0x0000_0000
TIMER2_INTSTS	TMR_BA23+0x08	R/W	Timer2 Interrupt Status Register	0x0000_0000
TIMER2_CNT	TMR_BA23+0x0C	R	Timer2 Data Register	0x0000_0000
TIMER2_CAP	TMR_BA23+0x10	R	Timer2 Capture Data Register	0x0000_0000
TIMER2_EXTCTL	TMR_BA23+0x14	R/W	Timer2 External Control Register	0x0000_0000
TIMER2_EINTSTS	TMR_BA23+0x18	R/W	Timer2 External Interrupt Status Register	0x0000_0000
TIMER3_CTL	TMR_BA23+0x20	R/W	Timer3 Control and Status Register	0x0000_0005
TIMER3_CMP	TMR_BA23+0x24	R/W	Timer3 Compare Register	0x0000_0000

TIMER3_INTSTS	TMR_BA23+0x28	R/W	Timer3 Interrupt Status Register	0x0000_0000
TIMER3_CNT	TMR_BA23+0x2C	R	Timer3 Data Register	0x0000_0000
TIMER3_CAP	TMR_BA23+0x30	R	Timer3 Capture Data Register	0x0000_0000
TIMER3_EXTCTL	TMR_BA23+0x34	R/W	Timer3 External Control Register	0x0000_0000
TIMER3_EINTSTS	TMR_BA23+0x38	R/W	Timer3 External Interrupt Status Register	0x0000_0000

6.8.7 Register Description

Timer Control Register (TIMERx_CTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_CTL	TMR_BA01+0x00	R/W	Timer0 Control and Status Register	0x0000_0005
TIMER1_CTL	TMR_BA01+0x20	R/W	Timer1 Control and Status Register	0x0000_0005
TIMER2_CTL	TMR_BA23+0x00	R/W	Timer2 Control and Status Register	0x0000_0005
TIMER3_CTL	TMR_BA23+0x20	R/W	Timer3 Control and Status Register	0x0000_0005

31	30	29	28	27	26	25	24
ICEDEBUG	CNTEN	INTEN	OPMODE		RSTCNT	ACTSTS	EXTCNTEN
23	22	21	20	19	18	17	16
WKEN	TGLPINSEL	TRGEADC	TRGDAC	TRGPWM	TRGSSEL	WKTKEN	Reserved
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
PSC							

Bits	Description	
[31]	**ICEDEBUG**	**ICE Debug Mode Acknowledge Disable (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 set 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 (TIMERx_CTL[28:27] = 00) when the timer interrupt flag TIF (TIMERx_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 is generated and inform to CPU.

[28:27]	**OPMODE**	**Timer Counting Mode Select** 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 Bit** Setting this bit will reset the 24-bit up counter value CNT (TIMERx_CNT[23:0]) and also force CNTEN (TIMERx_CTL[30]) to 0 if ACTSTS (TIMERx_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 Bit (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 PCLKx (x= 0~1) as timer clock source.
[23]	**WKEN**	**Wake-up Function Enable Bit** If this bit is set to 1, while timer interrupt flag TIF (TIMERx_INTSTS[0]) is 1 and INTEN (TIMERx_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]	**TGLPINSEL**	**Toggle-output Pin Select** 0 = Toggle mode output to Tx (Timer Event Counter Pin). 1 = Toggle mode output to Tx_EXT (Timer External Capture Pin).
[21]	**TRGEADC**	**Trigger EADC Enable Bit** If this bit is set to 1, timer time-out interrupt or capture interrupt can trigger EADC. 0 = Timer interrupt trigger EADC Disabled. 1 = Timer interrupt trigger EADC Enabled. **Note:** If TRGSSEL (TIMERx_CTL[18]) = 0, time-out interrupt signal will trigger EADC. If TRGSSEL (TIMERx_CTL[18]) = 1, capture interrupt signal will trigger EADC.
[20]	**TRGDAC**	**Trigger DAC Enable Bit** If this bit is set to 1, timer time-out interrupt or capture interrupt can trigger DAC. 0 = Timer interrupt trigger DAC Disabled. 1 = Timer interrupt trigger DAC Enabled. **Note:** If TRGSSEL (TIMERx_CTL[18]) = 0, time-out interrupt signal will trigger DAC. If TRGSSEL (TIMERx_CTL[18]) = 1, capture interrupt signal will trigger DAC.
[19]	**TRGPWM**	**Trigger PWM Enable Bit** If this bit is set to 1, timer time-out interrupt or capture interrupt can trigger PWM.

		0 = Timer interrupt trigger PWM Disabled. 1 = Timer interrupt trigger PWM Enabled. **Note:** If TRGSSEL (TIMERx_CTL[18]) = 0, time-out interrupt signal will trigger PWM. If TRGSSEL (TIMERx_CTL[18]) = 1, capture interrupt signal will trigger PWM.
[18]	**TRGSSEL**	**Trigger Source Select Bit** This bit is used to select trigger source is from Timer time-out interrupt signal or capture interrupt signal. 0 = Timer time-out interrupt signal is used to trigger PWM, EADC and DAC. 1 = Capture interrupt signal is used to trigger PWM, EADC and DAC.
[17]	**WKTKEN**	**Wake-up Touch-key Scan Enable Bit** If this bit is set to 1, timer time-out interrupt in Power-down mode can trigger Touck-Key start scan. 0 = Timer time-out interrupt signal trigger Touch-Key scan Disabled. 1 = Timer time-out interrupt signal trigger Touch-Key scan Enabled. **Note:** This bit is only available in TIMER0_CTL.
[16:8]	**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.

Timer Compare Register (TIMERx_CMP)

Register	Offset	R/W	Description	Reset Value
TIMER0_CMP	TMR_BA01+0x04	R/W	Timer0 Compare Register	0x0000_0000
TIMER1_CMP	TMR_BA01+0x24	R/W	Timer1 Compare Register	0x0000_0000
TIMER2_CMP	TMR_BA23+0x04	R/W	Timer2 Compare Register	0x0000_0000
TIMER3_CMP	TMR_BA23+0x24	R/W	Timer3 Compare Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CMPDAT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CMPDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMPDAT							

Bits	Description	
[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 TIF (TIMERx_INTSTS[0] Timer Interrupt Flag) 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 CMPDAT field.

Timer Interrupt Status Register (TIMERx_INTSTS)

Register	Offset	R/W	Description	Reset Value
TIMER0_INTSTS	TMR_BA01+0x08	R/W	Timer0 Interrupt Status Register	0x0000_0000
TIMER1_INTSTS	TMR_BA01+0x28	R/W	Timer1 Interrupt Status Register	0x0000_0000
TIMER2_INTSTS	TMR_BA23+0x08	R/W	Timer2 Interrupt Status Register	0x0000_0000
TIMER3_INTSTS	TMR_BA23+0x28	R/W	Timer3 Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**TWKF**	**TIF**

Bits	Description	
[31:2]	**Reserved**	Reserved.
[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 (TIMERx_CNT[23:0]) value reaches to CMPDAT (TIMERx_CMP[23:0]) value. 0 = No effect. 1 = CNT value matches the CMPDAT value. **Note:** This bit is cleared by writing 1 to it.

Timer Data Register (TIMERx_CNT)

Register	Offset	R/W	Description	Reset Value
TIMER0_CNT	TMR_BA01+0x0C	R	Timer0 Data Register	0x0000_0000
TIMER1_CNT	TMR_BA01+0x2C	R	Timer1 Data Register	0x0000_0000
TIMER2_CNT	TMR_BA23+0x0C	R	Timer2 Data Register	0x0000_0000
TIMER3_CNT	TMR_BA23+0x2C	R	Timer3 Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CNT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CNT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNT							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**CNT**	**Timer Data Register** This field can be reflected the internal 24-bit timer counter value or external event input counter value from Tx (x=0~3) pin. If EXTCNTEN (TIMERx_CTL[24]) is 0, user can read CNT value for getting current 24- bit counter value . If EXTCNTEN (TIMERx_CTL[24]) is 1, user can read CNT value for getting current 24- bit event input counter value.

Timer Capture Data Register (TIMERx_CAP)

Register	Offset	R/W	Description	Reset Value
TIMER0_CAP	TMR_BA01+0x10	R	Timer0 Capture Data Register	0x0000_0000
TIMER1_CAP	TMR_BA01+0x30	R	Timer1 Capture Data Register	0x0000_0000
TIMER2_CAP	TMR_BA23+0x10	R	Timer2 Capture Data Register	0x0000_0000
TIMER3_CAP	TMR_BA23+0x30	R	Timer3 Capture Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CAPDAT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CAPDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CAPDAT							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**CAPDAT**	**Timer Capture Data Register** When CAPEN (TIMERx_EXTCTL[3]) bit is set, CAPFUNCS (TIMERx_EXTCTL[4]) bit is 0, and a transition on Tx_EXT pin matched the CAPEDGE (TIMERx_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.

Timer External Control Register (TIMERx_EXTCTL)

Register	Offset	R/W	Description	Reset Value
TIMER0_EXTCTL	TMR_BA01+0x14	R/W	Timer0 External Control Register	0x0000_0000
TIMER1_EXTCTL	TMR_BA01+0x34	R/W	Timer1 External Control Register	0x0000_0000
TIMER2_EXTCTL	TMR_BA23+0x14	R/W	Timer2 External Control Register	0x0000_0000
TIMER3_EXTCTL	TMR_BA23+0x34	R/W	Timer3 External Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
CNTDBEN	CAPDBEN	CAPIEN	CAPFUNCS	CAPEN	CAPEDGE		CNTPHASE

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**CNTDBEN**	**Timer Counter Pin De-bounce Enable Bit** 0 = Tx (x= 0~3) pin de-bounce Disabled. 1 = Tx (x= 0~3) pin de-bounce Enabled. **Note:** If this bit is enabled, the edge detection of Tx pin is detected with de-bounce circuit.
[6]	**CAPDBEN**	**Timer External Capture Pin De-bounce Enable Bit** 0 = Tx_EXT (x= 0~3) pin de-bounce Disabled. 1 = Tx_EXT (x= 0~3) pin de-bounce Enabled. **Note:** If this bit is enabled, the edge detection of Tx_EXT pin is detected with de-bounce circuit.
[5]	**CAPIEN**	**Timer External Capture Interrupt Enable Bit** 0 = Tx_EXT (x= 0~3) pin detection Interrupt Disabled. 1 = Tx_EXT (x= 0~3) pin detection Interrupt Enabled. **Note:** CAPIEN is used to enable timer external interrupt. If CAPIEN enabled, timer will rise an interrupt when CAPIF (TIMERx_EINTSTS[0]) is 1. For example, while CAPIEN = 1, CAPEN = 1, and CAPEDGE = 00, a 1 to 0 transition on the Tx_EXT pin will cause the CAPIF to be set then the interrupt signal is generated and sent to NVIC to inform CPU.
[4]	**CAPFUNCS**	**Capture Function Selection** 0 = External Capture Mode Enabled.

		1 = External Reset Mode Enabled. **Note1:** When CAPFUNCS is 0, transition on Tx_EXT (x= 0~3) pin is using to save the 24-bit timer counter value. **Note2:** When CAPFUNCS is 1, transition on Tx_EXT (x= 0~3) pin is using to reset the 24-bit timer counter value.
[3]	**CAPEN**	**Timer External Capture Pin Enable Bit** This bit enables the Tx_EXT pin. 0 =Tx_EXT (x= 0~3) pin Disabled. 1 =Tx_EXT (x= 0~3) pin Enabled.
[2:1]	**CAPEDGE**	**Timer External Capture Pin Edge Detect** 00 = A Falling edge on Tx_EXT (x= 0~3) pin will be detected. 01 = A Rising edge on Tx_EXT (x= 0~3) pin will be detected. 10 = Either Rising or Falling edge on Tx_EXT (x= 0~3) pin will be detected. 11 = Reserved.
[0]	**CNTPHASE**	**Timer External Count Phase** This bit indicates the detection phase of external counting pin Tx (x= 0~3). 0 = A Falling edge of external counting pin will be counted. 1 = A Rising edge of external counting pin will be counted.

Timer External Interrupt Status Register (TIMERx_EINTSTS)

Register	Offset	R/W	Description	Reset Value
TIMER0_EINTSTS	TMR_BA01+0x18	R/W	Timer0 External Interrupt Status Register	0x0000_0000
TIMER1_EINTSTS	TMR_BA01+0x38	R/W	Timer1 External Interrupt Status Register	0x0000_0000
TIMER2_EINTSTS	TMR_BA23+0x18	R/W	Timer2 External Interrupt Status Register	0x0000_0000
TIMER3_EINTSTS	TMR_BA23+0x38	R/W	Timer3 External Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							CAPIF

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**CAPIF**	**Timer External Capture Interrupt Flag** This bit indicates the timer external capture interrupt flag status. 0 = Tx_EXT (x= 0~3) pin interrupt did not occur. 1 = Tx_EXT (x= 0~3) pin interrupt occurred. **Note1:** This bit is cleared by writing 1 to it. **Note2:** When CAPEN (TIMERx_EXTCTL[3]) bit is set, CAPFUNCS (TIMERx_EXTCTL[4]) bit is 0, and a transition on Tx_EXT (x= 0~3) pin matched the CAPEDGE (TIMERx_EXTCTL[2:1]) setting, this bit will set to 1 by hardware. **Note3:** There is a new incoming capture event detected before CPU clearing the CAPIF status. If the above condition occurred, the Timer will keep register TIMERx_CAP unchanged and drop the new capture value.

6.9 PWM Generator and Capture Timer (PWM)

6.9.1 Overview

The M4TK provides two PWM generators — PWM0 and PWM1 as Figure 6.9-1. Each PWM supports 6 channels of PWM output or input capture. There is a 12-bit prescaler to support flexible clock to the 16-bit PWM counter with 16-bit comparator. The PWM counter supports up, down and up-down counter types. PWM using comparator compared with counter to generate events. These events use to generate PWM pulse, interrupt and trigger signal for EADC/DAC to start conversion.

The PWM generator supports two standard PWM output modes: Independent mode and Complementary mode, they have difference architecture. There are two output functions based on standard output modes: Group function and Synchronous function. Group function can be enabled under Independent mode or complementary mode. Synchronous function only enabled under complementary mode. Complementary mode has two comparators to generate various PWM pulse with 12-bit dead-time generator and another free trigger comparator to generate trigger signal for EADC. For PWM output control unit, it supports polarity output, independent pin mask and brake functions.

The PWM generator also supports input capture function. It supports latch PWM counter value to corresponding register when input channel has a rising transition, falling transition or both transition is happened. Capture function also support PDMA to transfer captured data to memory.

6.9.2 Features

6.9.2.1 PWM function features

- Supports maximum clock frequency up to144MHz
- Supports up to two PWM modules, each module provides 6 output channels.
- Supports independent mode for PWM output/Capture input channel
- Supports complementary mode for 3 complementary paired PWM output channel
  - Dead-time insertion with 12-bit resolution
  - Synchronous function for phase control
  - Two compared values during one period
- Supports 12-bit pre-scalar from 1 to 4096
- Supports 16-bit resolution PWM counter
  - Up, down and up/down counter operation type
- Supports one-shot or auto-reload counter operation mode
- Supports group function
- Supports synchronous function
- Supports mask function and tri-state enable for each PWM pin
- Supports brake function
  - Brake source from pin, analog comparator and system safety events (clock failed, SRAM parity error, Brown-out detection and CPU lockup).
  - Noise filter for brake source from pin
  - Edge detect brake source to control brake state until brake interrupt cleared

- Level detect brake source to auto recover function after brake condition removed
- Supports interrupt on the following events:
  - PWM counter match zero, period value or compared value
  - Brake condition happened
- Supports trigger EADC/DAC on the following events:
  - PWM counter match zero, period value or compared value
  - PWM counter match free trigger comparator compared value (only for EADC)

6.9.2.2 Capture Function Features

- Supports up to 12 capture input channels with 16-bit resolution
- Supports rising or falling capture condition
- Supports input rising/falling capture interrupt
- Supports rising/falling capture with counter reload option
- Supports PDMA transfer function for PWM all channels

6.9.3 Block Diagram

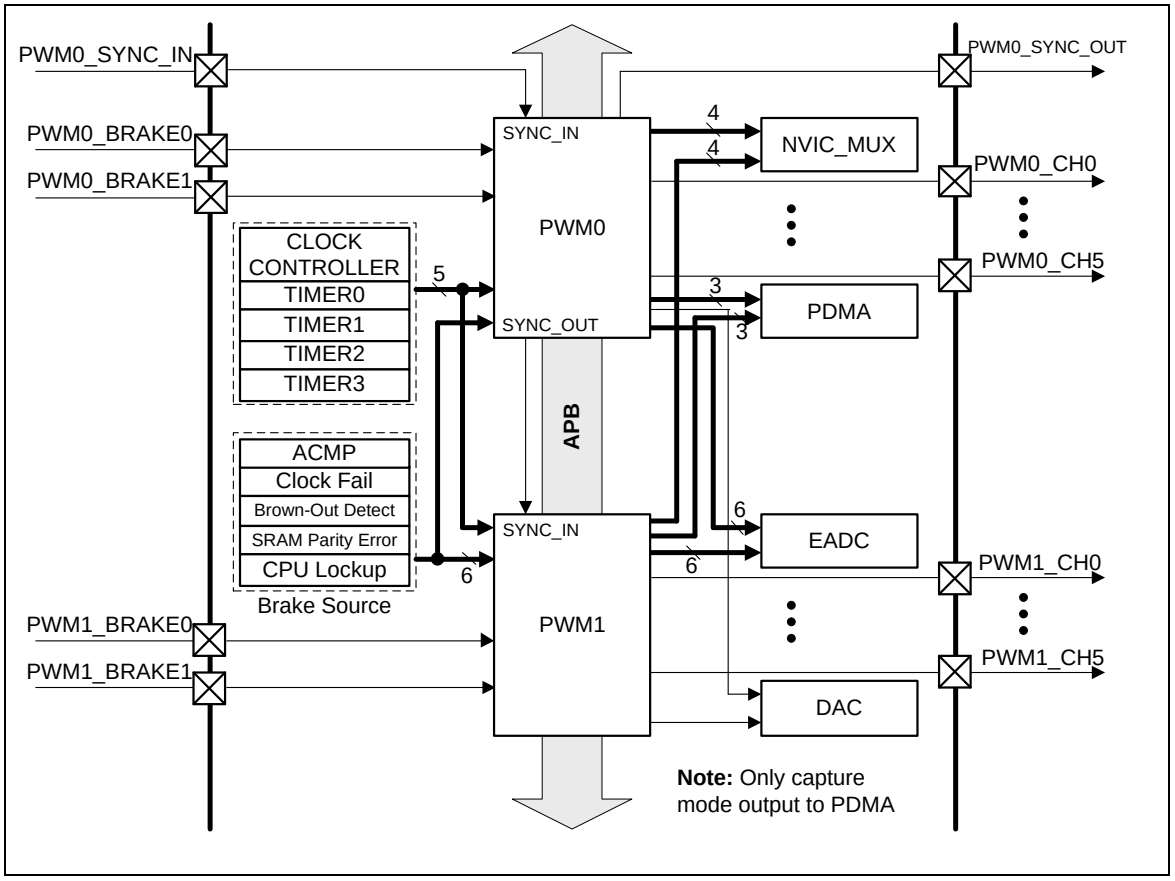


Figure 6.9-1 PWM Generator Overview Block Diagram

PWM system clock frequency can be set equal or double to HCLK frequency as Figure 6.9-2, the detail register setting, please refer to Table 6-12.

Each PWM generator has three clock source inputs, each clock source can be selected from system clock or four TIMER trigger PWM outputs as Figure 6.9-3 by ECLKSRC0 (PWM_CLKSRC[2:0]) for PWM_CLK0, ECLKSRC2 (PWM_CLKSRC[10:8]) for PWM_CLK2 and ECLKSRC4 (PWM_CLKSRC[18:16]).

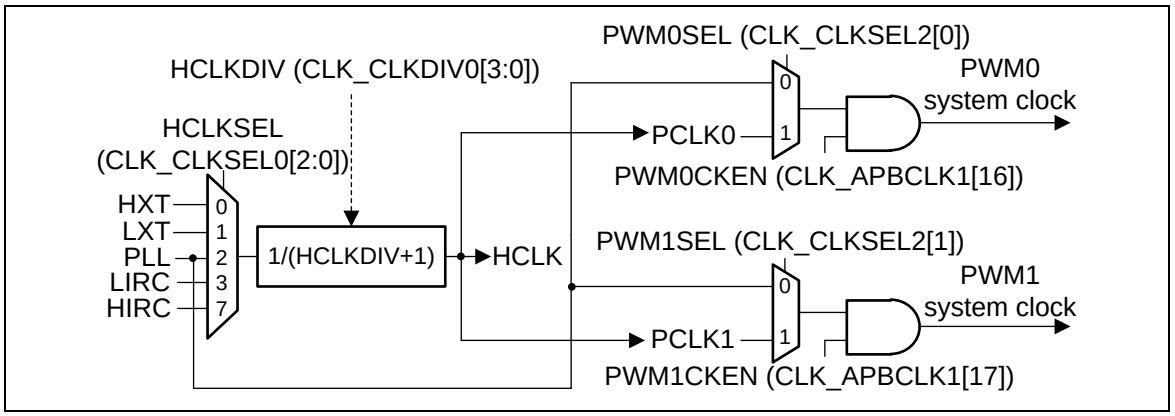


Figure 6.9-2 PWM System Clock Source Control

PWM System Clock/HCLK Frequency Ratio	HCLKSEL (CLK_CLKSEL0[2:0])	HCLKDIV (CLK_CLKDIV0[3:0])	PWMnSEL (CLK_CLKSEL2[N]), N Denotes 0 Or 1
1/1	Don't care	Don't care	1
2/1	2	1	0

Table 6-12 PWM System Clock Source Control Registers Setting Table

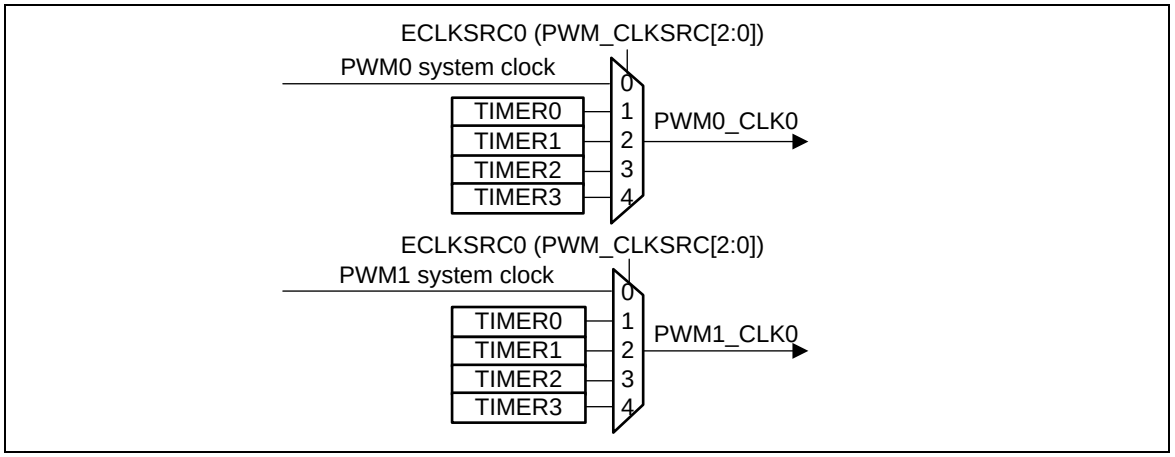


Figure 6.9-3 PWM Clock Source Control

Figure 6.9-4 and Figure 6.9-5 illustrate the architecture of PWM independent mode and complementary mode. No matter independent mode or complementary mode, paired channels' (PWM_CH0 and PWM_CH1, PWM_CH2 and PWM_CH3, PWM_CH4 and PWM_CH5) counters both come from the same clock source and prescaler. When counter count to 0, PERIOD (PWM_PERIODn[15:0]) or equal to comparator, events will be generated. These events are passed to corresponding generators to generate PWM pulse, interrupt signal and trigger signal for EADC/DAC to start conversion. Output control is used to changing PWM pulse output state; brake function in output control also generates interrupt events. In complementary mode, synchronize function is available and even channel use odd channel comparator to generate events, free trigger comparator events only use to generate trigger EADC signals.

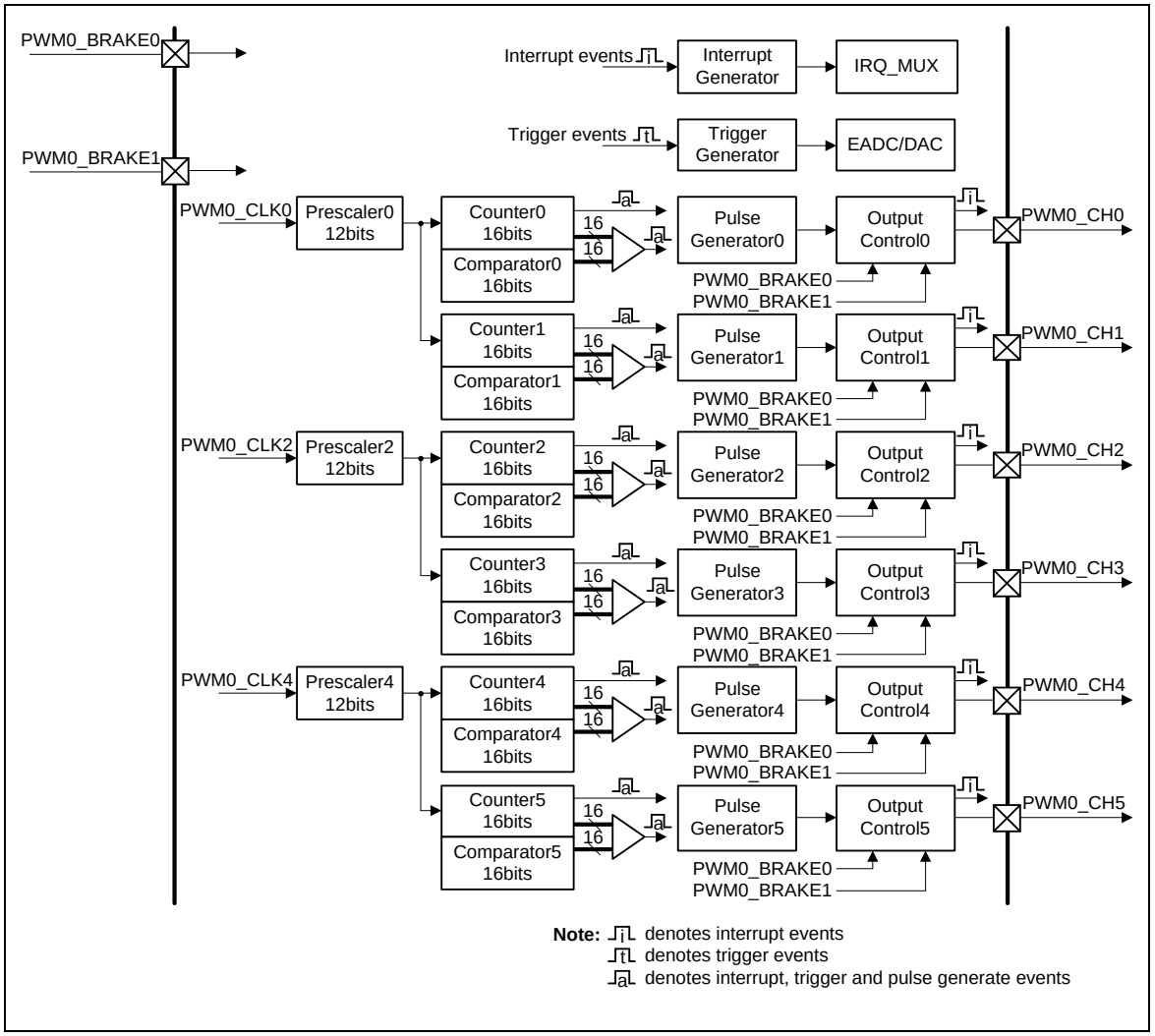


Figure 6.9-4 PWM Independent Mode Architecture Diagram

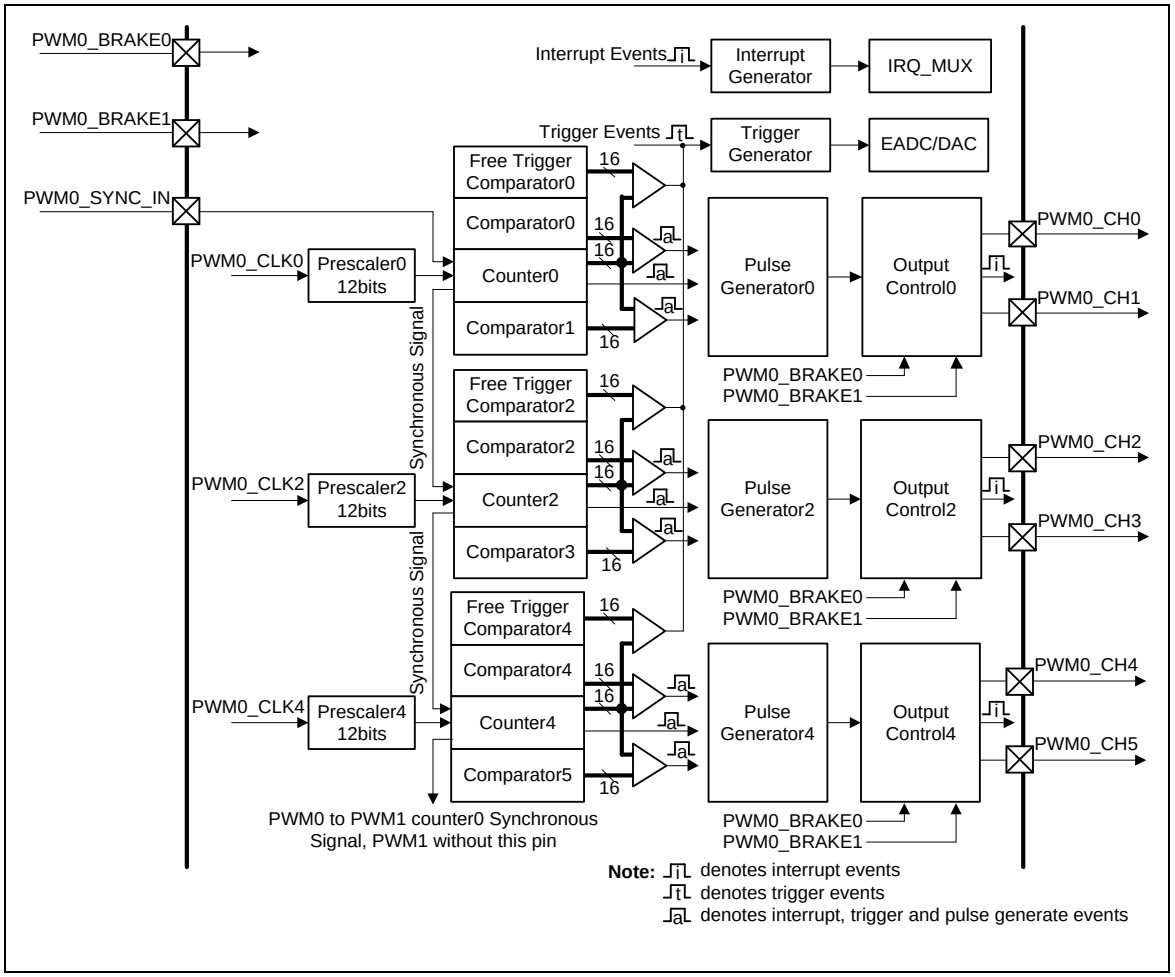


Figure 6.9-5 PWM Complementary Mode Architecture Diagram

6.9.4 Basic Configuration

The PWM0 pin functions are configured in SYS_GPC_MFPL, the PWM1 pin functions are configured in SYS_GPC_MFPH Multiple Function Registers.

PWM0_SYNC_IN, PWM0_BRAKE0 and PWM0_BRAKE1 pin functions are configured in SYS_GPD_MFPL Multiple Function Registers. PWM1_BRAKE0 and PWM1_BRAKE1 pin functions are configured in SYS_GPE_MFPL Multiple Function Registers.

PWM0_SYNC_OUT pin function is configured in SYS_GPB_MFPL Multiple Function Register.

The PWM clock can be enabled in CLK_APBCLK1[17:16]. The PWM clock source is selected by CLK_CLKSEL2[1:0].

6.9.5 Functional Description

6.9.5.1 PWM Prescaler

PWM prescaler is used to divide clock source, prescaler counting CLKPSC +1 times, PWM counter only count once. CLKPSC (Clock Pre-scale Register) is setting by CLKPSC (PWM_CLKPSCn[11:0], n denotes 0, 2, 4). Figure 6.9-6 is an example of PWM channel 0 prescale waveform.

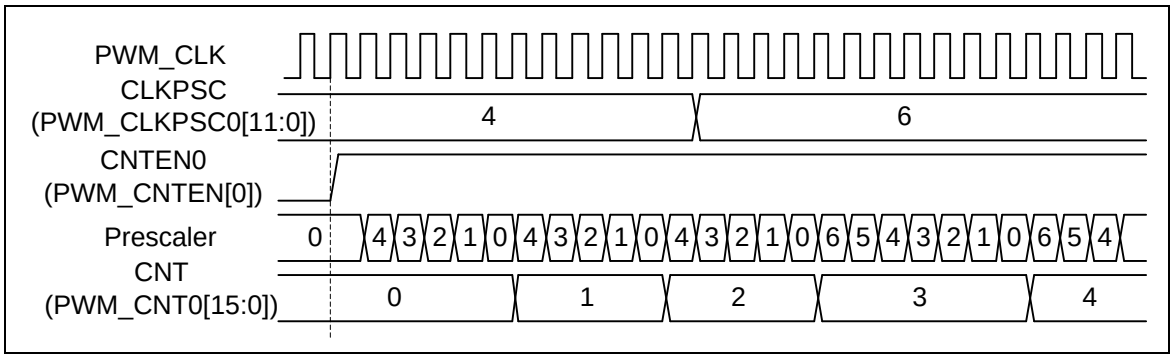


Figure 6.9-6 PWM_CH0 prescaler waveform

6.9.5.2 PWM Counter

PWM supports 3 counter types operation: Up Counter, Down Counter and Up-Down Counter types.

6.9.5.3 Up Counter Type

In the up counter operation, the 16 bits PWM counter is an up counter and starts up-counting from zero to PERIOD (PWM_PERIODn[15:0], where n denotes channel number) to complete a PWM period. The current counter value can be found by reading the CNT (PWM_CNTn[15:0]). PWM generates zero point event when the counter counts to 0 and generates period point event when counting to PERIOD. The Figure 6.9-7 shows an example of up counter, wherein PWM period time = (PERIOD+1) x PWM clock time.

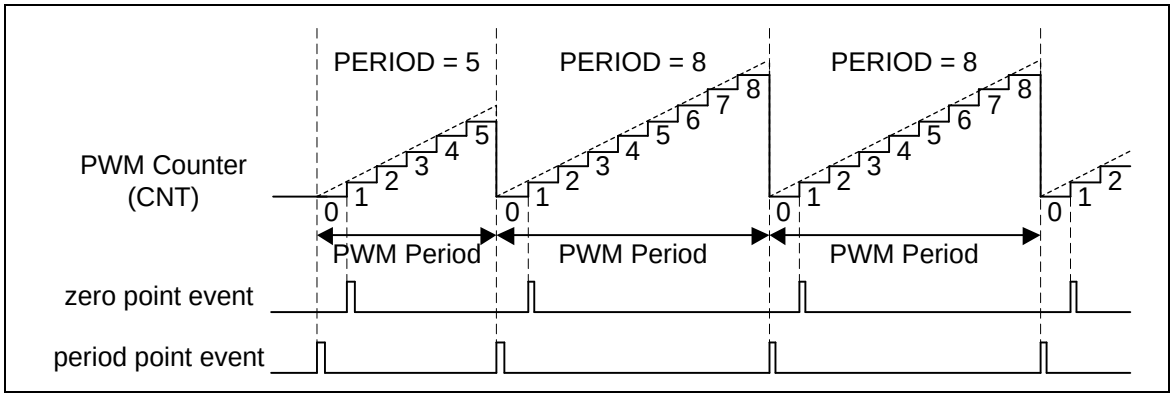


Figure 6.9-7 PWM Up Counter Type

6.9.5.4 Down Counter Type

In the down counter type operation, the 16 bits PWM counter is a down counter and starts down-counting from PERIOD to zero to complete a PWM period. The current counter value can be found by reading the CNT. PWM generates zero point event when the counter counts to 0 and generates period point event when counting to PERIOD. The Figure 6.9-8 shows an example of down counter, wherein PWM period time = (PERIOD+1) x PWM clock time.

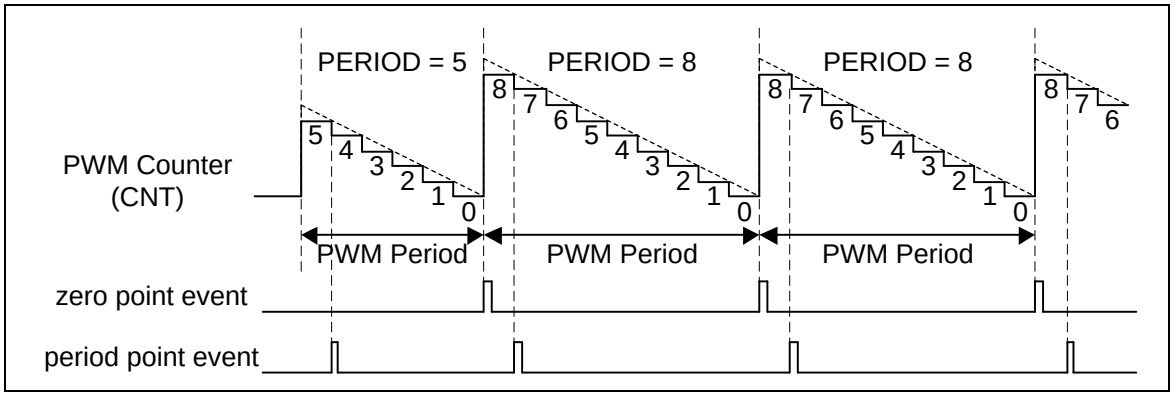


Figure 6.9-8 PWM Down Counter Type

6.9.5.5 Up-Down Counter Type

In the up-down counter operation, the 16 bits PWM counter is an up-down counter and starts counting-up from zero to PERIOD and then starts counting down to zero to complete a PWM period. The current counter value can be found by reading the CNT. PWM generates zero point event when the counter counts to 0 and generates center point event when counting to PERIOD. The Figure 6.9-9 shows an example of up-down counter, wherein PWM period time = (2xPERIOD) x PWM clock time. The DIRF (PWM_CNTn[16]) is counter direction indicator flag, where high is up counting, and low is down counting.

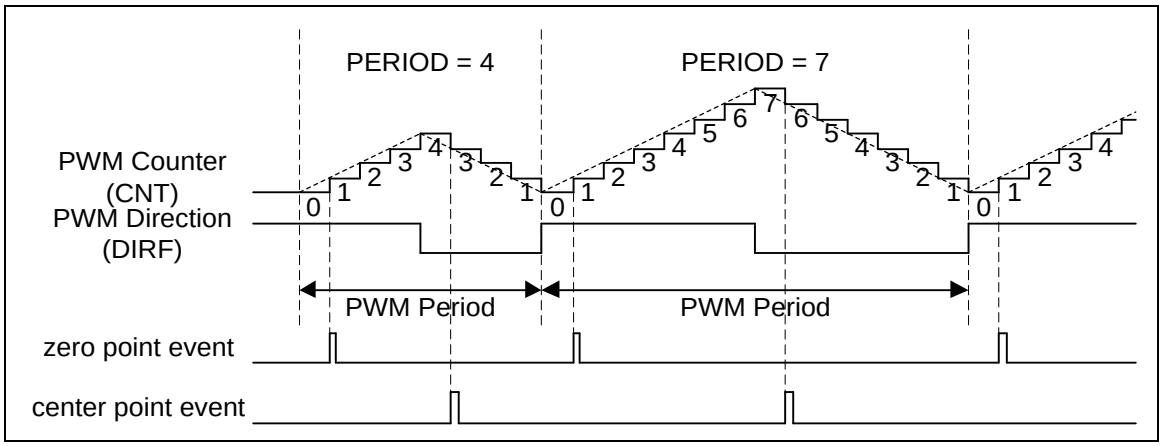


Figure 6.9-9 PWM Up-Down Counter Type

6.9.5.6 PWM Comparator

There are two kinds of comparator registers – one is CMPDAT (PWM_CMPDATn[15:0]) and the other is FTCMPDAT (PWM_FTCMPDATn[15:0]). CMPDAT is a basic comparator register of PWM channel n; each channel only has one CMPDAT. In Independent mode, the CMPDAT's value is continuously compared to the corresponding channel's counter value. In Complementary mode, odd channel's counter is useless and the corresponding comparator is continuously compared to the complementary even channel. For example, channel 0 and channel 1 are complementary channels, in Complementary mode, channel 1's comparator is continuously compared to channel 0's counter, but not channel 1's. When the counter is equal to compared register, PWM generates an event and uses the event to generate PWM pulse, interrupt or use to trigger EADC/DAC. In up-down counter type, two events will be generated in a PWM period as

shown in Figure 6.9-10.

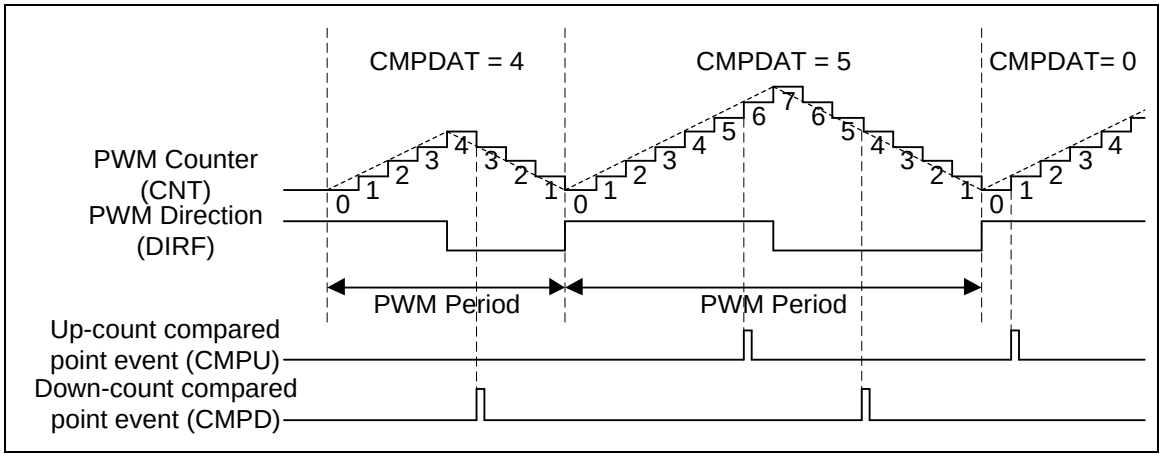


Figure 6.9-10 PWM CMPDAT Events in Up-Down Counter Type

FTCMPDAT is a free trigger comparator register. Each complementary paired channel only supports one FTCMPDAT. FTCMPDAT is continuously compared to even channel of complementary channels. When CNT is equal to FTCMPDAT, PWM generates an event and only uses the event to trigger EADC.

6.9.5.7 PWM Double Buffering

The double buffering uses double buffers to separate software writing and hardware action operation timing. After registers are modified through software, hardware will load register value to the buffer register according to the loading mode timing. The hardware action is based on the buffer value. This can prevent asynchronously operation problem due to software and hardware asynchronism.

The PWM has double buffering function for PERIOD, CMPDAT and FTCMPDAT. CMPDAT and FTCMPDAT to have the same loading timing. The concept of double buffering is used in loading modes, which are described in the following sections. For example, as shown in Figure 6.9-11, in period loading mode, writing PERIOD, CMPDAT and FTCMPDAT through software, PWM will load new values to their buffer PBUF (PWM_PBUFn[15:0]), CMPBUF (PWM_CMPBUFn[15:0]) and FTCMPBUF (PWM_FTCBUF[15:0]) at start of the next period without affecting the current period counter operation. FTCMPU denotes FTCMPDAT up-count compared point event and FTCMPD denotes FTCMPDAT down-count compared event. There are four loading modes for loading values to buffer: period loading mode, immediately loading mode, window loading mode and center loading mode.

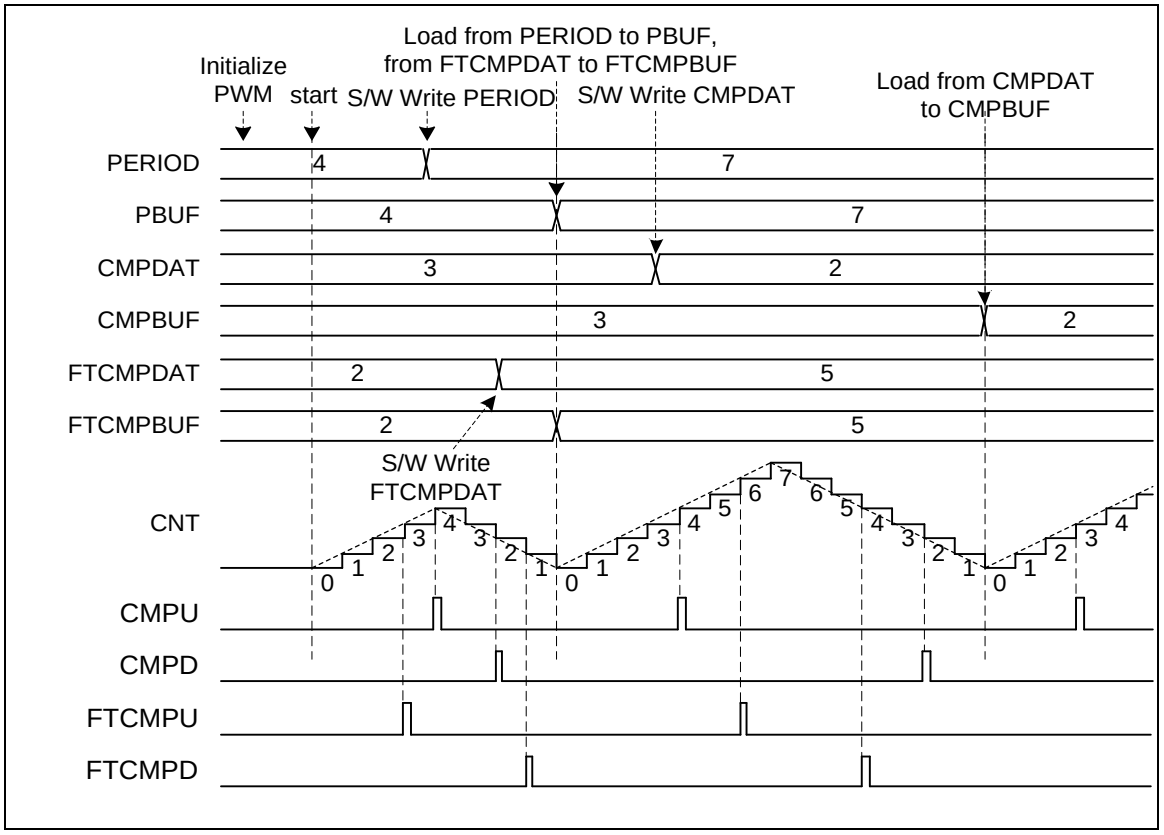


Figure 6.9-11 PWM Double Buffering Illustration

6.9.5.8 Period Loading Mode

Period Loading mode is the default loading mode. It has the lowest priority in loading modes. PERIOD and CMPDAT will both load to their buffer while a period is completed. For example, after PWM counter up counts from zero to PERIOD in the up-counter operation or down counts from PERIOD to zero in the down-counter operation or up counts from zero to PERIOD and then down counts to zero in the up-down counter operation.

Figure 6.9-12 shows period loading timing of up-count operation, where PERIOD DATA0 denotes the initial data of PERIOD, PERIOD DATA1 denotes the first updated PERIOD data by software and so on. CMPDAT also follows this rule. The following describes steps sequence of Figure 6.9-12. User can know the PERIOD and CMPDAT update condition, by watching PWM period and CMPU event.

1. Software writes CMPDAT DATA1 to CMPDAT at point 1.
2. Hardware loads CMPDAT DATA1 to CMPBUF at the end of PWM period at point 2.
3. Software writes PERIOD DATA1 to PERIOD at point 3.
4. Hardware loads PERIOD DATA1 to PBUF at the end of PWM period at point 4.
5. Software writes PERIOD DATA2 to PERIOD at point 5.
6. Hardware loads PERIOD DATA2 to PBUF at the end of PWM period at point 6.

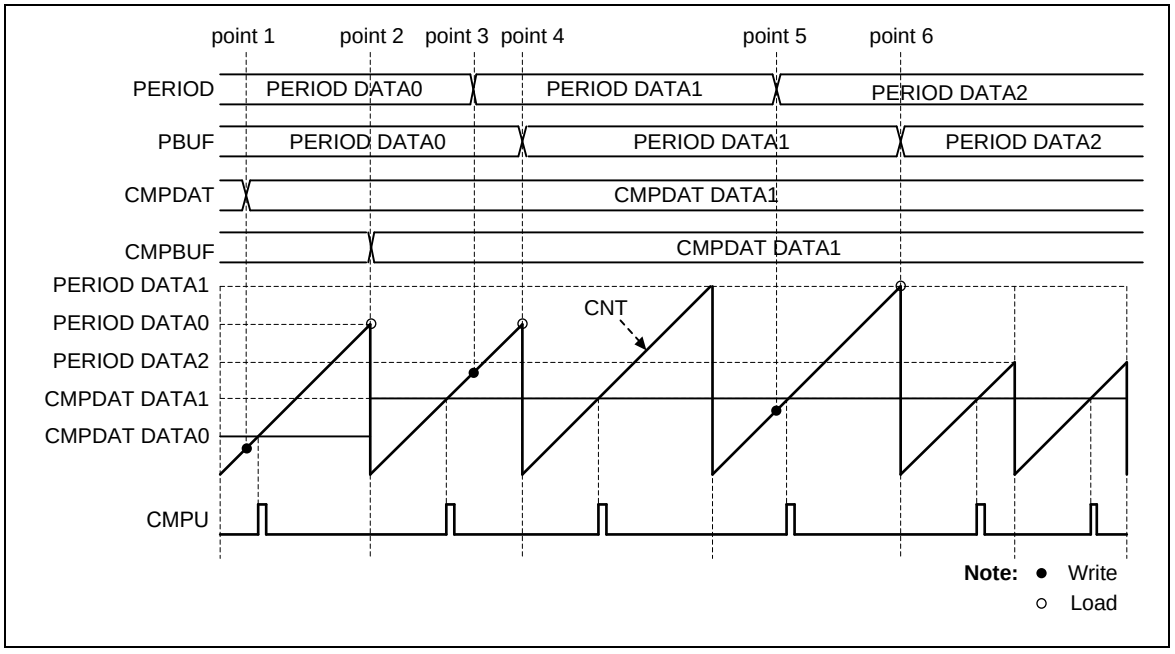


Figure 6.9-12 Period Loading in Up-Count Mode

6.9.5.9 Immediately Loading Mode

If the IMMLDENn (PWM_CTL0[21:16]) bit which corresponds to PWM channel n is set to 1, software will load a value to buffer from PERIOD and CMPDAT immediately while software updates PERIOD or CMPDAT. If the updated PERIOD value is less than current counter value, counter will count wraparound. Immediately loading mode has the highest priority. If IMMLDENn has been set, other loading mode for channel n will become invalid. Figure 6.9-13 shows an example and its steps sequence is described below.

1. Software writes CMPDAT DATA1 and hardware immediately loading CMPDAT DATA1 to CMPBUF at point 1.
2. Software writes PERIOD DATA1 which is greater than current counter value at point 2; counter will continue counting until equal to PERIOD DATA1 to finish a period loading.
3. Software writes PERIOD DATA2 which is less than the current counter value at point 3; counter will continue counting to its maximum value 0xFFFF and count wraparound from 0 to PERIOD DATA2 to finish this period loading.

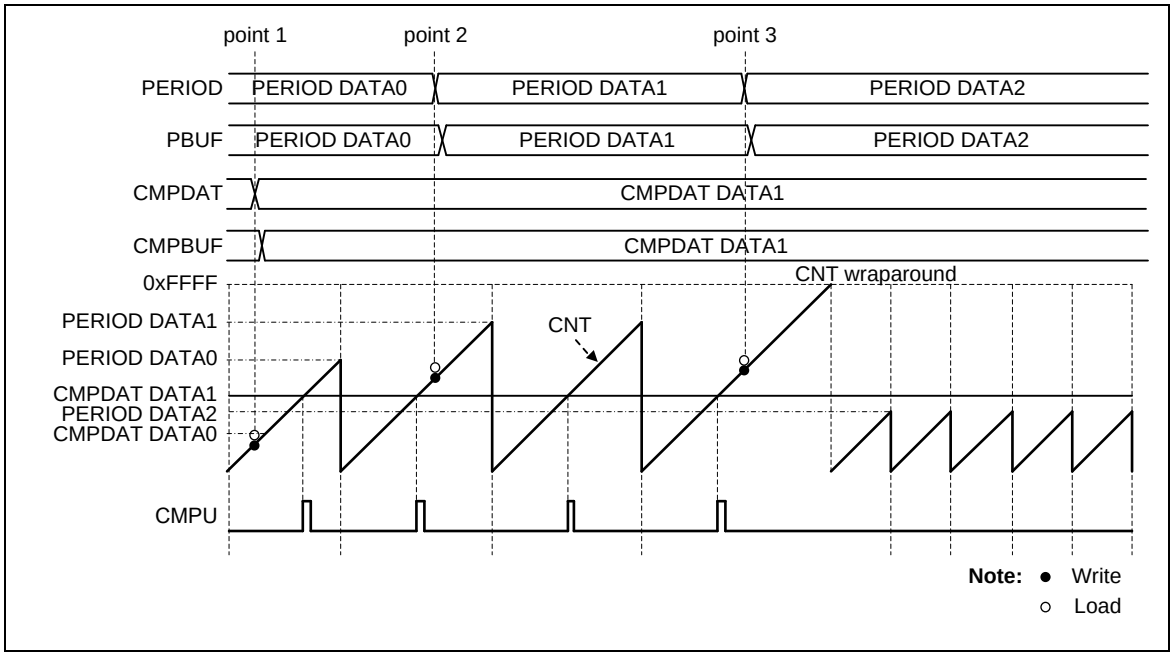


Figure 6.9-13 Immediately Loading in Up-Count Mode

6.9.5.10 Window Loading Mode

If the WINLDENn (PWM_CTL0[13:8]) bit which corresponds to PWM channel n is set to 1, the channel n window loading mode is enabled. Window loading mode also loads a value from PERIOD and CMPDAT to their buffer at the end of a period as period loading mode, but CMPDAT loading to CMPBUF is valid only when load window is opened and PERIOD loading to PBUF at the end of every PWM period no matter the load window is opened or not. Every channel n's load window is opened by setting the corresponding LOADn (PWM_LOAD[5:0]) to 1 and hardware will close the window at the end of PWM period. Window loading mode can work with center loading mode and CMPDAT loading time is also valid only at the interval of load window Figure 6.9-14 shows an example and its steps sequence is described below.

1. Software writes CMPDAT DATA1 at point 1, and the load window is not opened at this period so CMPDAT will not load to CMPBUF.
2. Software writes LOAD to open the load window at point2.
3. Software writes PERIOD DATA1 at point 3.
4. At point 4, load window has been opened, hardware loads PERIOD DATA1 and CMPDAT DATA1 to their buffer and closes the load window at the end of PWM period.
5. Software writes PERIOD DATA2 at point 5.
6. Hardware loads PERIOD DATA2 to PBUF at the end of PWM period at point 6.
7. Software writes PERIOD DATA3 at point 7.
8. Software writes LOAD to open the load window at point8.
9. Hardware loads PERIOD DATA3 to PBUF and closes the load window at the end of PWM period at point 9.

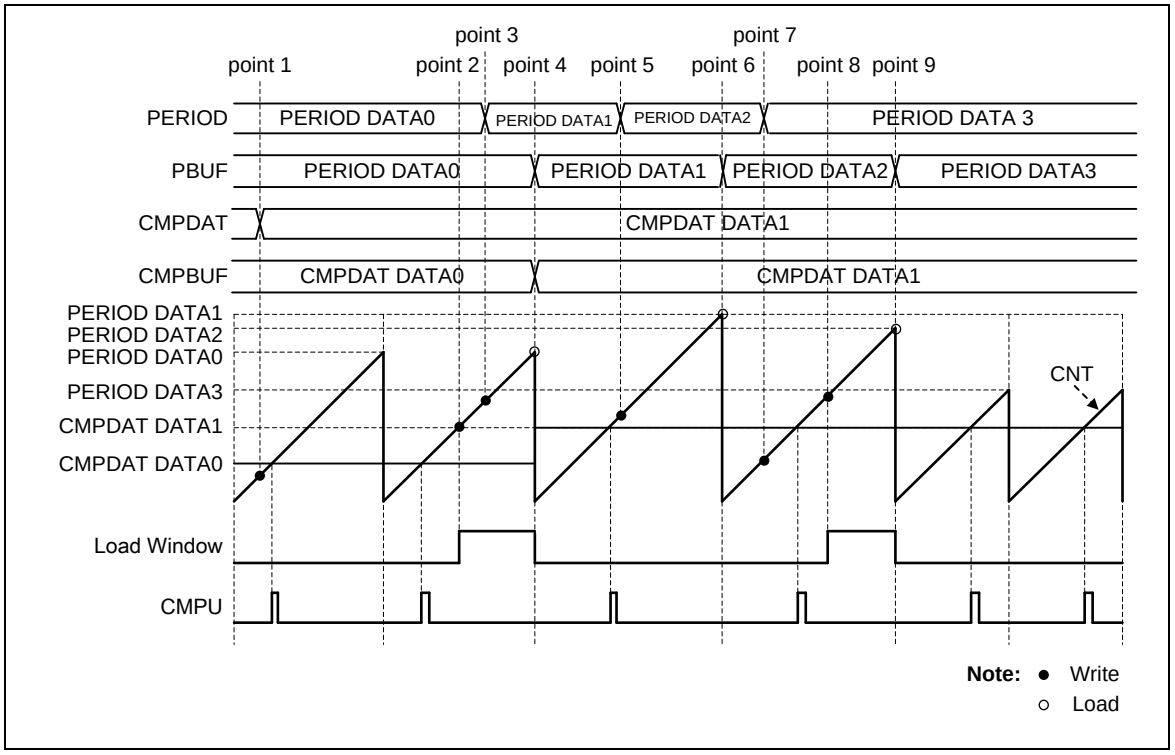


Figure 6.9-14 Window Loading in Up-Count Mode

6.9.5.11 Center Loading Mode

If the CTRLDn (PWM_CTL0[5:0]) bit which corresponds to PWM channel n set to 1 and in up-down counter type, CMPDAT will load to CMPBUFn in center of a period, that is, counter counts to PERIOD. PERIOD loading timing is the same as period loading mode. Figure 6.9-15 shows an example and its steps sequence is described below.

1. Software writes CMPDAT DATA1 at point 1.
2. Hardware loads CMPDAT DATA1 to CMPBUF at center of PWM period at point 2.
3. Software writes PERIOD DATA1 at point 3.
4. Hardware loads PERIOD DATA1 to PBUF at the end of PWM period at point 4.
5. Software writes CMPDAT DATA2 at point 5.
6. Hardware loads CMPDAT DATA2 to CMPBUF at center of PWM period at point 6.
7. Software writes PERIOD DATA2 at point 7.
8. Hardware loads PERIOD DATA2 to PBUF at the end of PWM period at point 8.

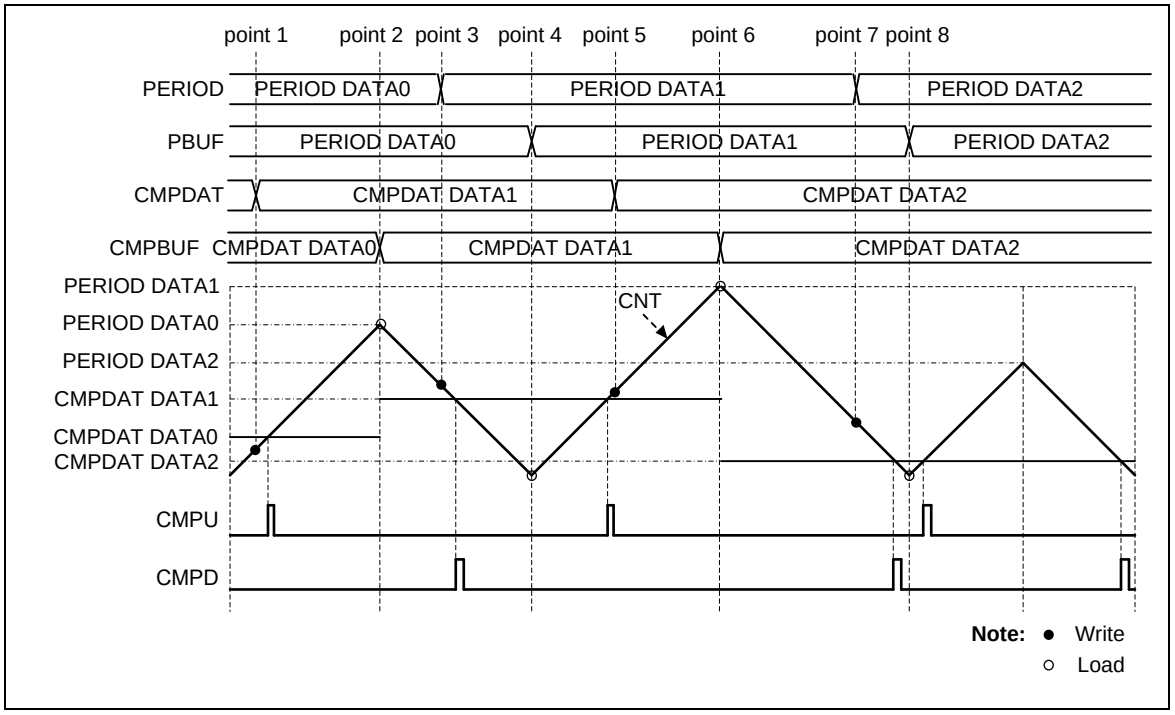


Figure 6.9-15 Center Loading in Up-Down-Count Mode

6.9.5.12 PWM Counter Operation mode

The PWM counter supports two operation modes: One-shot mode and Auto-reload mode. PWM counter will operate in One-shot mode if CNTMODEn (PWM_CTL1[21:16]) bit is set to 1, and operate in Auto-reload mode if set to 0.

In One-shot mode, CMPDAT and PERIOD should be written first and then set CNTENn (PWM_CNTEN[5:0]) channel n corresponding bit to 1 to enable PWM prescaler and counter start running. After PWM counter counted a period, counter value will keep in zero.

Software needs to write new CMPDAT to re-start next one-shot. If one-shot counter still running, write CMPDAT will cause next one-shot as continuous one-shot. Besides, if under continuous one-shot write CMPDAT twice, one-shot period will use latest value as CMPDAT and only generate one-shot pulse once. Figure 6.9-16 is an example and following is steps sequence.

1. Software writes PERIOD DATA1 and hardware immediately loading PERIOD DATA1 to PBUF at point 1.
2. Software writes CMPDAT DATA1 which is equal to CMPDAT DATA0 at point 2 and hardware immediately loading CMPDAT DATA1 to CMPBUF, this event also trigger one-shot.
3. Software writes CMPDAT DATA2 and re-trigger next one-shot (continuous one-shot) at point 3.
4. Software writes CMPDAT DATA3 to cover CMPDAT DATA2 and re-trigger next one-shot at point 4.
5. Period loading CMPDAT DATA3 to CMPBUF at point 5.
6. There are no new CMPDAT write in the previous period, and the counter value is kept as zero at point 6.

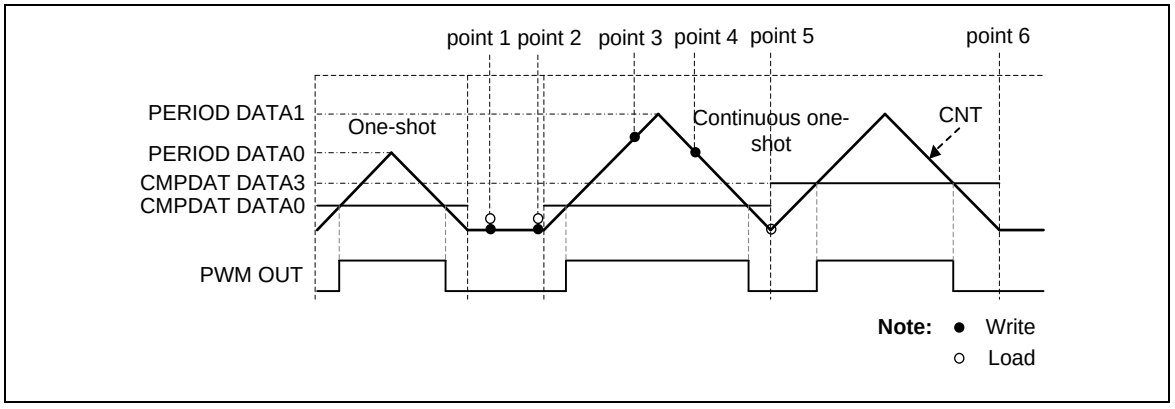


Figure 6.9-16 PWM One-shot Mode Output Waveform

In Auto-reload mode, CMPDAT and PERIOD should be written first and then the CNTENn channel n corresponding bit is set to 1 to enable PWM prescaler and start to run counter. The value of PERIOD and CMPDAT will auto reload to their buffer according different loading mode. If PERIOD is set to zero, PWM counter will be set to zero.

6.9.5.13 PWM Pulse Generator

The PWM pulse generator uses counter and comparator events to generate PWM pulse. The events are: zero point, period point in up counter type and down counter type, center point in up-down counter type and counter equal to comparator point in three types. As to up-down counter type, there are two counter equal comparator points, one at up count and the other at down count. Besides, Complementary mode has two comparators compared with counter, and thus comparing equal points will become four in up-down counter type and two for up or down counter type.

Each event point can decide PWM waveform to do nothing (X), set Low (L), set High (H) or toggle (T) by setting the PWM_WGCTL0 and PWM_WGCTL1 registers. Using these points can easily generate asymmetric PWM pulse or variant waveform as shown in Figure 6.9-17. In the figure, PWM is in complementary mode, there are two comparators n and m to generate PWM pulse. n denotes even channel number 0, 2, or 4, and m denotes odd channel number 1, 3, or 5. n channel and m channel are complementary paired. Complementary mode uses two channels (CH0 and CH1, CH2 and CH3, or CH4 and CH5) as a pair of PWM outputs to generate complement paired waveforms. CMPU denotes CNT is equal to CMPDAT when counting up. CMPD denotes CNT is equal to CMPDAT when counting down.

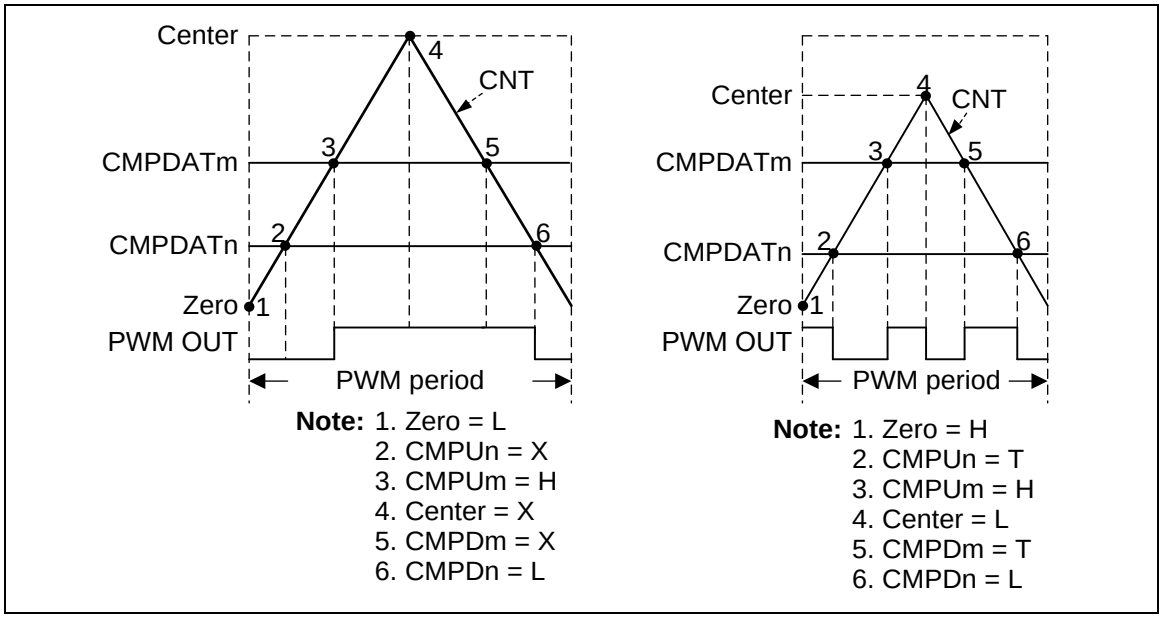


Figure 6.9-17 PWM Pulse Generation

The generation events may sometimes set to the same value, as the reason, events priority between different counter types are list below, up counter type(Table 6-13), down counter type (Table 6-14) and up-down counter type (Table 6-15). By using event priority, user can easily generate 0% to 100% duty pulse as shown in Figure 6.9-18.

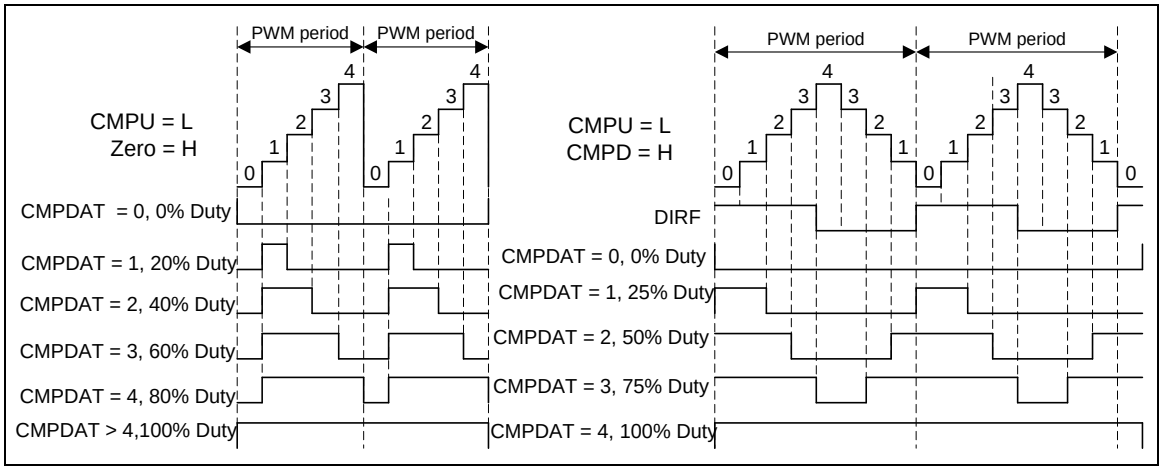


Figure 6.9-18 PWM 0% to 100% Pulse Generation

Priority	Up Event
1 (Highest)	CNT = period (PERIOD)
2	CNT = CMPUm
3	CNT = CMPUn
4 (Lowest)	CNT = zero

Table 6-13 PWM Pulse Generation Event Priority for Up-Counter

Priority	Down Event
1 (Highest)	CNT = zero
2	CNT = CMPDm
3	CNT = CMPDn
4 (Lowest)	CNT = period (PERIOD)

Table 6-14 PWM Pulse Generation Event Priority for Down-Counter

Priority	Up Event	Down Event
1 (Highest)	CNT = CMPUm	CNT = CMPDm
2	CNT= CMPUn	CNT = CMPDn
3	CNT = zero	CNT = center (PERIOD)
4	CNT = CMPDm	CNT = CMPUm
5 (Lowest)	PERIOD = CMPDn	CNT = CMPUn

Table 6-15 PWM Pulse Generation Event Priority for Up-Down-Counter

6.9.5.14 PWM Output Mode

The PWM supports two output modes: Independent mode which may be applied to DC motor system, Complementary mode with dead-time insertion which may be used in the application of AC induction motor and permanent magnet synchronous motor.

6.9.5.15 Independent mode

By default, the PWM is operating in independent mode, independent mode is enabled when channel n corresponding OUTMODEn (PWM_CTL1[26:24]) bit is set to 0. In this mode six PWM channels: PWM_CH0, PWM_CH1, PWM_CH2, PWM_CH3, PWM_CH4 and PWM_CH5 are running off its own period and duty as shown in Figure 6.9-19.

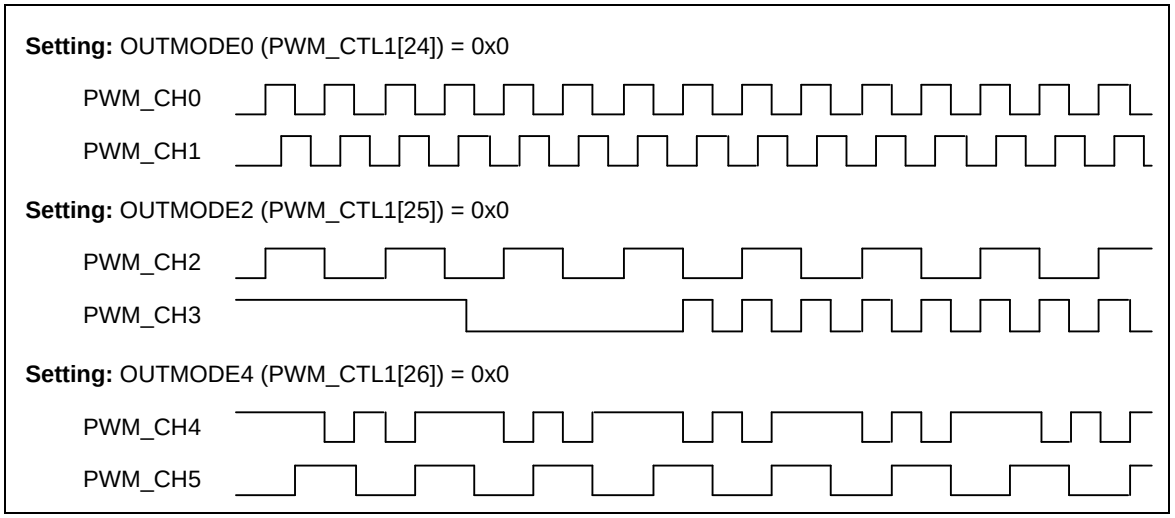


Figure 6.9-19 PWM Independent Mode Waveform

6.9.5.16 Complementary mode

Complementary mode is enabled when the pair channel corresponding OUTMODEn (PWM_CTL1[26:24]) bit set to 1. In this mode there are 3 PWM generators utilized for complementary mode, with total of 3 PWM output paired pins in this module. In Complimentary modes, the internal odd PWM signal must always be the complement of the corresponding even PWM signal. PWM_CH1 will be the complement of PWM_CH0. PWM_CH3 will be the complement of PWM_CH2 and PWM_CH5 will be the complement of PWM_CH4 as shown in Figure 6.9-20.

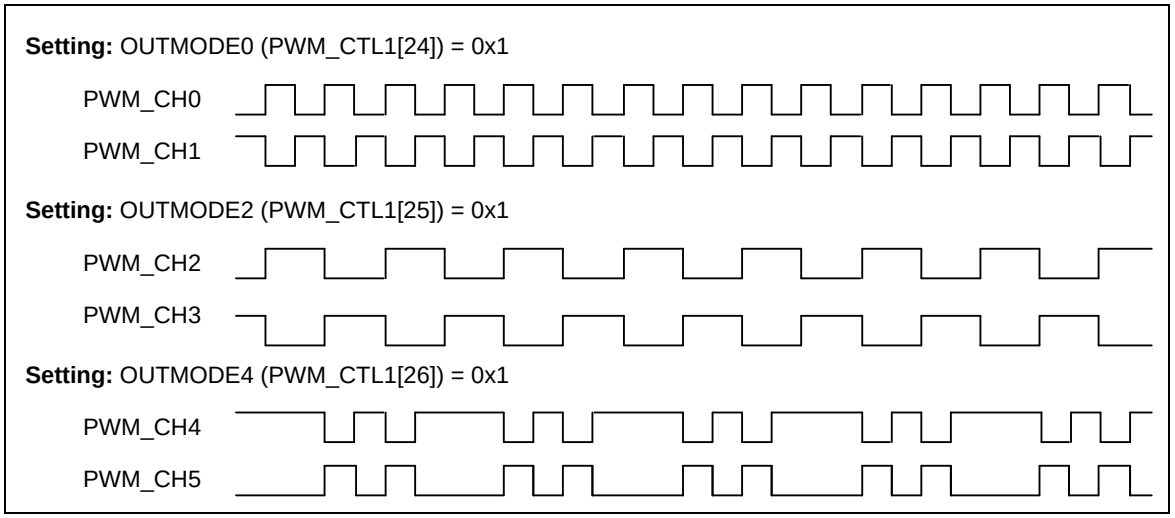


Figure 6.9-20 PWM Complementary Mode Waveform

6.9.5.17 PWM Output Function

Based on the output mode, there are two output functions: group and synchronous functions for advanced output control. Group function, forces the PWM_CH2 and PWM_CH4 synchronous with PWM_CH0 generator and forces the PWM_CH3 and PWM_CH5 synchronous with PWM_CH1, may simplify updating duty control in DC and BLDC motor applications. Besides, Synchronous function makes any channel of PWM0 and PWM1 in phase, user can control phase value and direction.

6.9.5.18 Group function

Group function is enabled when GROUPEN (PWM_CTL0[24]) set to 1, no matter in independent or complementary mode. This control allows all even PWM channels output to be controllable by PWM_PERIOD0 and PWM_CMPDAT0 registers and all odd PWM channels output to be controllable by PWM_PERIOD1 and PWM_CMPDAT1 registers. That is, user only needs to set PWM_CH0 to get PWM_CH0, PWM_CH2 and PWM_CH4 output the same pulse, and set PWM_CH1 to get PWM_CH1, PWM_CH3 and PWM_CH5 output the same pulse, as shown in Figure 6.9-21. When operating group function, OUTMODE0, OUTMODE2 and OUTMODE4 must all set to 0 for independent mode or all set to 1 for complementary mode.

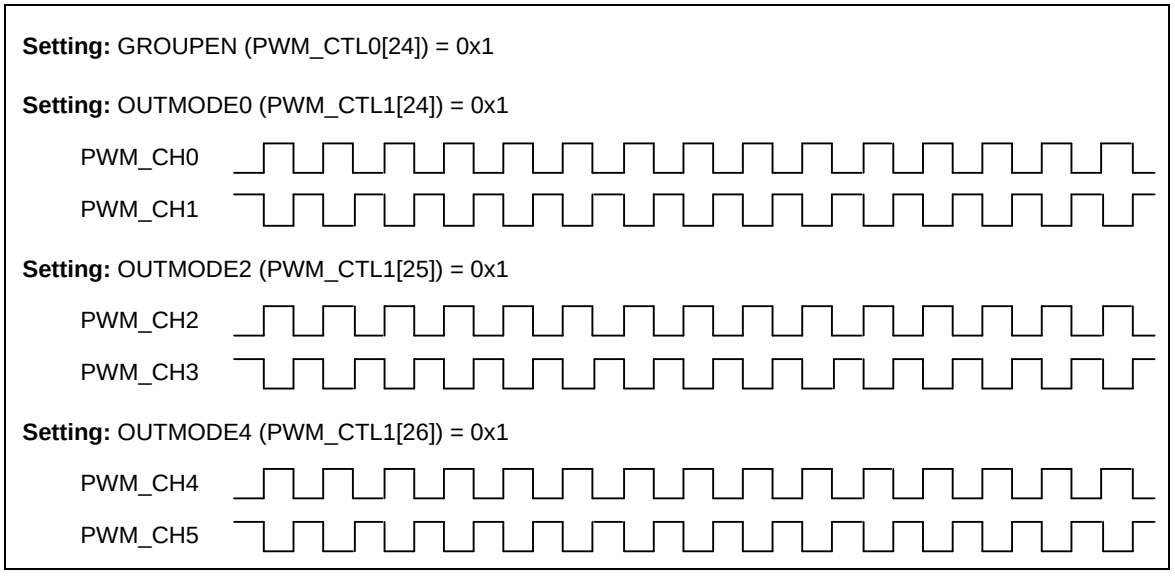


Figure 6.9-21 PWM Group Function Waveform

6.9.5.19 Synchronous function

Synchronous function can only be enabled when complementary mode is enabled. Figure 6.9-23 is counter synchronous function block diagram. Every counter of PWM pairs has a SYNC_IN and a SYNC_OUT signals. The SYNC_IN for the first PWM0 pair counter comes from GPIO pin, and the others come from the SYNC_OUT of previous PWM pair counter. The GPIO input signal will be filtered by a 3-bit noise filter as Figure 6.9-22. In addition, it can be inversed by setting the bit SINPINV (PWM_SYNC[23]) to realize the polarity setup for the input signal. The noise filter sampling clock can be selected by setting bits SFLTCSEL (PWM_SYNC[19:17]) to fit different noise properties. Moreover, by setting the bits SFLTCNT (PWM_SYNC[22:20]), user can define by how many sampling clock cycles a filter will recognize the effective edge of the SYNC_IN signal. Configuring theSNFLTEN (PWM_SYNC[16]) will enable the noise filter function. By default, it is disabled.

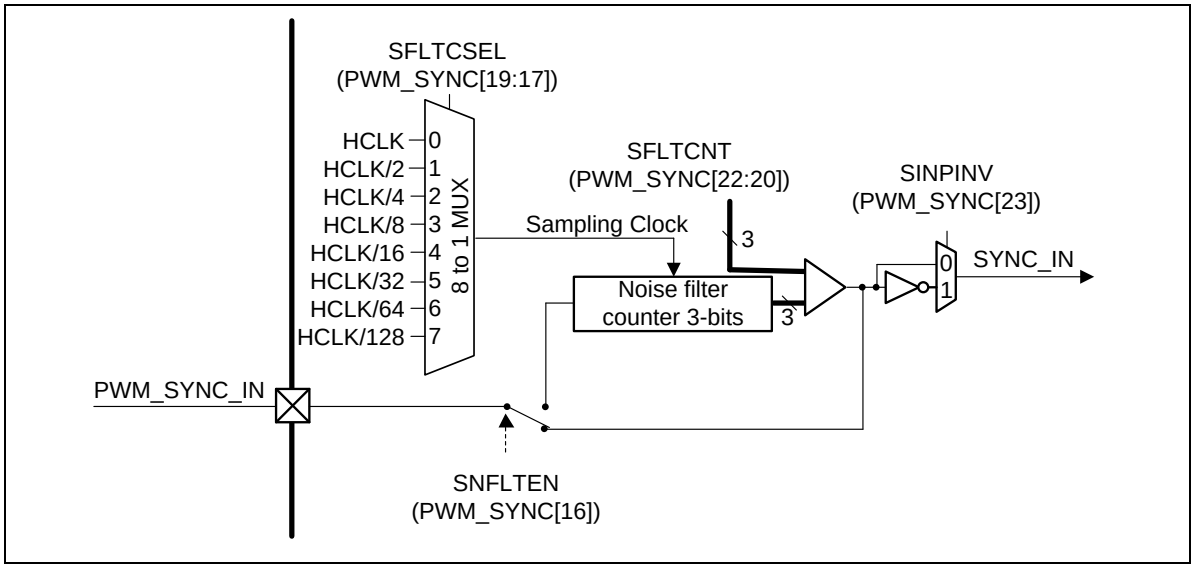


Figure 6.9-22 PWM SYNC_IN Noise Filter Block Diagram

The SYNC_OUT of the first PWM0 pair counter outputs not only to the next PWM0 pair counter SYNC_IN, but also outputs to PWM0_SYNC_OUT pin for different chip counters synchronization and the last pair of PWM0 will generate SYNC_OUT signal to the first pair counter of PWM1. By default setting, SYNC_IN “OR” SWSYNCn (PWM_SWSYNC[2:0]) to generate the synchronizing source for the next counter’s synchronization . User can use SINSRCn (PWM_SYNC[13:8]) to select the SYNC_OUT source. Synchronizing source can also be selected as CNT = 0 or CNT = PWM_CMPDATm (if being the up-down counter type, it will synchronize twice in a PWM period) to trigger a sync event or to disable SYNC_OUT (SYNC_OUT = 0).

When the PHSENn (PWM_SYNC[2:0]) is enabled and the synchronous source has a happening event, the counter will load a value from the PHS (PWM_PHS[15:0]) register. This method synchronizes counters to different phase in the same time. In the up-down counter type, user can set the value in PHSDIRn (PWM_SYNC[26:24]) to control the counter direction after synchronization. Although the Synchronous function can synchronize channels in phase, it can’t work from the beginning of PWM enable. To start these counters in the same time, user have to set the PWM Synchronous Start Control Register (PWM_SSCTL[5:0]) to enable the channel counters which are planned to sync together, followed by setting the PWM Synchronous Start Trigger Register CNTSEN (PWM_SSTRG[0]).

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

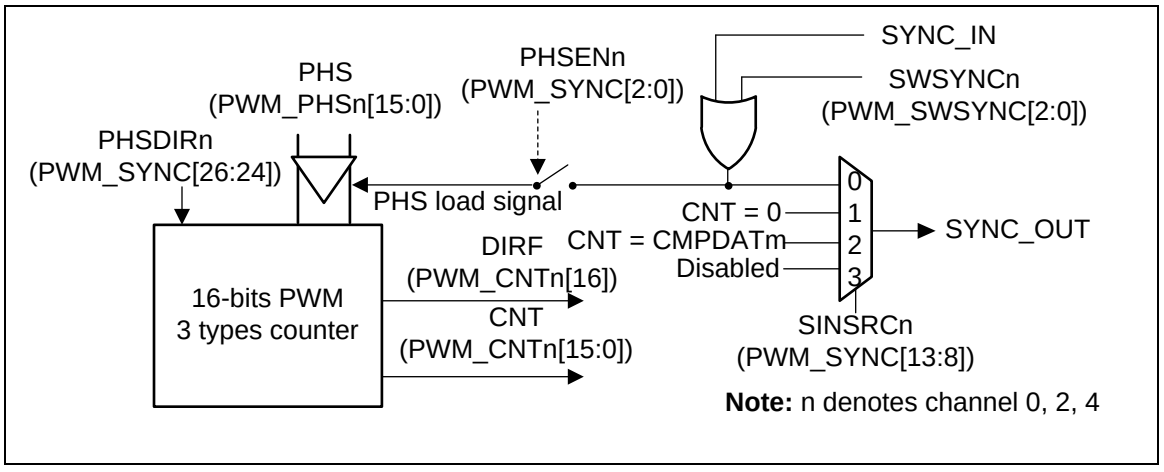


Figure 6.9-23 PWM Counter Synchronous Function Block Diagram

Figure 6.9-24 is an example of the synchronous function in the up-down counter type. In the example, synchronizing source comes from the external PWM SYNC_IN. At the beginning, PWM_CH0, PWM_CH2 and PWM_CH4 are in the same phase. Then at Point A, the PWM SYNC input signal comes as a sync event, resulting in phase shifts and counting direction changes for all of the counters. To realize the altered counter behaviors before the sync event coming, user has to setup the corresponding phase value in the PHS of(PWM_PHSn[15:0]) as well as the counting direction in the PHSDIRn (PWM_SYNC[26:24]). In this case, one third of phase shifts are made. by setting the corresponding channel n’s counter counting direction after synchronizing, as illustrated around the left side of the figure.

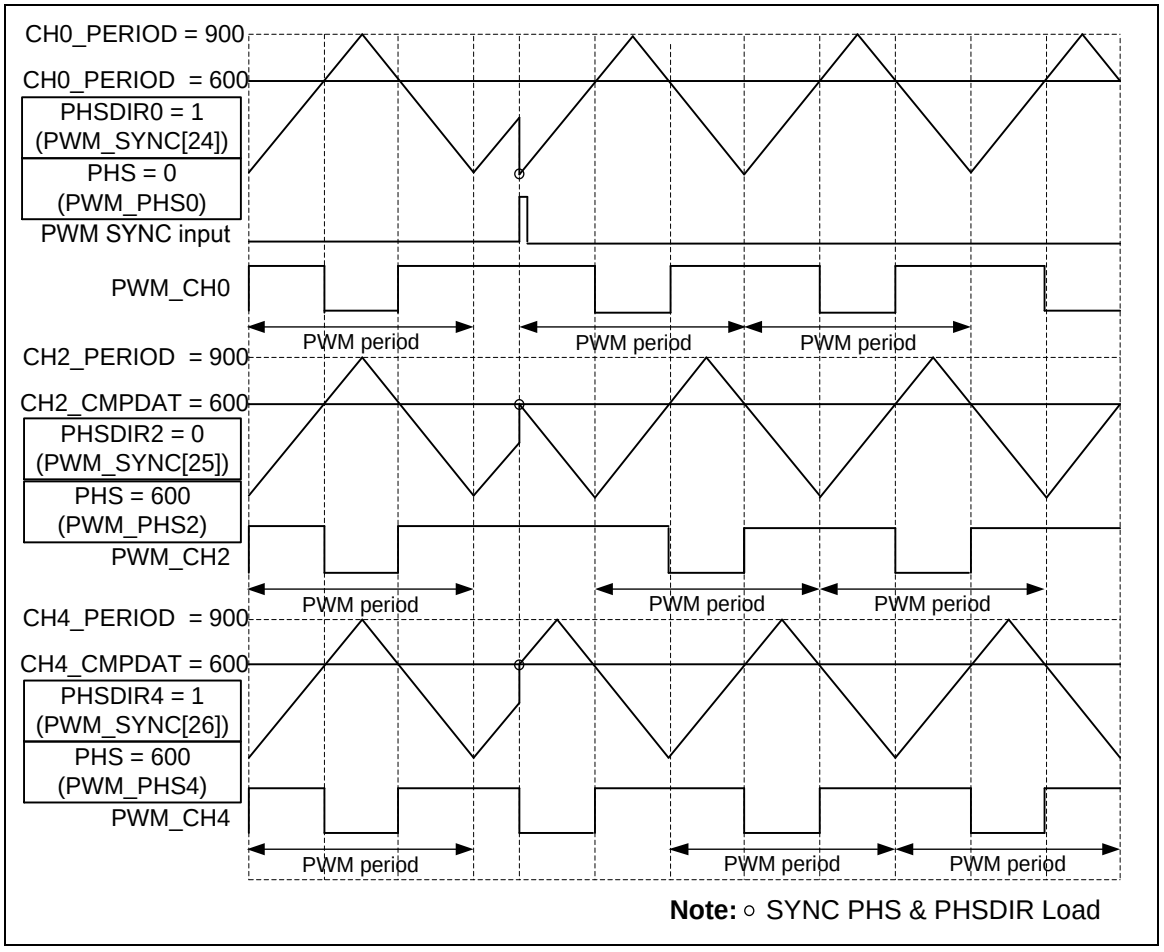


Figure 6.9-24 PWM Synchronous Function with SINSRC=0

6.9.5.20 PWM Output Control

After PWM pulse generation, there are four to six steps to control the output of PWM channels. In independent mode, there are Mask, Brake, Pin Polarity and Output Enable four steps as shown in Figure 6.9-25. In complementary mode, it needs two more steps to precede these four steps, Complementary channels and Dead-Time Insertion as shown in Figure 6.9-26.

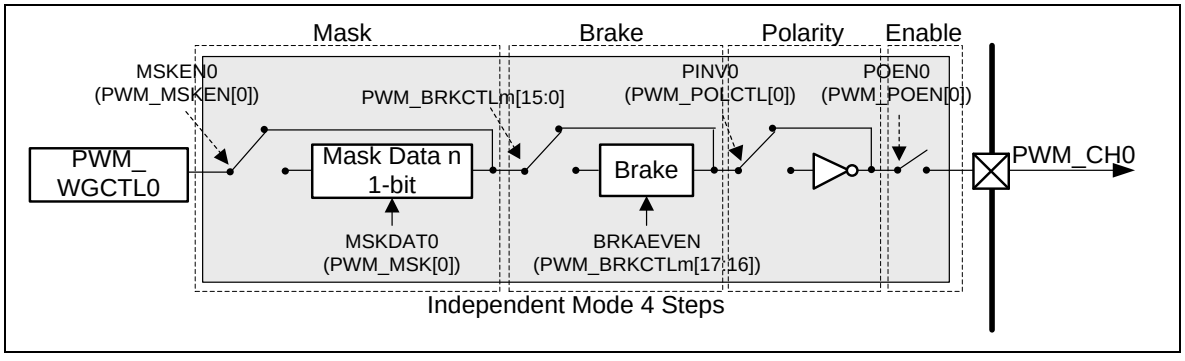


Figure 6.9-25 PWM_CH0 Output Control in Independent Mode

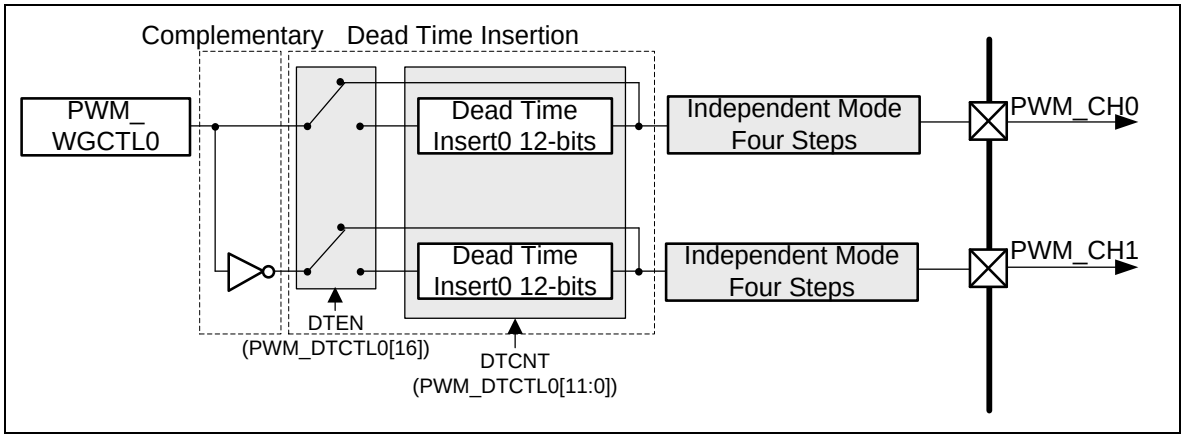


Figure 6.9-26 PWM_CH0 and PWM_CH1 Output Control in Complementary Mode

6.9.5.21 Dead-Time Insertion

In the complementary application, the complement channels may drive the external devices like power switches. The dead-time generator inserts a low level period called “dead-time” between complementary outputs to drive these devices safely and to prevent system or devices from the burn-out damage. Hence the dead-time control is a crucial mechanism to the proper operation of the complementary system. By setting corresponding channel n DTEN (PWM_DTCTLn[16]) bit to enable dead-time function and DTCNT (PWM_DTCTLn[11:0]) to control dead-time period, the dead-time can be calculated from the following formula:

Dead-time = (DTCNT (PWM_DTCTLn[11:0])+1) * PWMx_CLK period

Please note that the PWM_DTCTLn is a write-protected register.

Figure 6.9-27 indicates the dead-time insertion for one pair of PWM signals.

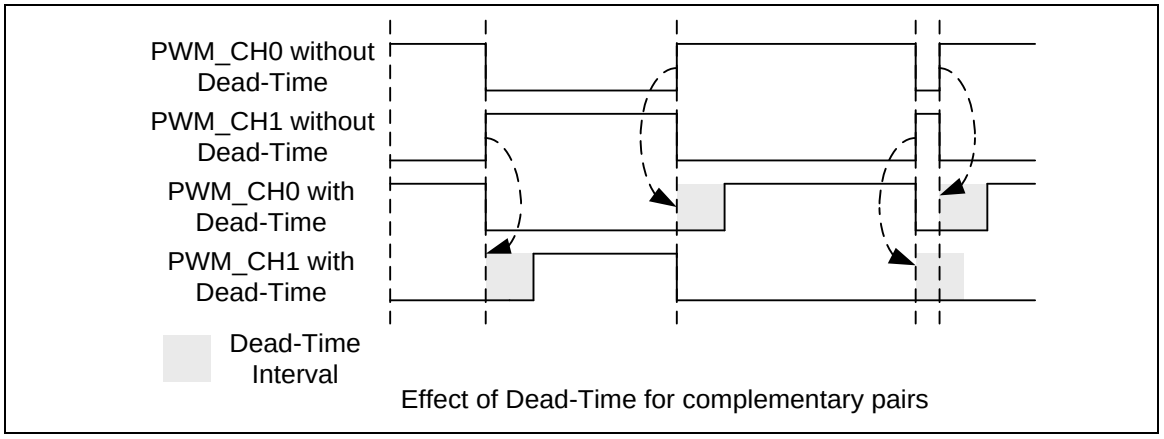


Figure 6.9-27 Dead-Time Insertion

6.9.5.22 PWM Mask Output Function

Each of the PWM channel output value can be manually overridden with the settings in the PWM Mask Enable Control Register (PWM_MSKEN) and the PWM Masked Data Register

(PWM_MSK) With these settings, the PWM channel outputs can be assigned to specified logic states independent of the duty cycle comparison units. The PWM mask bits are useful when controlling various types of Electrically Commutated Motor (ECM) like a BLDC motor. The PWM_MSKEN register contains six bits, MSKENn(PWM_MSKEN[5:0]). If the MASKENn is set to active-high, the PWM channel n output will be overridden. The PWM_MSK register contains six bits, MSKDATn(PWM_MSK[5:0]). The bit value of the MSKDATn determines the state value of the PWM channel n output when the channel is overridden. Figure 6.9-28 shows an example of how PWM mask control can be used for the override feature.

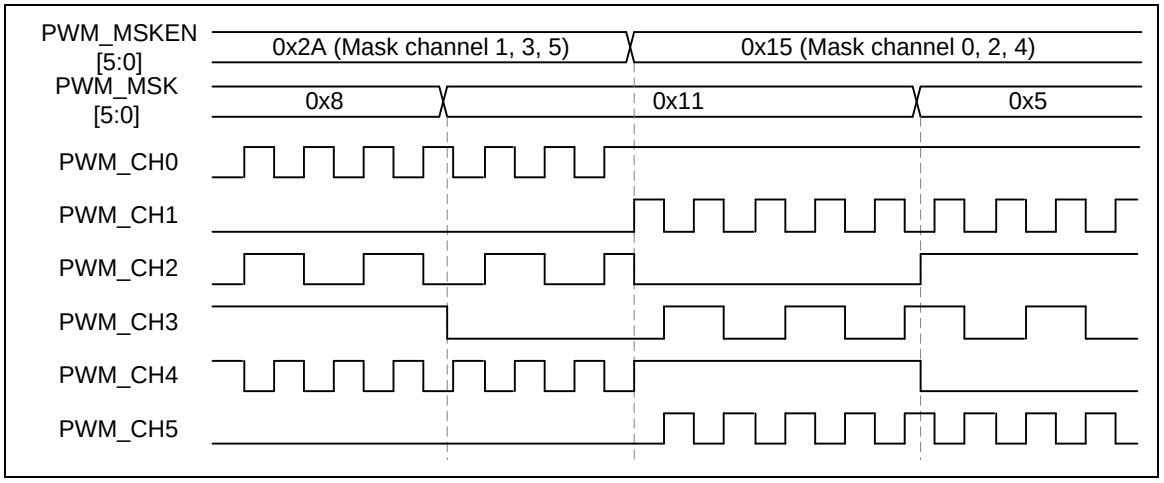


Figure 6.9-28 Illustration of Mask Control Waveform

6.9.5.23 PWM Brake

Each PWM module has two external input brake control signals. The external signals will be filtered by a 3-bit noise filter. In addition, it can be inversed by setting the bit BRKxPINV (PWM_BNF[15, 7], x denotes input external pin 0 or 1) to realize the polarity setup for the brake control signals. The noise filter sampling clock can be selected by setting bits BRKxFCS (PWM_BNF[11:9, 3:1]) to fit different noise properties. Moreover, by setting the bits BRKxFCNT (PWM_BNF[14:12, 6:4]), user can define by how many sampling clock cycles a filter will recognize the effective edge of the brake signal. Configuring the BRKxNFEN (PWM_BNF[8, 0]) will enable the noise filter function. By default, it is disabled. As shown in **錯誤! 找不到參照來源。**.

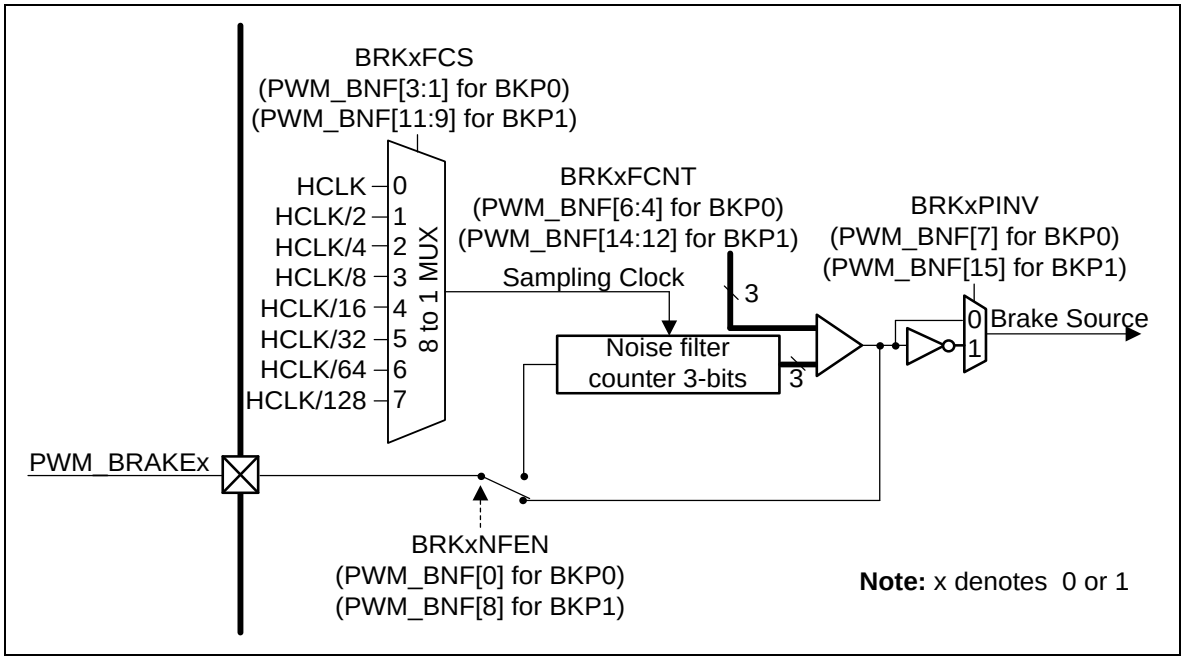


Figure 6.9-29 Brake Noise Filter Block Diagram

For Complementatry mode, it is often necessary to set a safe output state to the complement output pairs once the brake event occurs.

Each complementary channel pair shares a PWM brake function, as shown Figure 6.9-30. To control paired channels to output safety state, user can setup BRKAEVEN (PWM_BRKCTL0_1[17:16]) for even channels and BRKAODD (PWM_BRKCTL0_1[19:18]) for odd channels when the fault brake event happens. There are two brake detectors: Edge detector and Level detector. When the edge detector detects the brake signal and BRKEIENn_m (PWM_INTEN1[2:0]) is enabled, the brake function generates BRK_INT. This interrupt needs software to clear, and the BRKESTSn (PWM_INTSTS1[21:16]) brake state will keep until the next PWM period starts after the interrupt cleared. The brake function can also operate in another way through the level detector. Once the level detector detects the brake signal and the BRKLIENn_m (PWM_INTEN1[10:8]) is also enabled, the brake function will generate BRK_INT, but BRKLSTSn (PWM_INTSTS1[29:24]) brake state will auto recovery to normal output while level brake source recovery to high level and pass through “Low Level Detection” at the PWM waveform period when brake condition removed without clear interrupt.

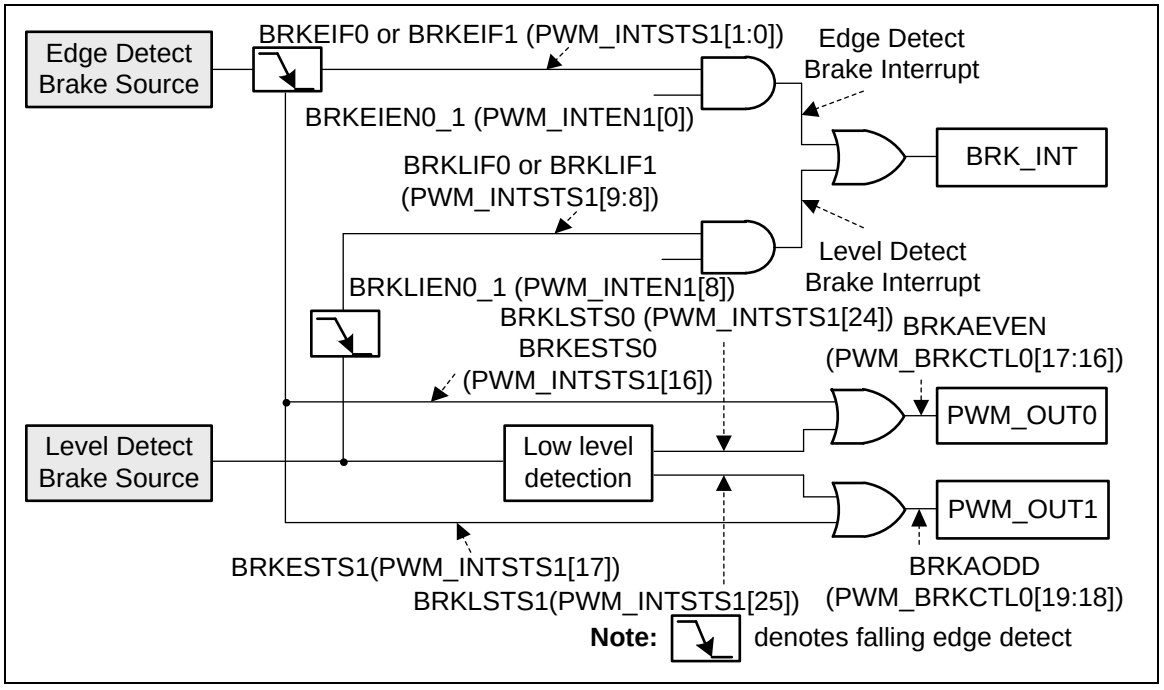


Figure 6.9-30 Brake Block Diagram for PWM_CH0 and PWM_CH1 Pair

Figure 6.9-31 illustrates the edge detector waveform for PWM_CH0 and PWM_CH1 pair. In this case, the edge detect brake source has occurred twice for the brake events. When the event occurs, both of the BRKEIF0 and BRKEIF1 flags are set and BRKESTS0 and BRKESTS1 are also set to indicate brake state of PWM_CH0 and PWM_CH1. For the first occurring event, software writes 1 to clear the BRKEIF0. After that, the BRKESTS0 is cleared by hardware at the next start of the PWM period. At the same moment, the PWM_CH0 outputs the normal waveform even though the brake event is still occurring. The second event also triggers the same flags, but at this time, software writes 1 to clear the BRKEIF1. Afterward, PWM_CH1 outputs normally at the next start of the PWM period.

As a contrast to the edge detector example, Figure 6.9-32 illustrates the level detector waveform for PWM_CH0 and PWM_CH1 pair. In this case, the BRKLIF0 and BRKLIF1 can only indicate the brake event having occurred. The BRKLSTS0 and BRKLSTS1 brake states will automatically recover at the start of the next PWM period no matter at what states the BRKLIF0 and BRKLIF1 are at that moment.

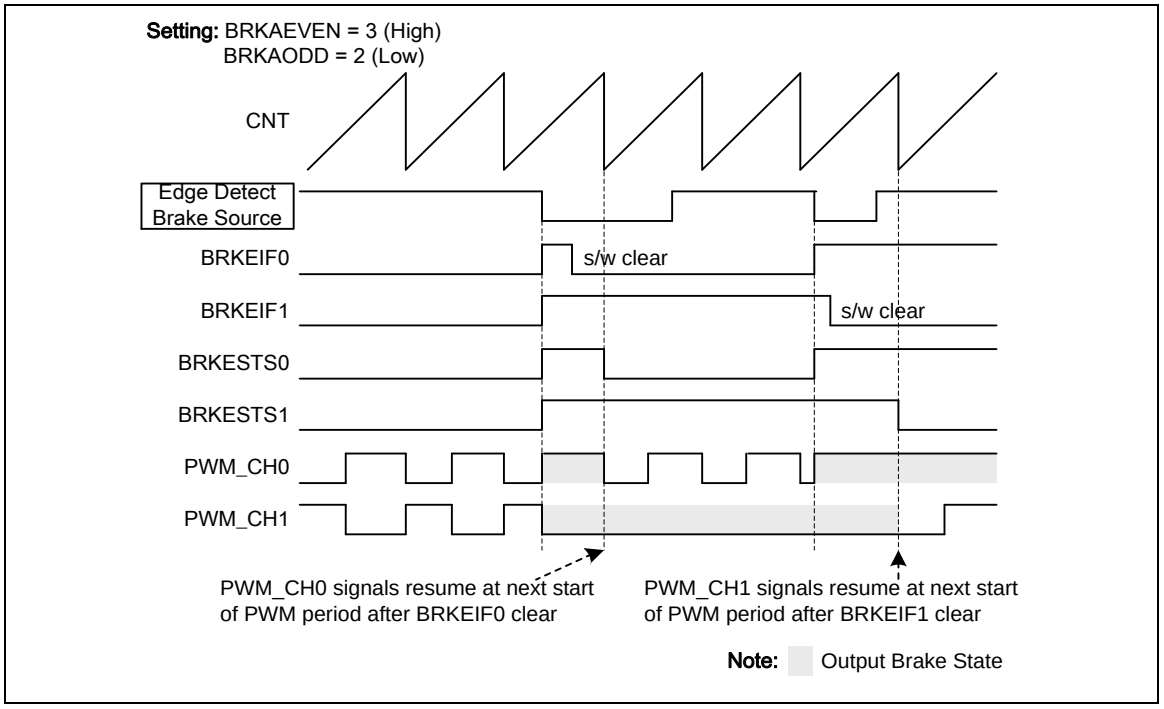


Figure 6.9-31 Edge Detector Waveform for PWM_CH0 and PWM_CH1 Pair

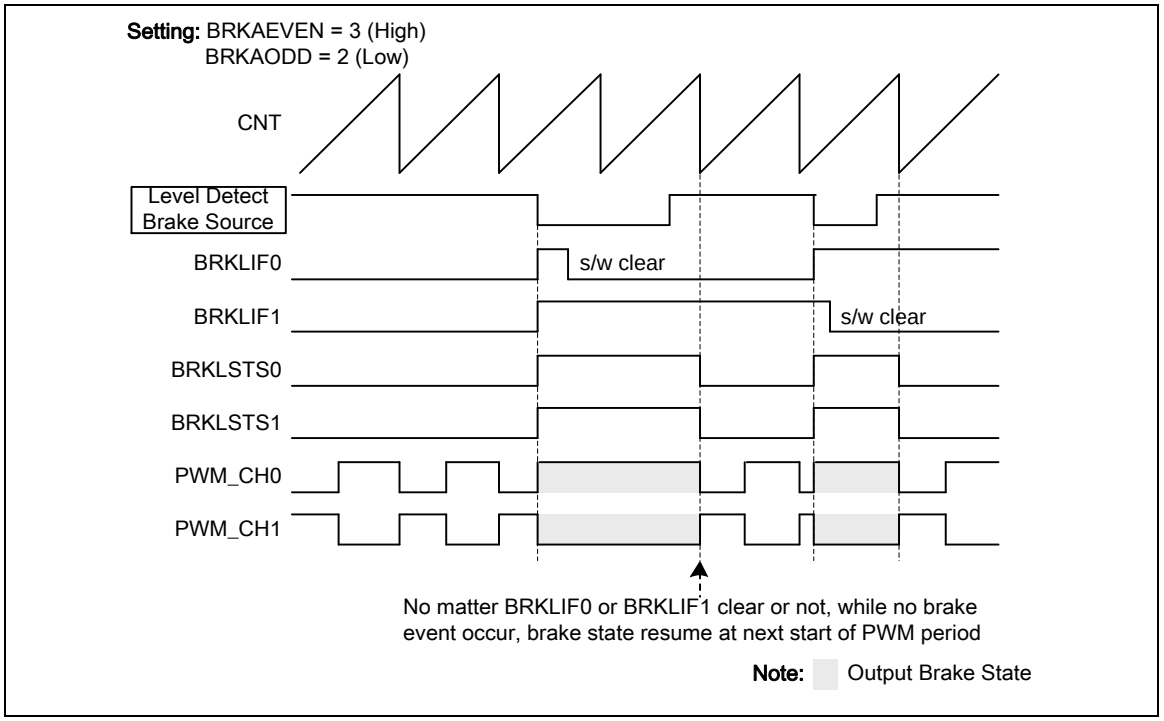


Figure 6.9-32 Level Detector Waveform for PWM_CH0 and PWM_CH1 Pair

Among the above described brake sources, the brake source coming from system fail can still be specified to several different system fail conditions. These conditions include clock fail, Brown-out detect, SRAM parity check error and Cortex™-M4 lockup. Figure 6.9-33 shows that by setting

corresponding enable bits, the enabled system fail condition can be one of the sources to issue the Brake system fail to the PWM brake.

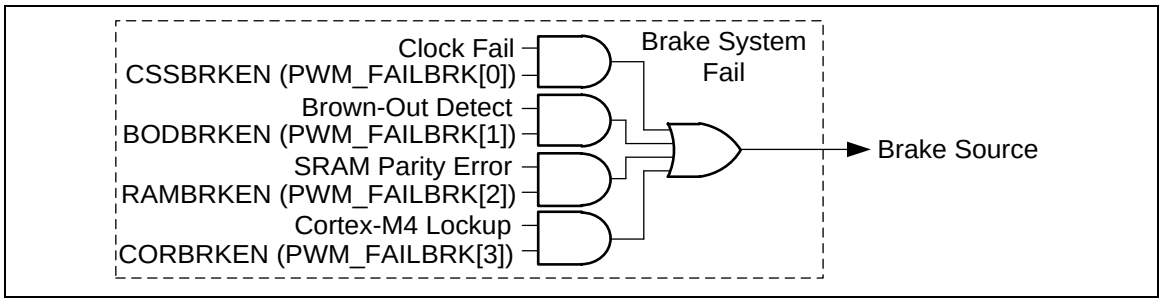


Figure 6.9-33 Brake System Fail Block Diagram

6.9.5.24 Polarity Control

Each PWM port, from PWM_CH0 to PWM_CH5, has an independent polarity control module to configure the polarity of the active state of the PWM output. By default, the PWM output is active high. This implies the PWM OFF state is low and ON state is high. This definition is variable through setting the PWM Negative Polarity Control Register (PWM_POLCTL), for each individual PWM channel. Figure 6.9-34 shows the initial state before PWM starting with different polarity settings.

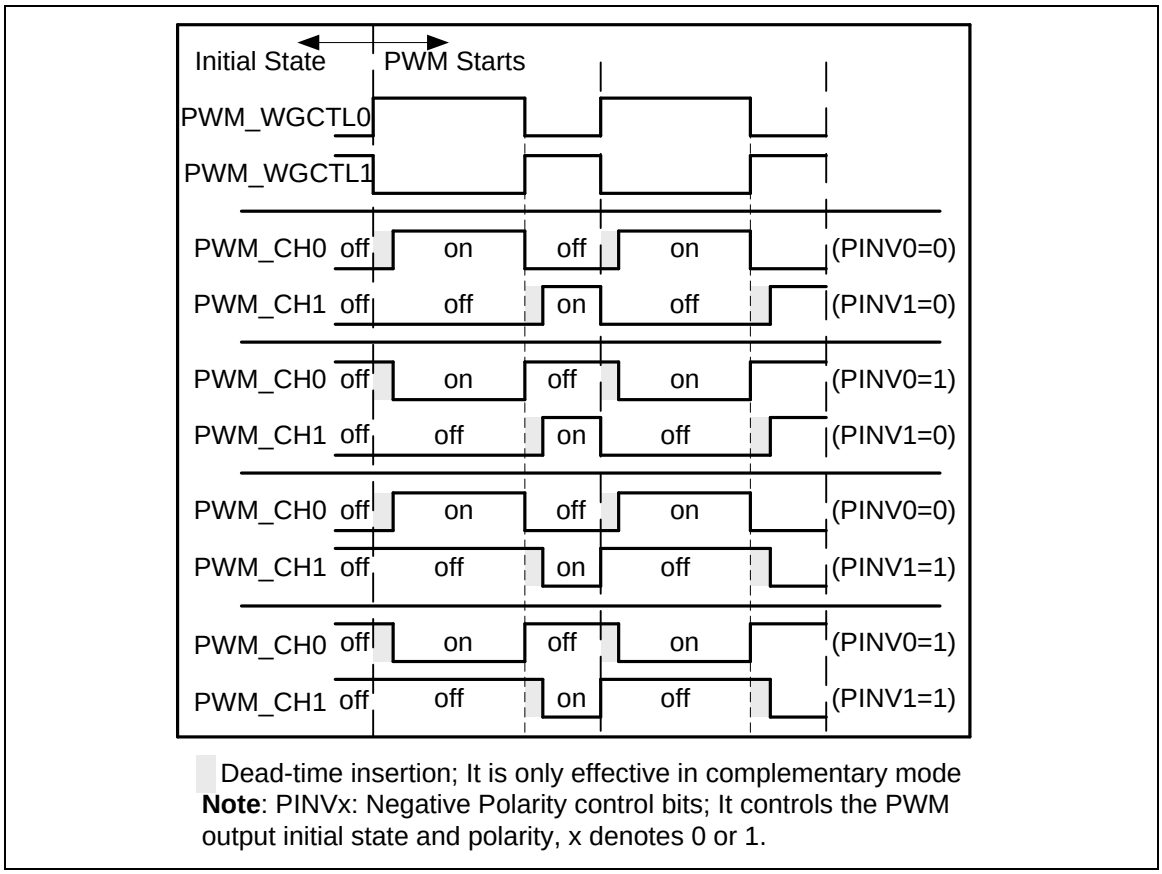


Figure 6.9-34 Initial State and Polarity Control with Rising Edge Dead-Time Insertion

6.9.5.25 PWM Interrupt Generator

There are three independent interrupts for each PWM as shown in Figure 6.9-36.

The 1st PWM interrupt (PWM_INT) comes from PWM complementary pair events. The counter can generate the Zero point Interrupt Flag ZIFn (PWM_INTSTS0[5:0]) and the Period point Interrupt Flag PIFn (PWM_INTSTS0[13:8]). When PWM channel n's counter equals to the comparator value stored in PWM_CMPDATn, the different interrupt flags will be triggered depending on the couting direction. If the matching occurs at up–count direction, the Up Interrupt Flag CMPUIFn (PWM_INTSTS0[21:16]) is set and if matching at the opposite direction, the Down Interrupt Flag CMPDIFn (PWM_INTSTS0[29:24]) is set. Channel n's complementary channel m's comparator also generates the CMPUIFm and CMPDIFm in the same way. If the corresponding interrupt enable bits are set, the trigger events will generates interrupt signals.

PWM_INT can use the register PWM_IFA to accumulate the number of times the interrupt flags have been triggerred. By setting IFAENn_m (IFAEN0_1(PWM_IFA[7]), IFAEN2_3 (PWM_IFA[15]) and IFAEN4_5 (PWM_IFA[23])) to 1 to enable accumulator. When the accumulator is enabled, PWM_INT will switch interrupt source from every event trigger interrupt to trigger interrupt once every accumulate times.

By setting the IFSELn_m (IFSEL0_1 (PWM_IFA[6:4]), IFSEL2_3 (PWM_IFA[14:12]) and IFSEL4_5 (PWM_IFA[22:20])), user can select one of the 8 interrupt sources to accumulate, and compare with 4 bits IFCNTn_m (IFCNT0_1 (PWM_IFA[3:0]), IFCNT2_3 (PWM_IFA[11:8]) and IFCNT4_5 (PWM_IFA[19:16])), when interrupt accumulator equals IFCNTn_m then set IFAIFn_m (IFAIF0_1 (PWM_INTSTS0[7]), IFAIF2_3 (PWM_INTSTS0[15]) and IFAIF4_5 (PWM_INTSTS0[23])) as PWM_INT signal when enable IFAIENn_m (IFAIEN0_1 (PWM_INTEN0[7]), IFAIEN2_3 (PWM_INTEN0[15]) and IFAEN4_5 (PWM_INTEN0[23])). Figure 6.9-35 is an example of channel 0 and channel 1 pair using PWM_IFA register to output PWM_INT once every IFCNT0_1+1 times interrupt events occurred.

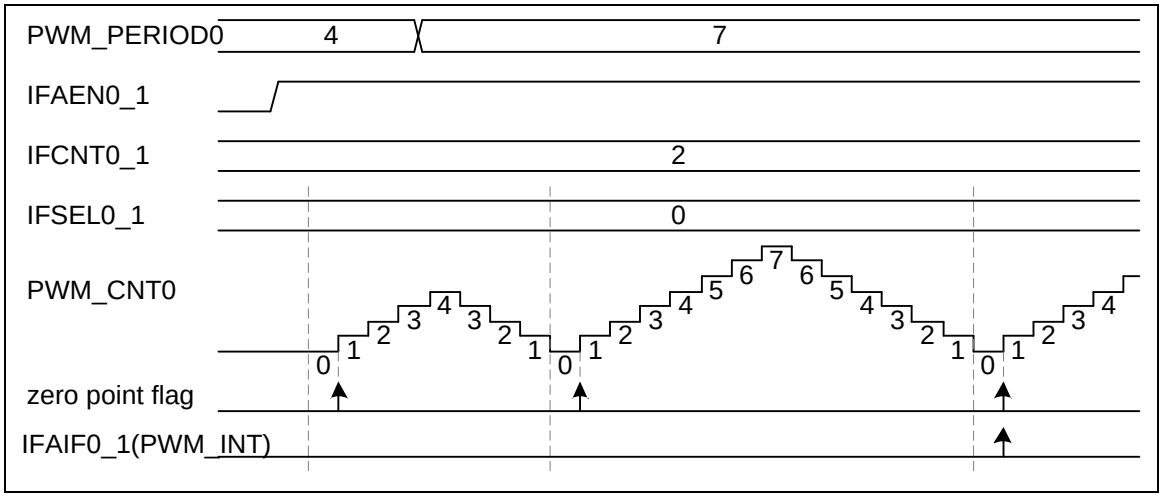


Figure 6.9-35 PWM_CH0 and PWM_CH1 Pair Accumulate Interrupt Waveform

The 2nd interrupt is the capture interrupt (CAP_INT). It shares the PWM_INT vector in NVIC. The CAP_INT can be generated when the CRLIFn (PWM_CAPIF[5:0]) is triggered and the Capture Rising Interrupt Enable bit CAPRIENn (PWM_CAPIEN[5:0]) is set to 1. Or in the falling edge condition, the CFLIFn (PWM_CAPIF[13:8]) can be triggered when the Capture Falling Interrupt Enable bit CAPFIENn (PWM_CAPIEN[13:8]) is set to 1.

The last one is the brake interrupt (BRK_INT). The details of the BRK_INT is described in the

PWM Brake section.

The Figure 6.9-36 demonstrates the architecture of the PWM interrupts.

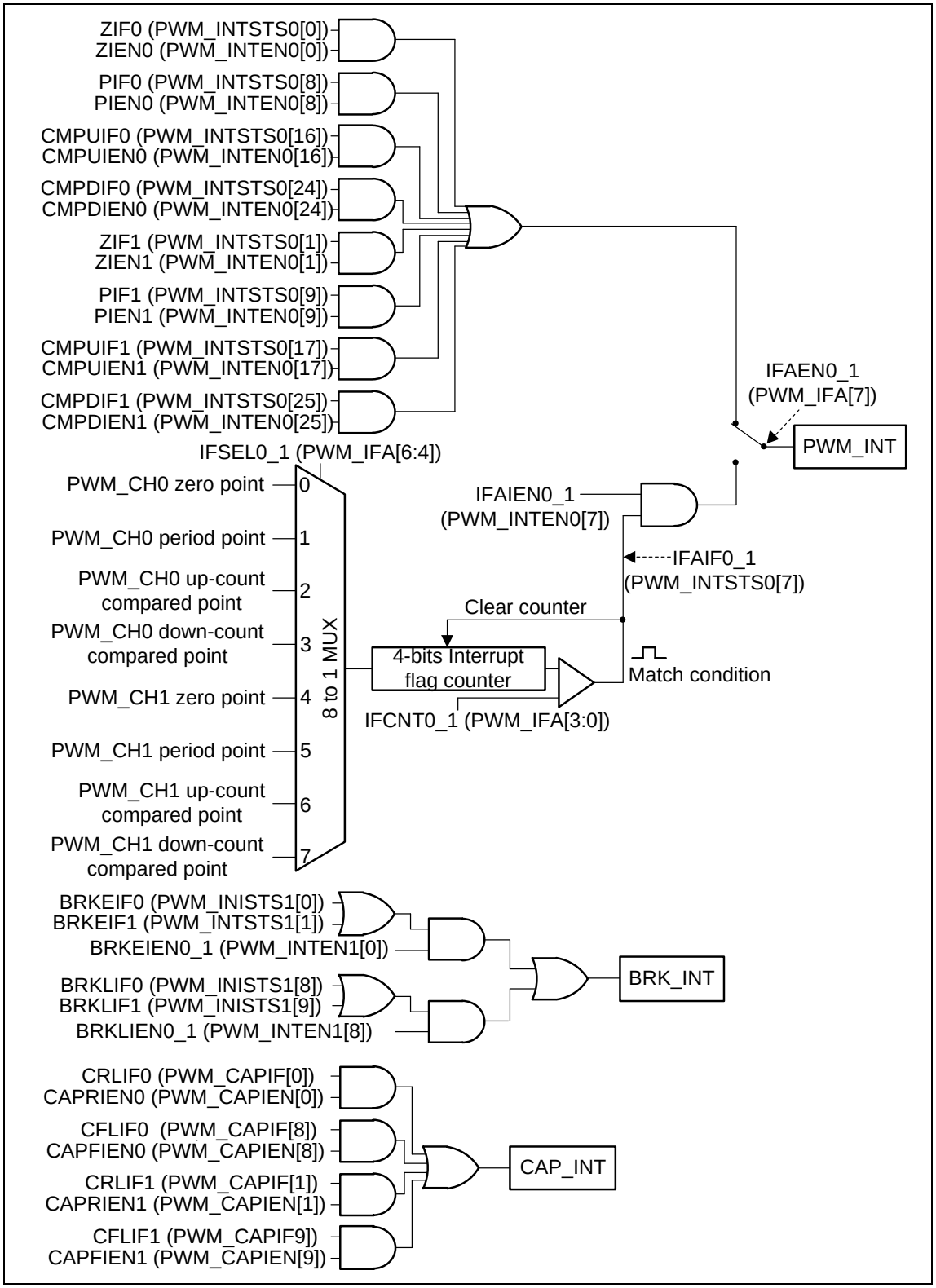


Figure 6.9-36 PWM_CH0 and PWM_CH1 Pair Interrupt Architecture Diagram

6.9.5.26 PWM Trigger EADC/DAC Generator

PWM can be one of the EADC conversion trigger source. Each PWM pair channels share the same trigger source. Setting TRGSELn is to select the trigger sources, where TRGSELn is TRGSEL0, TRGSEL1, …, and TRGSEL5, which are located in PWM_EADCTS0[3:0], PWM_EADCTS0[11:8], PWM_EADCTS0[19:16], PWM_EADCTS0[27:24], PWM_EADCTS1[3:0] and PWM_EADTS1[11:8], respectively. Setting TRGENn is to enable the trigger output to EADC, where TRGENn is TRGEN0, TRGEN1, …, TRGEN5, which are located in PWM_EADCTS0[7], PWM_EADCTS0[15], PWM_EADCTS0[23], PWM_EADCTS0[31], PWM_EADCTS1[7] and PWM_EADCTS1[15], respectively. The number n (n = 0,1, ..,5) denotes PWM channel number.

There are 16 PWM events can be selected as the trigger source for one pair of channels. Figure 6.9-37 is an example of PWM_CH0 and PWM_CH1. PWM can trigger EADC to start conversion in different timings by setting PERIOD, CMPDAT and FTCMPDAT (FTCMPDAT only use to trigger EADC). Figure 6.9-38 is the trigger EADC timing waveform in the up-down counter type.

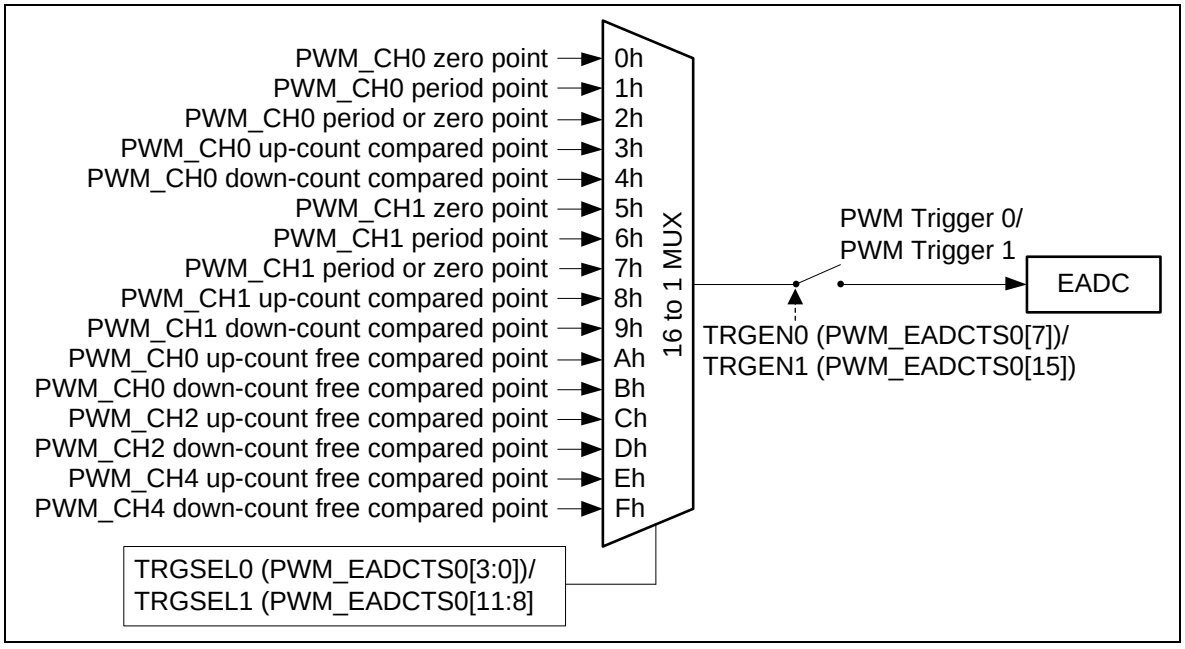


Figure 6.9-37 PWM_CH0 and PWM_CH1 Pair Trigger EADC Block Diagram

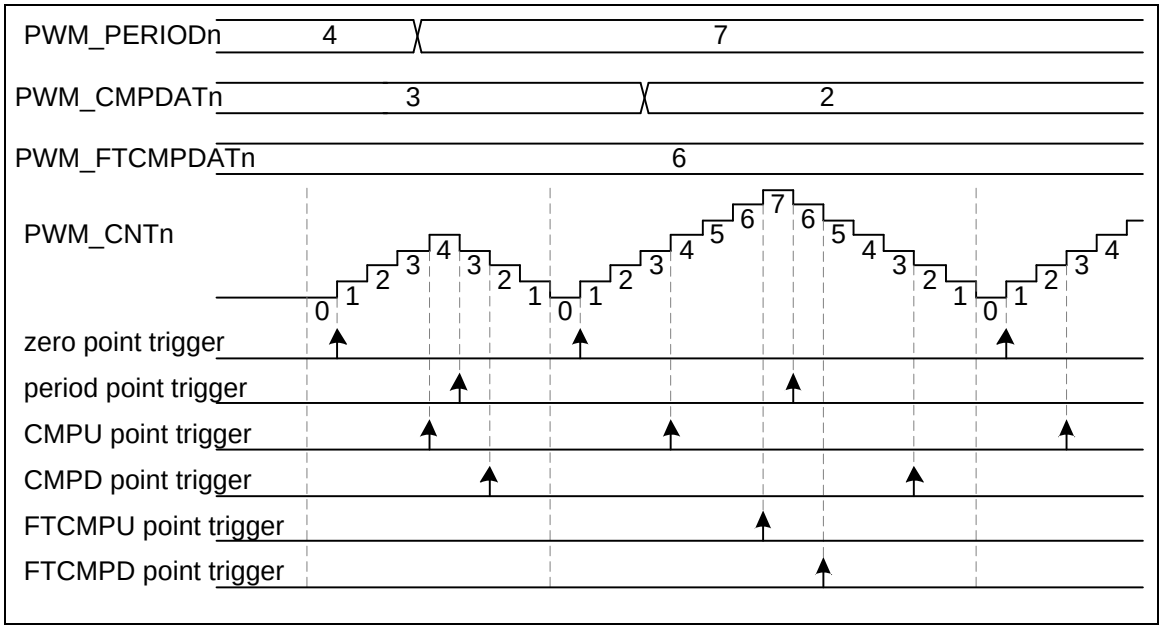


Figure 6.9-38 PWM Trigger EADC in Up-Down Counter Type Timing Waveform

PWM can also be used to trigger DAC conversion. Each PWM pair channel (CH0 and CH1, CH2 and CH3, CH4 and CH5) generates a trigger signal. Using the PWM Trigger DAC Enable Register (PWM_DACTRGEN) can decide at which points to trigger DAC. The timing of the PWM triggerring DAC is similar to those for triggerring EADC. However, DAC triggering function does not include the triggering events from comparison with FTCMPDAT, that is, there are no trigger points the same as FTCMPDATU and FTCMPDATD which are shown in EADC triggerrring.

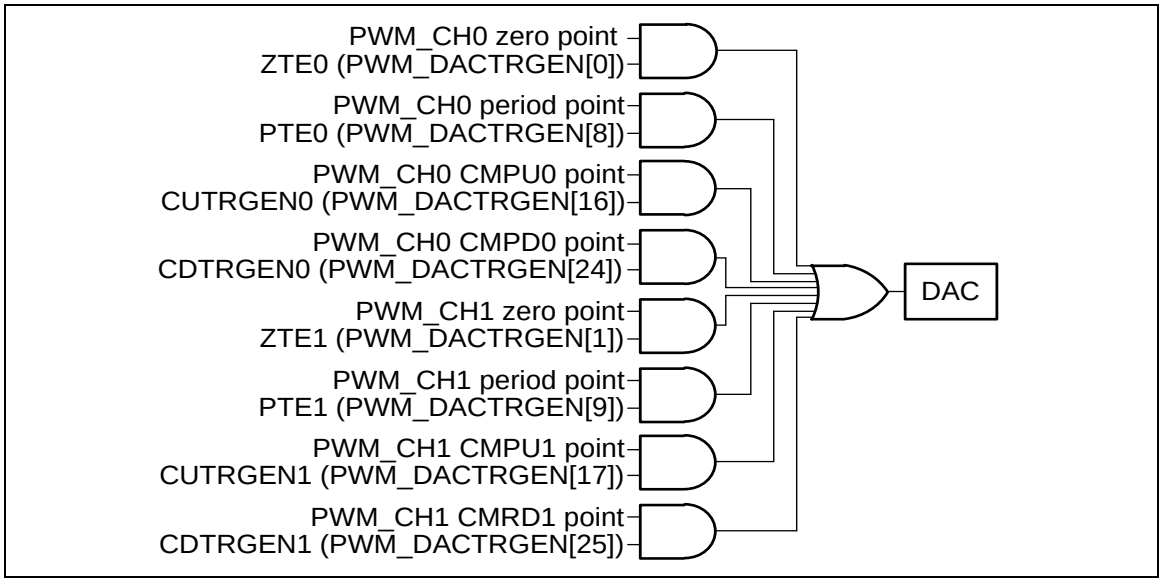


Figure 6.9-39 PWM_CH0 and PWM_CH1 Pair Trigger DAC Block Diagram

6.9.5.27 Capture Operation

The channels of the capture input and the PWM output share the same pin and counter. The

counter can operating in up or down counter type. The capture function will always latch the PWM counter to the register RCAPDATn (PWM_RCAPDATn[15:0]) or the register FCAPDATn (PWM_FCAPDATn[15:0]) if the input channel has a rising transition or a falling transition, respectively. The capture function will also generate an interrupt CAP_INT (using PWM_INT vector) if the rising or falling latch occurs and the corresponding channel n's rising or falling interrupt enable bits are set, where the CAPRIENn (PWM_CAPIEN[5:0]) is for the rising edge and the CAPFIENn (PWM_CAPIEN[13:8]) is for the falling edge. When rising or falling latch occurs, the corresponding PWM counter may be reloaded with the value PWM_PERIODn, depending on the setting of RCRLDENn or FCRLDENn (where RCRLDENn and FCRLDENn are located at PWM_CAPCTL[21:16] and PWM_CAPCTL[29:24], respectively). Note that the corresponding GPIO pins must be configured as the capture function by enable the CAPINENn (PWM_CAPINEN[5:0]) for the corresponding capture channel n. Figure 6.9-40 is the capture block diagram of channel 0.

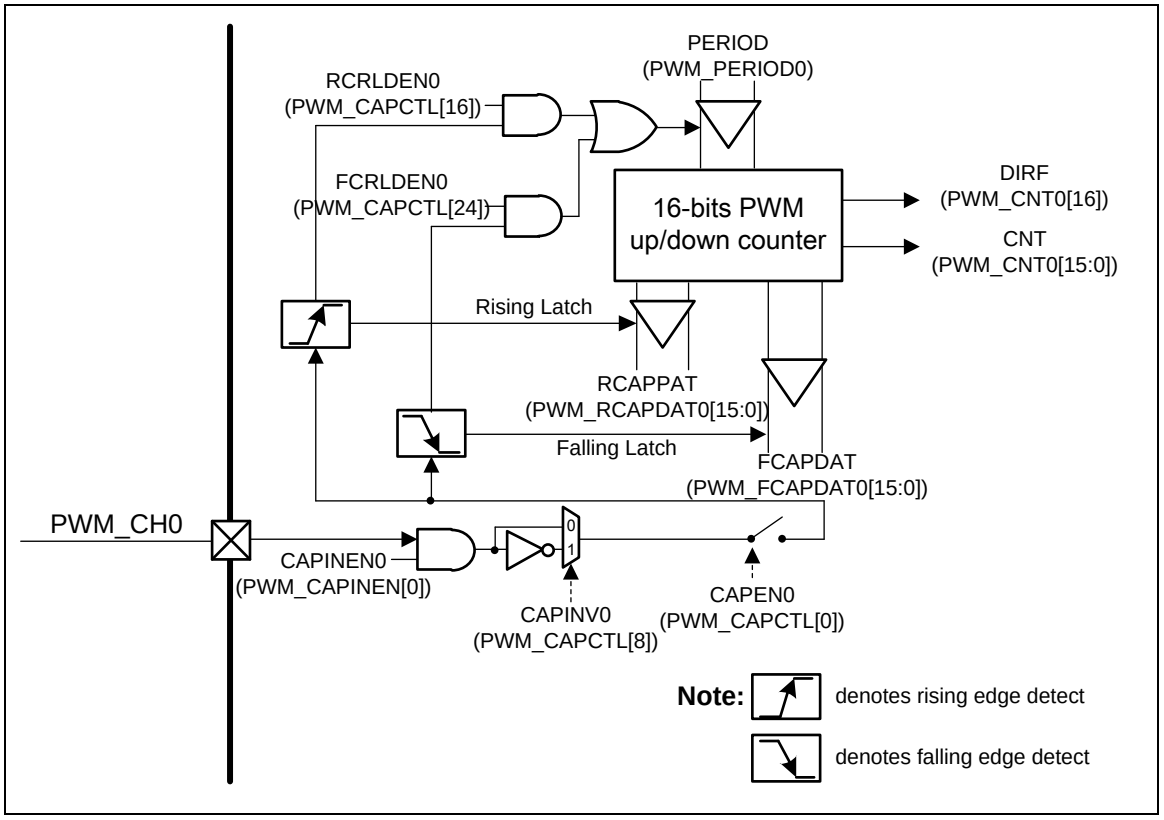


Figure 6.9-40 PWM_CH0 Capture Block Diagram

Figure 6.9-41 illustrates the capture function timing. In this case, the capture counter is set as PWM down counter type and the PERIOD is set to 8 so that the counter counts in the down direction, from 8 to 0. When detecting a falling edge at the capture input pin, the capture function latches the counter value to the PWM_FCAPDATn. When detecting the rising edge, it latches the counter value to the PWM_RCAPDATn. In this timing diagram, when the falling edge is detected at the first time, the capture function will reload the counter value from the PERIOD setting because the FCRLDENn is enabled. But at the second time, the falling edge does not result in a reload because of the disabled FCRLDENn. In this example, the counter also reloads at the rising edge of the capture input because the RCRLDENn is enabled, too.

Moreover, if the case is setup as the up counter type, the counter will reload the value zero and count up to the value PERIOD.

Figure 6.9-41 also illustrates the timing example for the interrupt and interrupt flag generation. When the rising edge at channel n is detected, the corresponding bit CRLIFn (PWM_CAPIF[5:0]) is set by hardware. Similarly, a falling edge detection at chnnel n causes the corresponding bit CFLIFn (PWM_CAPIF[13:8]) set by hardware. CRLIFn and CFLIFn can be cleared by software by writing '1'. If the CRLIFn is set and the CAPRIENn is enabled, the capture function generates an interrupt. If the CFLIFn is set and the CAPFIENn is enabled, the interrupt also happens.

A condition which is not shown in this figure is: if the rising latch happens again when the CRLIF is already set, the Over run status CRLIFOVn (PWM_CAPSTS[5:0]) will be set to 1 by hardware to indicate the CRLIF overrunning. Also, if the falling latch happens again, the same hardware operation occurs for the interrupt flag CFLIF and the Over run status CFLIFOVn (PWM_CAPSTS[13:8]).

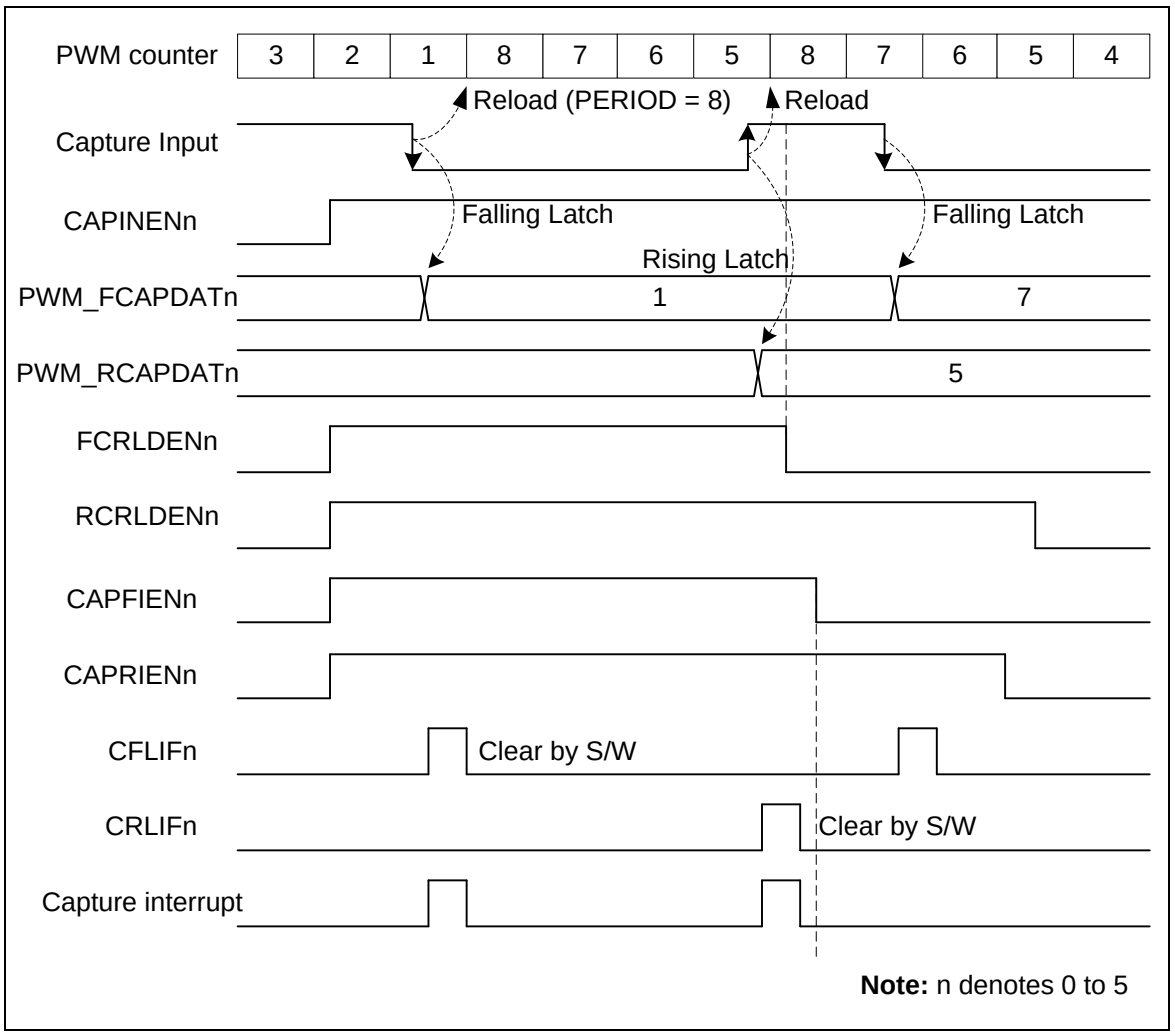


Figure 6.9-41 Capture Operation Waveform

The capture pulse width can be calculated accorting to the following formula:

For the negative pulse case, the channel low pulse width is calculated as (PWM_PERIODn + 1 - PWM_RCAPDATn). In Figure 6.9-41, the low pulse width is 8+1-5 = 4

For the positive pulse case, the channel high pulse width is calculated as (PWM_PERIODn + 1 - PWM_FCAPDATn). In Figure 6.9-41, the high pulse width is 8+1-7 = 2

6.9.5.28 Capture PDMA Function

The PWM module supports the PDMA transfer function when operating in the capture mode. When the corresponding PDMA enable bit CHENn_m (CHEN0_1 at PWM_PDMACTL[0], CHEN2_3 at PWM_PDMACTL[8] and CHEN4_5 at PWM_PDMACTL[16], where n and m denote complement pair channels) is set, the capture module will issue a request to PDMA controller when the preceding capture event has happened. The PDMA controller will issue an acknowledgement to the capture module after it has read back the CAPBUF (PWM_PDMACAPn_m[15:0], n, m denotes complement pair channels) register in the capture module and has sent the register value to the memory. By setting CAPMODn_m (CAPMOD0_1 at PWM_PDMACTL[2:1], CAPMOD2_3 at PWM_PDMACTL[10:9] and CAPMOD4_5 at PWM_PDMACTL[18:17]), the PDMA can transfer the rising edge captured data or falling edge captured data or both of them to the memory. When using the PDMA to transfer both of the falling and rising edge data, remember to set CAPORDn_m (CAPORD0_1 at PWM_PDMACTL[3], CAPORD2_3 at PWM_PDMACTL[11] and CAPORD4_5 at PWM_PDMACTL[19]) to decide the order of the transferred data (falling edge captured is first or rising edge captured first). The complement pair channels share a PDMA channel. Therefore, a selection bit CHSELn_m (CHSEL0_1 (PWM_PDMACTL[4]), CHSEL2_3 (PWM_PDMACTL[12]) and CHSEL4_5 (PWM_PDMACTL[20])) is used to decide either channel n or channel m can be serviced by the PDMA channel.

Figure 6.9-42 is capture PDMA waveform. In this case, the CHSEL0_1 (PWM_PDMACTL[4]) is set to 0. Hence the PDMA will service channel 0 for the capture data transfer. CAPMOD0_1 (PWM_PDMACTL[2:1]) is set to 3. That means both of the rising and falling edge captured data will be transferred to the memory. The CAPORD0_1 (PWM_PDMACTL[1]) is set to 1, so the rising edge data will be the first data to transfer and following is the falling edge data to transfer. As the figure shows, the last assertions of the CRLIF0 and CFLIF0 signal have some overlap. The PWM_RCAPDAT0 value 11 will be loaded to PWM_PDMACAP0_1 to wait for transfer but not the PWM_FCAPDAT0 value 6. The PWM_PDMACAP0_1 saves the data which will be transferred to the memory by PDMA. The HWDATA in this figure denotes the data which are being transferred by PDMA.

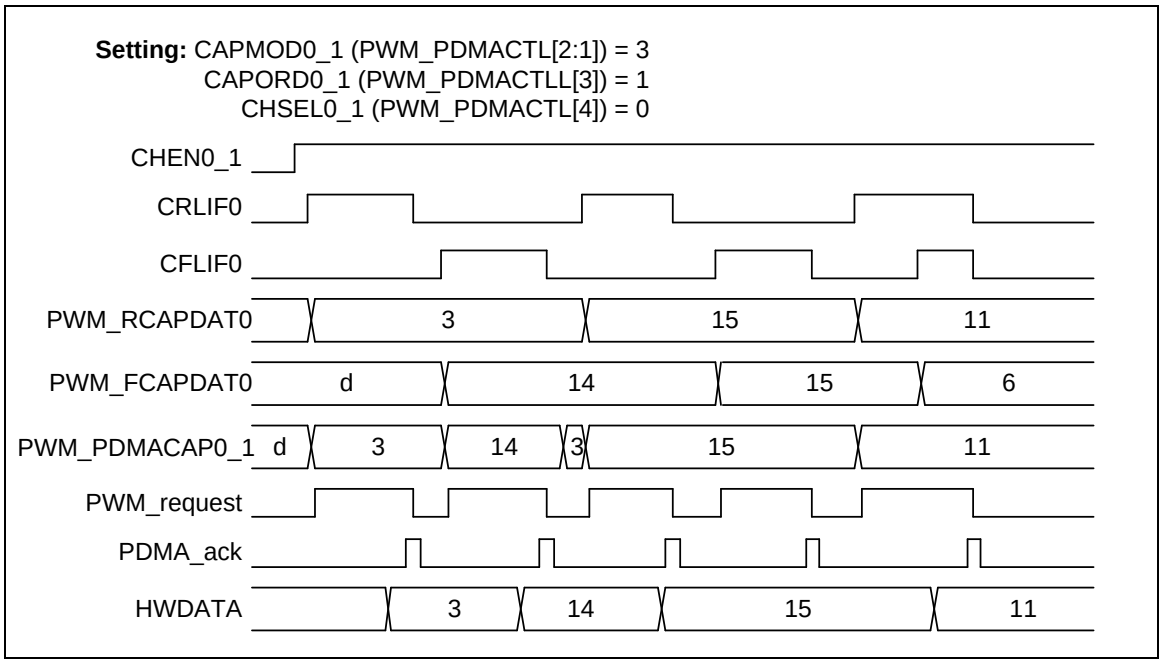


Figure 6.9-42 Capture PDMA Operation Waveform of Channel 0

6.9.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
PWM Base Address: **PWM0_BA = 0x4005_8000** **PWM1_BA = 0x4005_9000**				
PWM_CTL0 **x=0, 1**	PWMx_BA+0x00	R/W	PWM Control Register 0	0x0000_0000
PWM_CTL1 **x=0, 1**	PWMx_BA+0x04	R/W	PWM Control Register 1	0x0000_0000
PWM_SYNC **x=0, 1**	PWMx_BA+0x08	R/W	PWM Synchronization Register	0x0000_0000
PWM_SWSYNC **x=0, 1**	PWMx_BA+0x0C	R/W	PWM Software Control Synchronization Register	0x0000_0000
PWM_CLKSRC **x=0, 1**	PWMx_BA+0x10	R/W	PWM Clock Source Register	0x0000_0000
PWM_CLKPSC0_1 **x=0, 1**	PWMx_BA+0x14	R/W	PWM Clock Pre-scale Register 0	0x0000_0000
PWM_CLKPSC2_3 **x=0, 1**	PWMx_BA+0x18	R/W	PWM Clock Pre-scale Register 2	0x0000_0000
PWM_CLKPSC4_5 **x=0, 1**	PWMx_BA+0x1C	R/W	PWM Clock Pre-scale Register 4	0x0000_0000
PWM_CNTEN **x=0, 1**	PWMx_BA+0x20	R/W	PWM Counter Enable Register	0x0000_0000
PWM_CNTCLR **x=0, 1**	PWMx_BA+0x24	R/W	PWM Clear Counter Register	0x0000_0000
PWM_LOAD **x=0, 1**	PWMx_BA+0x28	R/W	PWM Load Register	0x0000_0000
PWM_PERIOD0 **x=0, 1**	PWMx_BA+0x30	R/W	PWM Period Register 0	0x0000_0000
PWM_PERIOD1 **x=0, 1**	PWMx_BA+0x34	R/W	PWM Period Register 1	0x0000_0000
PWM_PERIOD2 **x=0, 1**	PWMx_BA+0x38	R/W	PWM Period Register 2	0x0000_0000
PWM_PERIOD3 **x=0, 1**	PWMx_BA+0x3C	R/W	PWM Period Register 3	0x0000_0000
PWM_PERIOD4 **x=0, 1**	PWMx_BA+0x40	R/W	PWM Period Register 4	0x0000_0000
PWM_PERIOD5 **x=0, 1**	PWMx_BA+0x44	R/W	PWM Period Register 5	0x0000_0000
PWM_CMPDAT0	PWMx_BA+0x50	R/W	PWM Comparator Register 0	0x0000_0000

x=0, 1				
PWM_CMPDAT1 x=0, 1	PWMx_BA+0x54	R/W	PWM Comparator Register 1	0x0000_0000
PWM_CMPDAT2 x=0, 1	PWMx_BA+0x58	R/W	PWM Comparator Register 2	0x0000_0000
PWM_CMPDAT3 x=0, 1	PWMx_BA+0x5C	R/W	PWM Comparator Register 3	0x0000_0000
PWM_CMPDAT4 x=0, 1	PWMx_BA+0x60	R/W	PWM Comparator Register 4	0x0000_0000
PWM_CMPDAT5 x=0, 1	PWMx_BA+0x64	R/W	PWM Comparator Register 5	0x0000_0000
PWM_DTCTL0_1 x=0, 1	PWMx_BA+0x70	R/W	PWM Dead-Time Control Register 0	0x0000_0000
PWM_DTCTL2_3 x=0, 1	PWMx_BA+0x74	R/W	PWM Dead-Time Control Register 2	0x0000_0000
PWM_DTCTL4_5 x=0, 1	PWMx_BA+0x78	R/W	PWM Dead-Time Control Register 4	0x0000_0000
PWM_PHS0_1 x=0, 1	PWMx_BA+0x80	R/W	PWM Counter Phase Register 0	0x0000_0000
PWM_PHS2_3 x=0, 1	PWMx_BA+0x84	R/W	PWM Counter Phase Register 2	0x0000_0000
PWM_PHS4_5 x=0, 1	PWMx_BA+0x88	R/W	PWM Counter Phase Register 4	0x0000_0000
PWM_CNT0 x=0, 1	PWMx_BA+0x90	R	PWM Counter Register 0	0x0000_0000
PWM_CNT1 x=0, 1	PWMx_BA+0x94	R	PWM Counter Register 1	0x0000_0000
PWM_CNT2 x=0, 1	PWMx_BA+0x98	R	PWM Counter Register 2	0x0000_0000
PWM_CNT3 x=0, 1	PWMx_BA+0x9C	R	PWM Counter Register 3	0x0000_0000
PWM_CNT4 x=0, 1	PWMx_BA+0xA0	R	PWM Counter Register 4	0x0000_0000
PWM_CNT5 x=0, 1	PWMx_BA+0xA4	R	PWM Counter Register 5	0x0000_0000
PWM_WGCTL0 x=0, 1	PWMx_BA+0xB0	R/W	PWM Generation Register 0	0x0000_0000
PWM_WGCTL1 x=0, 1	PWMx_BA+0xB4	R/W	PWM Generation Register 1	0x0000_0000

PWM_MSKEN **x=0, 1**	PWMx_BA+0xB8	R/W	PWM Mask Enable Register	0x0000_0000
PWM_MSK **x=0, 1**	PWMx_BA+0xBC	R/W	PWM Mask Data Register	0x0000_0000
PWM_BNF **x=0, 1**	PWMx_BA+0xC0	R/W	PWM Brake Noise Filter Register	0x0000_0000
PWM_FAILBRK **x=0, 1**	PWMx_BA+0xC4	R/W	PWM System Fail Brake Control Register	0x0000_0000
PWM_BRKCTL0_1 **x=0, 1**	PWMx_BA+0xC8	R/W	PWM Brake Edge Detect Control Register 0	0x0000_0000
PWM_BRKCTL2_3 **x=0, 1**	PWMx_BA+0xCC	R/W	PWM Brake Edge Detect Control Register 2	0x0000_0000
PWM_BRKCTL4_5 **x=0, 1**	PWMx_BA+0xD0	R/W	PWM Brake Edge Detect Control Register 4	0x0000_0000
PWM_POLCTL **x=0, 1**	PWMx_BA+0xD4	R/W	PWM Pin Polar Inverse Register	0x0000_0000
PWM_POEN **x=0, 1**	PWMx_BA+0xD8	R/W	PWM Output Enable Register	0x0000_0000
PWM_SWBRK **x=0, 1**	PWMx_BA+0xDC	W	PWM Software Brake Control Register	0x0000_0000
PWM_INTEN0 **x=0, 1**	PWMx_BA+0xE0	R/W	PWM Interrupt Enable Register 0	0x0000_0000
PWM_INTEN1 **x=0, 1**	PWMx_BA+0xE4	R/W	PWM Interrupt Enable Register 1	0x0000_0000
PWM_INTSTS0 **x=0, 1**	PWMx_BA+0xE8	R/W	PWM Interrupt Flag Register 0	0x0000_0000
PWM_INTSTS1 **x=0, 1**	PWMx_BA+0xEC	R/W	PWM Interrupt Flag Register 1	0x0000_0000
PWM_IFA **x=0, 1**	PWMx_BA+0xF0	R/W	PWM Interrupt Flag Accumulator Register	0x0000_0000
PWM_DACTRGEN **x=0, 1**	PWMx_BA+0xF4	R/W	PWM Trigger DAC Enable Register	0x0000_0000
PWM_EADCTS0 **x=0, 1**	PWMx_BA+0xF8	R/W	PWM Trigger EADC Source Select Register 0	0x0000_0000
PWM_EADCTS1 **x=0, 1**	PWMx_BA+0xFC	R/W	PWM Trigger EADC Source Select Register 1	0x0000_0000
PWM_FTCMPDAT0_1 **x=0, 1**	PWMx_BA+0x100	R/W	PWM Free Trigger Compare Register 0	0x0000_0000

PWM_FTCMPDAT2_3 x=0, 1	PWMx_BA+0x104	R/W	PWM Free Trigger Compare Register 2	0x0000_0000
PWM_FTCMPDAT4_5 x=0, 1	PWMx_BA+0x108	R/W	PWM Free Trigger Compare Register 4	0x0000_0000
PWM_SSCTL x=0, 1	PWMx_BA+0x110	R/W	PWM Synchronous Start Control Register	0x0000_0000
PWM_SSTRG x=0, 1	PWMx_BA+0x114	W	PWM Synchronous Start Trigger Register	0x0000_0000
PWM_STATUS x=0, 1	PWMx_BA+0x120	R/W	PWM Status Register	0x0000_0000
PWM_CAPINEN x=0, 1	PWMx_BA+0x200	R/W	PWM Capture Input Enable Register	0x0000_0000
PWM_CAPCTL x=0, 1	PWMx_BA+0x204	R/W	PWM Capture Control Register	0x0000_0000
PWM_CAPSTS x=0, 1	PWMx_BA+0x208	R	PWM Capture Status Register	0x0000_0000
PWM_RCAPDAT0 x=0, 1	PWMx_BA+0x20C	R	PWM Rising Capture Data Register 0	0x0000_0000
PWM_FCAPDAT0 x=0, 1	PWMx_BA+0x210	R	PWM Falling Capture Data Register 0	0x0000_0000
PWM_RCAPDAT1 x=0, 1	PWMx_BA+0x214	R	PWM Rising Capture Data Register 1	0x0000_0000
PWM_FCAPDAT1 x=0, 1	PWMx_BA+0x218	R	PWM Falling Capture Data Register 1	0x0000_0000
PWM_RCAPDAT2 x=0, 1	PWMx_BA+0x21C	R	PWM Rising Capture Data Register 2	0x0000_0000
PWM_FCAPDAT2 x=0, 1	PWMx_BA+0x220	R	PWM Falling Capture Data Register 2	0x0000_0000
PWM_RCAPDAT3 x=0, 1	PWMx_BA+0x224	R	PWM Rising Capture Data Register 3	0x0000_0000
PWM_FCAPDAT3 x=0, 1	PWMx_BA+0x228	R	PWM Falling Capture Data Register 3	0x0000_0000
PWM_RCAPDAT4 x=0, 1	PWMx_BA+0x22C	R	PWM Rising Capture Data Register 4	0x0000_0000

PWM_FCAPDAT4 x=0, 1	PWMx_BA+0x230	R	PWM Falling Capture Data Register 4	0x0000_0000
PWM_RCAPDAT5 x=0, 1	PWMx_BA+0x234	R	PWM Rising Capture Data Register 5	0x0000_0000
PWM_FCAPDAT5 x=0, 1	PWMx_BA+0x238	R	PWM Falling Capture Data Register 5	0x0000_0000
PWM_PDMACTL x=0, 1	PWMx_BA+0x23C	R/W	PWM PDMA Control Register	0x0000_0000
PWM_PDMACAP0_1 x=0, 1	PWMx_BA+0x240	R	PWM Capture Channel 01 PDMA Register	0x0000_0000
PWM_PDMACAP2_3 x=0, 1	PWMx_BA+0x244	R	PWM Capture Channel 23 PDMA Register	0x0000_0000
PWM_PDMACAP4_5 x=0, 1	PWMx_BA+0x248	R	PWM Capture Channel 45 PDMA Register	0x0000_0000
PWM_CAPIEN x=0, 1	PWMx_BA+0x250	R/W	PWM Capture Interrupt Enable Register	0x0000_0000
PWM_CAPIF x=0, 1	PWMx_BA+0x254	R/W	PWM Capture Interrupt Flag Register	0x0000_0000
PWM_PBUF0 x=0, 1	PWMx_BA+0x304	R	PWM PERIOD0 Buffer	0x0000_0000
PWM_PBUF1 x=0, 1	PWMx_BA+0x308	R	PWM PERIOD1 Buffer	0x0000_0000
PWM_PBUF2 x=0, 1	PWMx_BA+0x30C	R	PWM PERIOD2 Buffer	0x0000_0000
PWM_PBUF3 x=0, 1	PWMx_BA+0x310	R	PWM PERIOD3 Buffer	0x0000_0000
PWM_PBUF4 x=0, 1	PWMx_BA+0x314	R	PWM PERIOD4 Buffer	0x0000_0000
PWM_PBUF5 x=0, 1	PWMx_BA+0x318	R	PWM PERIOD5 Buffer	0x0000_0000
PWM_CMPBUF0 x=0, 1	PWMx_BA+0x31C	R	PWM CMPDAT0 Buffer	0x0000_0000
PWM_CMPBUF1 x=0, 1	PWMx_BA+0x320	R	PWM CMPDAT1 Buffer	0x0000_0000
PWM_CMPBUF2 x=0, 1	PWMx_BA+0x324	R	PWM CMPDAT2 Buffer	0x0000_0000
PWM_CMPBUF3 x=0, 1	PWMx_BA+0x328	R	PWM CMPDAT3 Buffer	0x0000_0000

PWM_CMPBUF4 **x=0, 1**	PWMx_BA+0x32C	R	PWM CMPDAT4 Buffer	0x0000_0000
PWM_CMPBUF5 **x=0, 1**	PWMx_BA+0x330	R	PWM CMPDAT5 Buffer	0x0000_0000
PWM_FTCBUF0_1 **x=0, 1**	PWMx_BA+0x340	R	PWM FTCMPDAT0_1 Buffer	0x0000_0000
PWM_FTCBUF2_3 **x=0, 1**	PWMx_BA+0x344	R	PWM FTCMPDAT2_3 Buffer	0x0000_0000
PWM_FTCBUF4_5 **x=0, 1**	PWMx_BA+0x348	R	PWM FTCMPDAT4_5 Buffer	0x0000_0000
PWM_FTCI **x=0, 1**	PWMx_BA+0x34C	R/W	PWM FTCMPDAT Indicator Register	0x0000_0000

6.9.7 Register Description

PWM Control Register 0 (PWM_CTL0)

Register	Offset	R/W	Description	Reset Value
PWM_CTL0	PWMx_BA+0x00	R/W	PWM Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
DBGTRIOFF	**DBGHALT**	**Reserved**					**GROUPEN**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved		**IMMLDEN5**	**IMMLDEN4**	**IMMLDEN3**	**IMMLDEN2**	**IMMLDEN1**	**IMMLDEN0**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**WINLDEN5**	**WINLDEN4**	**WINLDEN3**	**WINLDEN2**	**WINLDEN1**	**WINLDEN0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CTRLD5**	**CTRLD4**	**CTRLD3**	**CTRLD2**	**CTRLD1**	**CTRLD0**

Bits	Description	
[31]	**DBGTRIOFF**	**ICE Debug Mode Acknowledge Disable (Write Protect)** 0 = ICE debug mode acknowledgement effects PWM output. PWM pin will be forced as tri-state while ICE debug mode acknowledged. 1 = ICE debug mode acknowledgement disabled. PWM pin will keep output no matter ICE debug mode acknowledged or not. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[30]	**DBGHALT**	**ICE Debug Mode Counter Halt (Write Protect)** If counter halt is enabled, PWM all counters will keep current value until exit ICE debug mode. 0 = ICE debug mode counter halt Disabled. 1 = ICE debug mode counter halt Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[29:26]	**Reserved**	Reserved.
[24]	**GROUPEN**	**Group Function Enable Bit** 0 = The output waveform of each PWM channel are independent. 1 = Unify the PWM_CH2 and PWM_CH4 to output the same waveform as PWM_CH0 and unify the PWM_CH3 and PWM_CH5 to output the same waveform as PWM_CH1.
[23:22]	**Reserved**	Reserved.
[21:16]	**IMMLDENn**	**Immediately Load Enable Bits** Each bit n controls the corresponding PWM channel n. 0 = PERIOD will load to PBUF at the end point of each period. CMPDAT will load to CMPBUF at the end point or center point of each period by setting CTRLD bit. 1 = PERIOD/CMPDAT will load to PBUF and CMPBUF immediately when software update PERIOD/CMPDAT. **Note:** If IMMLDENn Enabled, WINLDENn and CTRLDn will be invalid.

[15:14]	**Reserved**	Reserved.
[13:8]	**WINLDENn**	**Window Load Enable Bit** Each bit n controls the corresponding PWM channel n. 0 = PERIOD will load to PBUF at the end point of each period. CMPDAT will load to CMPBUF at the end point or center point of each period by setting CTRLD bit. 1 = PERIOD will load to PBUF at the end point of each period. CMPDAT will load to CMPBUF at the end point of each period when valid reload window is set. The valid reload window is set by software write 1 to PWM_LOAD register and cleared by hardware after load success.
[7:6]	**Reserved**	Reserved.
[5:0]	**CTRLDn**	**Center Re-load** Each bit n controls the corresponding PWM channel n. In up-down counter type, PERIOD will load to PBUF at the end point of each period. CMPDAT will load to CMPBUF at the center point of a period.

PWM Control Register 1 (PWM_CTL1)

Register	Offset	R/W	Description	Reset Value
PWM_CTL1	PWMx_BA+0x04	R/W	PWM Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved					OUTMODE4	OUTMODE2	OUTMODE0
23	22	21	20	19	18	17	16
Reserved		CNTMODE5	CNTMODE4	CNTMODE3	CNTMODE2	CNTMODE1	CNTMODE0
15	14	13	12	11	10	9	8
Reserved				CNTTYPE5		CNTTYPE4	
7	6	5	4	3	2	1	0
CNTTYPE3		CNTTYPE2		CNTTYPE1		CNTTYPE0	

Bits	Description	
[31:27]	**Reserved**	Reserved.
[26:24]	**OUTMODEn**	**PWM Output Mode** Each bit n controls the output mode of corresponding PWM channel n. 0 = PWM independent mode. 1 = PWM complementary mode. **Note:** When operating in group function, these bits must all set to the same mode.
[23:22]	**Reserved**	Reserved.
[21:16]	**CNTMODEn**	**PWM Counter Mode** Each bit n controls the corresponding PWM channel n. 0 = Auto-reload mode. 1 = One-shot mode.
[15:12]	**Reserved**	Reserved.
[11:0]	**CNTTYPEn**	**PWM Counter Behavior Type** Each bit n controls corresponding PWM channel n. 00 = Up counter type (supports in capture mode). 01 = Down count type (supports in capture mode). 10 = Up-down counter type. 11 = Reserved.

PWM Synchronization Register (PWM_SYNC)

Register	Offset	R/W	Description	Reset Value
PWM_SYNC	PWMx_BA+0x08	R/W	PWM Synchronization Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved					PHSDIR4	PHSDIR2	PHSDIR0
23	22	21	20	19	18	17	16
SINPINV	SFLTCNT			SFLTCSEL			SNFLTEN
15	14	13	12	11	10	9	8
Reserved		SINSRC4		SINSRC2		SINSRC0	
7	6	5	4	3	2	1	0
Reserved					PHSEN4	PHSEN2	PHSEN0

Bits	Description	
[31:27]	**Reserved**	Reserved.
[26:24]	**PHSDIRn**	**PWM Phase Direction Control** Each bit n controls corresponding PWM channel n. 0 = Control PWM counter count decrement after synchronizing. 1 = Control PWM counter count increment after synchronizing.
[23]	**SINPINV**	**SYNC Input Pin Inverse** 0 = The state of pin SYNC is passed to the positive edge detector. 1 = The inversed state of pin SYNC is passed to the positive edge detector.
[22:20]	**SFLTCNT**	**SYNC Edge Detector Filter Count** The register bits control the counter number of edge detector.
[19:17]	**SFLTCSEL**	**SYNC Edge Detector Filter Clock Selection** 000 = Filter clock = HCLK. 001 = Filter clock = HCLK/2. 010 = Filter clock = HCLK/4. 011 = Filter clock = HCLK/8. 100 = Filter clock = HCLK/16. 101 = Filter clock = HCLK/32. 110 = Filter clock = HCLK/64. 111 = Filter clock = HCLK/128.
[16]	**SNFLTEN**	**PWM0_SYNC_IN Noise Filter Enable Bit** 0 = Noise filter of input pin PWM0_SYNC_IN Disabled. 1 = Noise filter of input pin PWM0_SYNC_IN Enabled.
[15:14]	**Reserved**	Reserved.
[13:8]	**SINSRCn**	**PWM0_SYNC_IN Source Selection** Each bit n controls corresponding PWM channel n. 00 = Synchronize source from SYNC_IN or SWSYNC.

		01 = Counter equal to 0. 10 = Counter equal to PWM_CMPDATm, m denotes 1, 3, 5. 11 = SYNC_OUT will not be generated.
[7:3]	**Reserved**	Reserved.
[2:0]	**PHSENn**	**SYNC Phase Enable Bit** Each bit n controls corresponding PWM channel n. 0 = PWM counter load PHS value Disabled. 1 = PWM counter load PHS value Enabled.

PWM Software Control Synchronization Register (PWM_SWSYNC)

Register	Offset	R/W	Description	Reset Value
PWM_SWSYNC	PWMx_BA+0x0C	R/W	PWM Software Control Synchronization Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					SWSYNC4	SWSYNC2	SWSYNC0

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2:0]	**SWSYNCn**	**Software SYNC Function** Each bit n controls corresponding PWM channel n. When SINSRCn (PWM_SYNC[13:8]) is selected to 0, SYNC_OUT source is come from SYNC_IN or this bit.

PWM Clock Source Register (PWM_CLKSRC)

Register	Offset	R/W	Description	Reset Value
PWM_CLKSRC	PWMx_BA+0x10	R/W	PWM Clock Source Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					ECLKSRC4		
15	14	13	12	11	10	9	8
Reserved					ECLKSRC2		
7	6	5	4	3	2	1	0
Reserved					ECLKSRC0		

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:16]	**ECLKSRC4**	**PWM_CH45 External Clock Source Select** 000 = PWMx_CLK, x denotes 0 or 1. 001 = TIMER0 overflow. 010 = TIMER1 overflow. 011 = TIMER2 overflow. 100 = TIMER3 overflow. Others = Reserved.
[15:11]	**Reserved**	Reserved.
[10:8]	**ECLKSRC2**	**PWM_CH23 External Clock Source Select** 000 = PWMx_CLK, x denotes 0 or 1. 001 = TIMER0 overflow. 010 = TIMER1 overflow. 011 = TIMER2 overflow. 100 = TIMER3 overflow. Others = Reserved.
[7:3]	**Reserved**	Reserved.
[2:0]	**ECLKSRC0**	**PWM_CH01 External Clock Source Select** 000 = PWMx_CLK, x denotes 0 or 1. 001 = TIMER0 overflow. 010 = TIMER1 overflow. 011 = TIMER2 overflow. 100 = TIMER3 overflow. Others = Reserved.

PWM Clock Pre-Scale Register 0_1, 2_3, 4_5 (PWM_CLKPSC0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
PWM_CLKPSC0_1	PWMx_BA+0x14	R/W	PWM Clock Pre-scale Register 0	0x0000_0000
PWM_CLKPSC2_3	PWMx_BA+0x18	R/W	PWM Clock Pre-scale Register 2	0x0000_0000
PWM_CLKPSC4_5	PWMx_BA+0x1C	R/W	PWM Clock Pre-scale Register 4	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**CLKPSC**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CLKPSC							

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**CLKPSC**	**PWM Counter Clock Pre-scale** The clock of PWM counter is decided by clock prescaler. Each PWM pair share one PWM counter clock prescaler. The clock of PWM counter is divided by (CLKPSC+ 1).

PWM Counter Enable Register (PWM_CNTEN)

Register	Offset	R/W	Description	Reset Value
PWM_CNTEN	PWMx_BA+0x20	R/W	PWM Counter Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CNTEN5**	**CNTEN4**	**CNTEN3**	**CNTEN2**	**CNTEN1**	**CNTEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**CNTENn**	**PWM Counter Enable Bits** Each bit n controls the corresponding PWM channel n. 0 = PWM Counter and clock prescaler Stop Running. 1 = PWM Counter and clock prescaler Start Running.

PWM Clear Counter Register (PWM_CNTCLR)

Register	Offset	R/W	Description	Reset Value
PWM_CNTCLR	PWMx_BA+0x24	R/W	PWM Clear Counter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		CNTCLR5	CNTCLR4	CNTCLR3	CNTCLR2	CNTCLR1	CNTCLR0

Bits	Description	
[31:6]	Reserved	Reserved.
[5:0]	CNTCLRn	**Clear PWM Counter Control Bit** It is automatically cleared by hardware. Each bit n controls the corresponding PWM channel n. 0 = No effect. 1 = Clear 16-bit PWM counter to 0000H.

PWM Load Register (PWM_LOAD)

Register	Offset	R/W	Description	Reset Value
PWM_LOAD	PWMx_BA+0x28	R/W	PWM Load Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**LOAD5**	**LOAD4**	**LOAD3**	**LOAD2**	**LOAD1**	**LOAD0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**LOADn**	**Re-load PWM Comparator Register (CMPDAT) Control Bit** This bit is software write, hardware clear when current PWM period end. Each bit n controls the corresponding PWM channel n. Write Operation: 0 = No effect. 1 = Set load window of window loading mode. Read Operation: 0 = No load window is set. 1 = Load window is set. **Note:** This bit only use in window loading mode, WINLDENn(PWM_CTL0[13:8]) = 1.

PWM Period Register 0~5 (PWM_PERIOD0~5)

Register	Offset	R/W	Description	Reset Value
PWM_PERIOD0	PWMx_BA+0x30	R/W	PWM Period Register 0	0x0000_0000
PWM_PERIOD1	PWMx_BA+0x34	R/W	PWM Period Register 1	0x0000_0000
PWM_PERIOD2	PWMx_BA+0x38	R/W	PWM Period Register 2	0x0000_0000
PWM_PERIOD3	PWMx_BA+0x3C	R/W	PWM Period Register 3	0x0000_0000
PWM_PERIOD4	PWMx_BA+0x40	R/W	PWM Period Register 4	0x0000_0000
PWM_PERIOD5	PWMx_BA+0x44	R/W	PWM Period Register 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PERIOD							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PERIOD							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PERIOD**	**PWM Period Register** Up-Count mode: In this mode, PWM counter counts from 0 to PERIOD, and restarts from 0. Down-Count mode: In this mode, PWM counter counts from PERIOD to 0, and restarts from PERIOD. PWM period time = (PERIOD+1) * PWM_CLK period. Up-Down-Count mode: In this mode, PWM counter counts from 0 to PERIOD, then decrements to 0 and repeats again. PWM period time = 2 * PERIOD * PWM_CLK period.

PWM Comparator Register 0~5 (PWM_CMPDAT0~5)

Register	Offset	R/W	Description	Reset Value
PWM_CMPDAT0	PWMx_BA+0x50	R/W	PWM Comparator Register 0	0x0000_0000
PWM_CMPDAT1	PWMx_BA+0x54	R/W	PWM Comparator Register 1	0x0000_0000
PWM_CMPDAT2	PWMx_BA+0x58	R/W	PWM Comparator Register 2	0x0000_0000
PWM_CMPDAT3	PWMx_BA+0x5C	R/W	PWM Comparator Register 3	0x0000_0000
PWM_CMPDAT4	PWMx_BA+0x60	R/W	PWM Comparator Register 4	0x0000_0000
PWM_CMPDAT5	PWMx_BA+0x64	R/W	PWM Comparator Register 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CMP							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMP							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CMP**	**PWM Comparator Register** CMP use to compare with CNTR to generate PWM waveform, interrupt and trigger EADC/DAC. In independent mode, CMPDAT0~5 denote as 6 independent PWM_CH0~5 compared point. In complementary mode, CMPDAT0, 2, 4 denote as first compared point, and CMPDAT1, 3, 5 denote as second compared point for the corresponding 3 complementary pairs PWM_CH0 and PWM_CH1, PWM_CH2 and PWM_CH3, PWM_CH4 and PWM_CH5.

PWM Dead-Time Control Register 0_1, 2_3, 4_5 (PWM_DTCTL0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
PWM_DTCTL 0_1	PWMx_BA+0x70	R/W	PWM Dead-Time Control Register 0	0x0000_0000
PWM_DTCTL 2_3	PWMx_BA+0x74	R/W	PWM Dead-Time Control Register 2	0x0000_0000
PWM_DTCTL 4_5	PWMx_BA+0x78	R/W	PWM Dead-Time Control Register 4	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							DTEN
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				DTCNT			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DTCNT							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:17]	**Reserved**	Reserved.
[16]	**DTEN**	**Dead-time Insertion for PWM Pair (PWM_CH0, PWM_CH1) (PWM_CH2, PWM_CH3) (PWM_CH4, PWM_CH5) Enable Bit (Write Protect)** Dead-time insertion is only active when this pair of complementary PWM is enabled. If dead- time insertion is inactive, the outputs of pin pair are complementary without any delay. 0 = Dead-time insertion on the pin pair Disabled. 1 = Dead-time insertion on the pin pair Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[15:12]	**Reserved**	Reserved.
[11:0]	**DTCNT**	**Dead-time Counter (Write Protect)** The dead-time can be calculated from the following formula: Dead-time = (DTCNT[11:0]+1) * PWM_CLK period. **Note:** This register is write protected. Refer to SYS_REGLCTL register.

PWM Counter Phase Register 0_1, 2_3, 4_5 (PWM_PHS0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
PWM_PHS0_1	PWMx_BA+0x80	R/W	PWM Counter Phase Register 0	0x0000_0000
PWM_PHS2_3	PWMx_BA+0x84	R/W	PWM Counter Phase Register 2	0x0000_0000
PWM_PHS4_5	PWMx_BA+0x88	R/W	PWM Counter Phase Register 4	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PHS							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PHS							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PHS**	**PWM Synchronous Start Phase Bits** PHS determines the PWM synchronous start phase value. These bits only use in synchronous function.

PWM Counter Register 0~5 (PWM_CNT0~5)

Register	Offset	R/W	Description	Reset Value
PWM_CNT0	PWMx_BA+0x90	R	PWM Counter Register 0	0x0000_0000
PWM_CNT1	PWMx_BA+0x94	R	PWM Counter Register 1	0x0000_0000
PWM_CNT2	PWMx_BA+0x98	R	PWM Counter Register 2	0x0000_0000
PWM_CNT3	PWMx_BA+0x9C	R	PWM Counter Register 3	0x0000_0000
PWM_CNT4	PWMx_BA+0xA0	R	PWM Counter Register 4	0x0000_0000
PWM_CNT5	PWMx_BA+0xA4	R	PWM Counter Register 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							**DIRF**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CNT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNT							

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**DIRF**	**PWM Direction Indicator Flag (Read Only)** 0 = Counter is Down count. 1 = Counter is UP count.
[15:0]	**CNT**	**PWM Data Register (Read Only)** User can monitor CNTR to know the current value in 16-bit period counter.

PWM Generation Register 0 (PWM_WGCTL0)

Register	Offset	R/W	Description	Reset Value
PWM_WGCTL0	PWMx_BA+0xB0	R/W	PWM Generation Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				PRDPCTL5		PRDPCTL4	
23	22	21	20	19	18	17	16
PRDPCTL3		PRDPCTL2		PRDPCTL1		PRDPCTL0	
15	14	13	12	11	10	9	8
Reserved				ZPCTL5		ZPCTL4	
7	6	5	4	3	2	1	0
ZPCTL3		ZPCTL2	ZPCTL1			ZPCTL0	

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:16]	**PRDPCTLn**	**PWM Period (Center) Point Control** Each bit n controls the corresponding PWM channel n. 00 = Do nothing. 01 = PWM period (center) point output Low. 10 = PWM period (center) point output High. 11 = PWM period (center) point output Toggle. PWM can control output level when PWM counter count to (PERIODn+1). **Note:** This bit is center point control when PWM counter operating in up-down counter type.
[15:12]	**Reserved**	Reserved.
[11:0]	**ZPCTLn**	**PWM Zero Point Control** Each bit n controls the corresponding PWM channel n. 00 = Do nothing. 01 = PWM zero point output Low. 10 = PWM zero point output High. 11 = PWM zero point output Toggle. PWM can control output level when PWM counter count to zero.

PWM Generation Register 1 (PWM_WGCTL1)

Register	Offset	R/W	Description	Reset Value
PWM_WGCTL1	PWMx_BA+0xB4	R/W	PWM Generation Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				CMPDCTL5		CMPDCTL4	
23	22	21	20	19	18	17	16
CMPDCTL3		CMPDCTL2		CMPDCTL1		CMPDCTL0	
15	14	13	12	11	10	9	8
Reserved				CMPUCTL5		CMPUCTL4	
7	6	5	4	3	2	1	0
CMPUCTL3		CMPUCTL2		CMPUCTL1		CMPUCTL0	

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:16]	**CMPDCTLn**	**PWM Compare Down Point Control** Each bit n controls the corresponding PWM channel n. 00 = Do nothing. 01 = PWM compare down point output Low. 10 = PWM compare down point output High. 11 = PWM compare down point output Toggle. PWM can control output level when PWM counter down count to CMPDAT. **Note:** In complementary mode, CMPDCTL1, 3, 5 use as another CMPDCTL for channel 0, 2, 4.
[15:12]	**Reserved**	Reserved.
[11:0]	**CMPUCTLn**	**PWM Compare Up Point Control** Each bit n controls the corresponding PWM channel n. 00 = Do nothing. 01 = PWM compare up point output Low. 10 = PWM compare up point output High. 11 = PWM compare up point output Toggle. PWM can control output level when PWM counter up count to CMPDAT. **Note:** In complementary mode, CMPUCTL1, 3, 5 use as another CMPUCTL for channel 0, 2, 4.

PWM Mask Enable Register (PWM_MSKEN)

Register	Offset	R/W	Description	Reset Value
PWM_MSKEN	PWMx_BA+0xB8	R/W	PWM Mask Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**MSKEN5**	**MSKEN4**	**MSKEN3**	**MSKEN2**	**MSKEN1**	**MSKEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**MSKENn**	**PWM Mask Enable Bits** Each bit n controls the corresponding PWM channel n. The PWM output signal will be masked when this bit is enabled. The corresponding PWM channel n will output MSKDATn (PWM_MSK[5:0]) data. 0 = PWM output signal is non-masked. 1 = PWM output signal is masked and output MSKDATn data.

PWM Mask DATA Register (PWM_MSK)

Register	Offset	R/W	Description	Reset Value
PWM_MSK	PWMx_BA+0xBC	R/W	PWM Mask Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**MSKDAT5**	**MSKDAT4**	**MSKDAT3**	**MSKDAT2**	**MSKDAT1**	**MSKDAT0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**MSKDATn**	**PWM Mask Data Bit** This data bit control the state of PWMn output pin, if corresponding mask function is enabled. Each bit n controls the corresponding PWM channel n. 0 = Output logic low to PWMn. 1 = Output logic high to PWMn.

PWM Brake Noise Filter Register (PWM_BNF)

Register	Offset	R/W	Description	Reset Value
PWM_BNF	PWMx_BA+0xC0	R/W	PWM Brake Noise Filter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
BRK1PINV	BRK1FCNT			BRK1NFSEL			BRK1NFEN
7	6	5	4	3	2	1	0
BRK0PINV	BRK0FCNT			BRK0NFSEL			BRK0NFEN

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**BRK1PINV**	**Brake 1 Pin Inverse** 0 = The state of pin PWMx_BRAKE1 is passed to the negative edge detector. 1 = The inversed state of pin PWMx_BRAKE1 is passed to the negative edge detector.
[14:12]	**BRK1FCNT**	**Brake 1 Edge Detector Filter Count** The register bits control the Brake1 filter counter to count from 0 to BRK1FCNT.
[11:9]	**BRK1NFSEL**	**Brake 1 Edge Detector Filter Clock Selection** 000 = Filter clock = HCLK. 001 = Filter clock = HCLK/2. 010 = Filter clock = HCLK/4. 011 = Filter clock = HCLK/8. 100 = Filter clock = HCLK/16. 101 = Filter clock = HCLK/32. 110 = Filter clock = HCLK/64. 111 = Filter clock = HCLK/128.
[8]	**BRK1NFEN**	**PWM Brake 1 Noise Filter Enable Bit** 0 = Noise filter of PWM Brake 1 Disabled. 1 = Noise filter of PWM Brake 1 Enabled.
[7]	**BRK0PINV**	**Brake 0 Pin Inverse** 0 = The state of pin PWMx_BRAKE0 is passed to the negative edge detector. 1 = The inversed state of pin PWMx_BRAKE10 is passed to the negative edge detector.
[6:4]	**BRK0FCNT**	**Brake 0 Edge Detector Filter Count** The register bits control the Brake0 filter counter to count from 0 to BRK1FCNT.
[3:1]	**BRK0NFSEL**	**Brake 0 Edge Detector Filter Clock Selection** 000 = Filter clock = HCLK.

		001 = Filter clock = HCLK/2. 010 = Filter clock = HCLK/4. 011 = Filter clock = HCLK/8. 100 = Filter clock = HCLK/16. 101 = Filter clock = HCLK/32. 110 = Filter clock = HCLK/64. 111 = Filter clock = HCLK/128.
[0]	**BRK0NFEN**	**PWM Brake 0 Noise Filter Enable Bit** 0 = Noise filter of PWM Brake 0 Disabled. 1 = Noise filter of PWM Brake 0 Enabled.

PWM System Fail Brake Control Register (PWM_FAILBRK)

Register	Offset	R/W	Description	Reset Value
PWM_FAILBRK	PWMx_BA+0xC4	R/W	PWM System Fail Brake Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				CORBRKEN	RAMBRKEN	BODBRKEN	CSSBRKEN

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**CORBRKEN**	**Core Lockup Detection Trigger PWM Brake Function 0 Enable Bit** 0 = Brake Function triggered by Core lockup detection Disabled. 1 = Brake Function triggered by Core lockup detection Enabled.
[2]	**RAMBRKEN**	**SRAM Parity Error Detection Trigger PWM Brake Function 0 Enable Bit** 0 = Brake Function triggered by SRAM parity error detection Disabled. 1 = Brake Function triggered by SRAM parity error detection Enabled.
[1]	**BODBRKEN**	**Brown-out Detection Trigger PWM Brake Function 0 Enable Bit** 0 = Brake Function triggered by BOD Disabled. 1 = Brake Function triggered by BOD Enabled.
[0]	**CSSBRKEN**	**Clock Security System Detection Trigger PWM Brake Function 0 Enable Bit** 0 = Brake Function triggered by CSS detection Disabled. 1 = Brake Function triggered by CSS detection Enabled.

PWM Brake Edge Detect Control Register 0_1, 2_3, 4_5 (PWM_BRKCTL0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
PWM_BRKCTL0_1	PWMx_BA+0xC8	R/W	PWM Brake Edge Detect Control Register 0	0x0000_0000
PWM_BRKCTL2_3	PWMx_BA+0xCC	R/W	PWM Brake Edge Detect Control Register 2	0x0000_0000
PWM_BRKCTL4_5	PWMx_BA+0xD0	R/W	PWM Brake Edge Detect Control Register 4	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved				BRKAODD		BRKAEVEN	
15	14	13	12	11	10	9	8
SYSLBEN	Reserved	BRKP1LEN	BRKP0LEN	Reserved		CPO1LBEN	CPO0LBEN
7	6	5	4	3	2	1	0
SYSEBEN	Reserved	BRKP1EEN	BRKP0EEN	Reserved		CPO1EBEN	CPO0EBEN

Bits	Description	
[31:20]	**Reserved**	Reserved.
[19:18]	**BRKAODD**	**PWM Brake Action Select for Odd Channel (Write Protect)** 00 = PWM odd channel level-detect brake function not affect channel output. 01 = PWM odd channel output tri-state when level-detect brake happened. 10 = PWM odd channel output low level when level-detect brake happened. 11 = PWM odd channel output high level when level-detect brake happened. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[17:16]	**BRKAEVEN**	**PWM Brake Action Select for Even Channel (Write Protect)** 00 = PWM even channel level-detect brake function not affect channel output. 01 = PWM even channel output tri-state when level-detect brake happened. 10 = PWM even channel output low level when level-detect brake happened. 11 = PWM even channel output high level when level-detect brake happened. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[15]	**SYSLBEN**	**System Fail As Level-detect Brake Source Enable Bit (Write Protect)** 0 = System Fail condition as level-detect brake source Disabled. 1 = System Fail condition as level-detect brake source Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[14]	**Reserved**	Reserved.
[13]	**BRKP1LEN**	**BKP1 Pin As Level-detect Brake Source Enable Bit (Write Protect)** 0 = PWMx_BRAKE1 pin as level-detect brake source Disabled. 1 = PWMx_BRAKE1 pin as level-detect brake source Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.

[12]	**BRKP0LEN**	**BKP0 Pin As Level-detect Brake Source Enable Bit (Write Protect)** 0 = PWMx_BRAKE0 pin as level-detect brake source Disabled. 1 = PWMx_BRAKE0 pin as level-detect brake source Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[11:10]	**Reserved**	Reserved.
[9]	**CPO1LBEN**	**ACMP1_O Digital Output As Level-detect Brake Source Enable Bit (Write Protect)** 0 = ACMP1_O as level-detect brake source Disabled. 1 = ACMP1_O as level-detect brake source Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[8]	**CPO0LBEN**	**ACMP0_O Digital Output As Level-detect Brake Source Enable Bit (Write Protect)** 0 = ACMP0_O as level-detect brake source Disabled. 1 = ACMP0_O as level-detect brake source Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[7]	**SYSEBEN**	**System Fail As Edge-detect Brake Source Enable Bit (Write Protect)** 0 = System Fail condition as edge-detect brake source Disabled. 1 = System Fail condition as edge-detect brake source Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[6]	**Reserved**	Reserved.
[5]	**BRKP1EEN**	**PWMx_BRAKE1 Pin As Edge-detect Brake Source Enable Bit (Write Protect)** 0 = BKP1 pin as edge-detect brake source Disabled. 1 = BKP1 pin as edge-detect brake source Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[4]	**BRKP0EEN**	**PWMx_BRAKE0 Pin As Edge-detect Brake Source Enable Bit (Write Protect)** 0 = BKP0 pin as edge-detect brake source Disabled. 1 = BKP0 pin as edge-detect brake source Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[3:2]	**Reserved**	Reserved.
[1]	**CPO1EBEN**	**ACMP1_O Digital Output As Edge-detect Brake Source Enable Bit (Write Protect)** 0 = ACMP1_O as edge-detect brake source Disabled. 1 = ACMP1_O as edge-detect brake source Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[0]	**CPO0EBEN**	**ACMP0_O Digital Output As Edge-detect Brake Source Enable Bit (Write Protect)** 0 = ACMP0_O as edge-detect brake source Disabled. 1 = ACMP0_O as edge-detect brake source Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.

PWM Pin Polar Inverse Control (PWM_POLCTL)

Register	Offset	R/W	Description	Reset Value
PWM_POLCTL	PWMx_BA+0xD4	R/W	PWM Pin Polar Inverse Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		PINV5	PINV4	PINV3	PINV2	PINV1	PINV0

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**PINVn**	**PWM PIN Polar Inverse Control Bits** The register controls polarity state of PWM output. Each bit n controls the corresponding PWM channel n. 0 = PWM output polar inverse Disabled. 1 = PWM output polar inverse Enabled.

PWM Output Enable Register (PWM_POEN)

Register	Offset	R/W	Description	Reset Value
PWM_POEN	PWMx_BA+0xD8	R/W	PWM Output Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**POEN5**	**POEN4**	**POEN3**	**POEN2**	**POEN1**	**POEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**POENn**	**PWM Pin Output Enable Bits** Each bit n controls the corresponding PWM channel n. 0 = PWM pin at tri-state. 1 = PWM pin in output mode.

PWM Software Brake Control Register (PWM_SWBRK)

Register	Offset	R/W	Description	Reset Value
PWM_SWBRK	PWMx_BA+0xDC	W	PWM Software Brake Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				BRKLTRG4		BRKLTRG2	BRKLTRG0
7	6	5	4	3	2	1	0
Reserved							

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:8]	**BRKLTRGn**	**PWM Level Brake Software Trigger (Write Only) (Write Protect)** Each bit n controls the corresponding PWM pair n. Write 1 to this bit will trigger level brake, and set BRKLIFn to 1 in PWM_INTSTS1 register. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[7:0]	**Reserved**	Reserved.

PWM Interrupt Enable Register 0 (PWM_INTEN0)

Register	Offset	R/W	Description	Reset Value
PWM_INTEN0	PWMx_BA+0xE0	R/W	PWM Interrupt Enable Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CMPDIEN5	CMPDIEN4	CMPDIEN3	CMPDIEN2	CMPDIEN1	CMPDIEN0
23	22	21	20	19	18	17	16
IFAIEN4_5	Reserved	CMPUIEN5	CMPUIEN4	CMPUIEN3	CMPUIEN2	CMPUIEN1	CMPUIEN0
15	14	13	12	11	10	9	8
IFAIEN2_3	Reserved	PIEN5	PIEN4	PIEN3	PIEN2	PIEN1	PIEN0
7	6	5	4	3	2	1	0
IFAIEN0_1	Reserved	ZIEN5	ZIEN4	ZIEN3	ZIEN2	ZIEN1	ZIEN0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:24]	**CMPDIENn**	**PWM Compare Down Count Interrupt Enable Bits** Each bit n controls the corresponding PWM channel n. 0 = Compare down count interrupt Disabled. 1 = Compare down count interrupt Enabled. **Note:** In complementary mode, CMPDIEN1, 3, 5 use as another CMPDIEN for channel 0, 2, 4.
[23]	**IFAIEN4_5**	**PWM_CH4/5 Interrupt Flag Accumulator Interrupt Enable Bit** 0 = Interrupt Flag accumulator interrupt Disabled. 1 = Interrupt Flag accumulator interrupt Enabled.
[22]	**Reserved**	Reserved.
[21:16]	**CMPUIENn**	**PWM Compare Up Count Interrupt Enable Bits** Each bit n controls the corresponding PWM channel n. 0 = Compare up count interrupt Disabled. 1 = Compare up count interrupt Enabled. **Note:** In complementary mode, CMPUIEN1, 3, 5 use as another CMPUIEN for channel 0, 2, 4.
[15]	**IFAIEN2_3**	**PWM_CH2/3 Interrupt Flag Accumulator Interrupt Enable Bit** 0 = Interrupt Flag accumulator interrupt Disabled. 1 = Interrupt Flag accumulator interrupt Enabled.
[14]	**Reserved**	Reserved.
[13:8]	**PIENn**	**PWM Period Point Interrupt Enable Bits** Each bit n controls the corresponding PWM channel n. 0 = Period point interrupt Disabled. 1 = Period point interrupt Enabled. **Note1:** When up-down counter type period point means center point. **Note2:** Odd channels will read always 0 at complementary mode.

[7]	**IFAIEN0_1**	**PWM_CH0/1 Interrupt Flag Accumulator Interrupt Enable Bit** 0 = Interrupt Flag accumulator interrupt Disabled. 1 = Interrupt Flag accumulator interrupt Enabled.
[6]	**Reserved**	Reserved.
[5:0]	**ZIENn**	**PWM Zero Point Interrupt Enable Bits** Each bit n controls the corresponding PWM channel n. 0 = Zero point interrupt Disabled. 1 = Zero point interrupt Enabled. **Note:** Odd channels will read always 0 at complementary mode.

PWM Interrupt Enable Register 1 (PWM_INTEN1)

Register	Offset	R/W	Description	Reset Value
PWM_INTEN1	PWMx_BA+0xE4	R/W	PWM Interrupt Enable Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					BRKLIEN4_5	BRKLIEN2_3	BRKLIEN0_1
7	6	5	4	3	2	1	0
Reserved					BRKEIEN4_5	BRKEIEN2_3	BRKEIEN0_1

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10]	**BRKLIEN4_5**	**PWM Level-detect Brake Interrupt Enable Bit for Channel4/5 (Write Protect)** 0 = Level-detect Brake interrupt for channel4/5 Disabled. 1 = Level-detect Brake interrupt for channel4/5 Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[9]	**BRKLIEN2_3**	**PWM Level-detect Brake Interrupt Enable Bit for Channel2/3 (Write Protect)** 0 = Level-detect Brake interrupt for channel2/3 Disabled. 1 = Level-detect Brake interrupt for channel2/3 Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[8]	**BRKLIEN0_1**	**PWM Level-detect Brake Interrupt Enable Bit for Channel0/1 (Write Protect)** 0 = Level-detect Brake interrupt for channel0/1 Disabled. 1 = Level-detect Brake interrupt for channel0/1 Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[7:3]	**Reserved**	Reserved.
[2]	**BRKEIEN4_5**	**PWM Edge-detect Brake Interrupt Enable Bit for Channel4/5 (Write Protect)** 0 = Edge-detect Brake interrupt for channel4/5 Disabled. 1 = Edge-detect Brake interrupt for channel4/5 Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[1]	**BRKEIEN2_3**	**PWM Edge-detect Brake Interrupt Enable Bit for Channel2/3 (Write Protect)** 0 = Edge-detect Brake interrupt for channel2/3 Disabled. 1 = Edge-detect Brake interrupt for channel2/3 Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[0]	**BRKEIEN0_1**	**PWM Edge-detect Brake Interrupt Enable Bit for Channel0/1 (Write Protect)** 0 = Edge-detect Brake interrupt for channel0/1 Disabled. 1 = Edge-detect Brake interrupt for channel0/1 Enabled. **Note:** This register is write protected. Refer to SYS_REGLCTL register.

PWM Interrupt Flag Register 0 (PWM_INTSTS0)

Register	Offset	R/W	Description	Reset Value
PWM_INTSTS0	PWMx_BA+0xE8	R/W	PWM Interrupt Flag Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CMPDIF5	CMPDIF4	CMPDIF3	CMPDIF2	CMPDIF1	CMPDIF0
23	22	21	20	19	18	17	16
IFAIF4_5	Reserved	CMPUIF5	CMPUIF4	CMPUIF3	CMPUIF2	CMPUIF1	CMPUIF0
15	14	13	12	11	10	9	8
IFAIF2_3	Reserved	PIF5	PIF4	PIF3	PIF2	PIF1	PIF0
7	6	5	4	3	2	1	0
IFAIF0_1	Reserved	ZIF5	ZIF4	ZIF3	ZIF2	ZIF1	ZIF0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:24]	**CMPDIFn**	**PWM Compare Down Count Interrupt Flag** Each bit n controls the corresponding PWM channel n. Flag is set by hardware when PWM counter down count and reaches PWM_CMPDATn, software can clear this bit by writing 1 to it. **Note1:** If CMPDAT equal to PERIOD, this flag is not working in down counter type selection. **Note2:** In complementary mode, CMPDIF1, 3, 5 use as another CMPDIF for channel 0, 2, 4.
[23]	**IFAIF4_5**	**PWM_CH4/5 Interrupt Flag Accumulator Interrupt Flag** Flag is set by hardware when condition match IFSEL4_5 in PWM_IFA register, software can clear this bit by writing 1 to it.
[22]	**Reserved**	Reserved.
[21:16]	**CMPUIFn**	**PWM Compare Up Count Interrupt Flag** Flag is set by hardware when PWM counter up count and reaches PWM_CMPDATn, software can clear this bit by writing 1 to it. Each bit n controls the corresponding PWM channel n. **Note1:** If CMPDAT equal to PERIOD, this flag is not working in up counter type selection. **Note2:** In complementary mode, CMPUIF1, 3, 5 use as another CMPUIF for channel 0, 2, 4.
[15]	**IFAIF2_3**	**PWM_CH2/3 Interrupt Flag Accumulator Interrupt Flag** Flag is set by hardware when condition match IFSEL2_3 in PWM_IFA register, software can clear this bit by writing 1 to it.
[14]	**Reserved**	Reserved.
[13:8]	**PIFn**	**PWM Period Point Interrupt Flag** This bit is set by hardware when PWM counter reaches PWM_PERIODn, software can write 1 to clear this bit to zero. Each bit n controls the corresponding PWM channel n.
[7]	**IFAIF0_1**	**PWM_CH0/1 Interrupt Flag Accumulator Interrupt Flag**

		Flag is set by hardware when condition match IFSEL0_1 in PWM_IFA register, software can clear this bit by writing 1 to it.
[6]	**Reserved**	Reserved.
[5:0]	**ZIFn**	**PWM Zero Point Interrupt Flag** Each bit n controls the corresponding PWM channel n. This bit is set by hardware when PWM counter reaches zero, software can write 1 to clear this bit to zero.

PWM Interrupt Flag Register 1 (PWM_INTSTS1)

Register	Offset	R/W	Description	Reset Value
PWM_INTSTS1	PWMx_BA+0xEC	R/W	PWM Interrupt Flag Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		BRKLSTS5	BRKLSTS4	BRKLSTS3	BRKLSTS2	BRKLSTS1	BRKLSTS0
23	22	21	20	19	18	17	16
Reserved		BRKESTS5	BRKESTS4	BRKESTS3	BRKESTS2	BRKESTS1	BRKESTS0
15	14	13	12	11	10	9	8
Reserved		BRKLIF5	BRKLIF4	BRKLIF3	BRKLIF2	BRKLIF1	BRKLIF0
7	6	5	4	3	2	1	0
Reserved		BRKEIF5	BRKEIF4	BRKEIF3	BRKEIF2	BRKEIF1	BRKEIF0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**BRKLSTS5**	**PWM Channel5 Level-detect Brake Status (Read Only)** 0 = PWM channel5 level-detect brake state is released. 1 = When PWM channel5 level-detect brake detects a falling edge of any enabled brake source; this flag will be set to indicate the PWM channel5 at brake state. **Note:** This bit is read only and auto cleared by hardware. When enabled brake source return to high level, PWM will release brake state until current PWM period finished. The PWM waveform will start output from next full PWM period.
[28]	**BRKLSTS4**	**PWM Channel4 Level-detect Brake Status (Read Only)** 0 = PWM channel4 level-detect brake state is released. 1 = When PWM channel4 level-detect brake detects a falling edge of any enabled brake source; this flag will be set to indicate the PWM channel4 at brake state. **Note:** This bit is read only and auto cleared by hardware. When enabled brake source return to high level, PWM will release brake state until current PWM period finished. The PWM waveform will start output from next full PWM period.
[27]	**BRKLSTS3**	**PWM Channel3 Level-detect Brake Status (Read Only)** 0 = PWM channel3 level-detect brake state is released. 1 = When PWM channel3 level-detect brake detects a falling edge of any enabled brake source; this flag will be set to indicate the PWM channel3 at brake state. **Note:** This bit is read only and auto cleared by hardware. When enabled brake source return to high level, PWM will release brake state until current PWM period finished. The PWM waveform will start output from next full PWM period.
[26]	**BRKLSTS2**	**PWM Channel2 Level-detect Brake Status (Read Only)** 0 = PWM channel2 level-detect brake state is released. 1 = When PWM channel2 level-detect brake detects a falling edge of any enabled brake source; this flag will be set to indicate the PWM channel2 at brake state. **Note:** This bit is read only and auto cleared by hardware. When enabled brake source return to high level, PWM will release brake state until current PWM period finished. The PWM waveform will start output from next full PWM period.
[25]	**BRKLSTS1**	**PWM Channel1 Level-detect Brake Status (Read Only)**

		0 = PWM channel1 level-detect brake state is released. 1 = When PWM channel1 level-detect brake detects a falling edge of any enabled brake source; this flag will be set to indicate the PWM channel1 at brake state. **Note:** This bit is read only and auto cleared by hardware. When enabled brake source return to high level, PWM will release brake state until current PWM period finished. The PWM waveform will start output from next full PWM period.
[24]	**BRKLSTS0**	**PWM Channel0 Level-detect Brake Status (Read Only)** 0 = PWM channel0 level-detect brake state is released. 1 = When PWM channel0 level-detect brake detects a falling edge of any enabled brake source; this flag will be set to indicate the PWM channel0 at brake state. **Note:** This bit is read only and auto cleared by hardware. When enabled brake source return to high level, PWM will release brake state until current PWM period finished. The PWM waveform will start output from next full PWM period.
[23:22]	**Reserved**	Reserved.
[21]	**BRKESTS5**	**PWM Channel5 Edge-detect Brake Status** 0 = PWM channel5 edge-detect brake state is released. 1 = When PWM channel5 edge-detect brake detects a falling edge of any enabled brake source; this flag will be set to indicate the PWM channel5 at brake state, writing 1 to clear.
[20]	**BRKESTS4**	**PWM Channel4 Edge-detect Brake Status** 0 = PWM channel4 edge-detect brake state is released. 1 = When PWM channel4 edge-detect brake detects a falling edge of any enabled brake source; this flag will be set to indicate the PWM channel4 at brake state, writing 1 to clear.
[19]	**BRKESTS3**	**PWM Channel3 Edge-detect Brake Status** 0 = PWM channel3 edge-detect brake state is released. 1 = When PWM channel3 edge-detect brake detects a falling edge of any enabled brake source; this flag will be set to indicate the PWM channel3 at brake state, writing 1 to clear.
[18]	**BRKESTS2**	**PWM Channel2 Edge-detect Brake Status** 0 = PWM channel2 edge-detect brake state is released. 1 = When PWM channel2 edge-detect brake detects a falling edge of any enabled brake source; this flag will be set to indicate the PWM channel2 at brake state, writing 1 to clear.
[17]	**BRKESTS1**	**PWM Channel1 Edge-detect Brake Status** 0 = PWM channel1 edge-detect brake state is released. 1 = When PWM channel1 edge-detect brake detects a falling edge of any enabled brake source; this flag will be set to indicate the PWM channel1 at brake state, writing 1 to clear.
[16]	**BRKESTS0**	**PWM Channel0 Edge-detect Brake Status** 0 = PWM channel0 edge-detect brake state is released. 1 = When PWM channel0 edge-detect brake detects a falling edge of any enabled brake source; this flag will be set to indicate the PWM channel0 at brake state, writing 1 to clear.
[15:14]	**Reserved**	Reserved.
[13]	**BRKLIF5**	**PWM Channel5 Level-detect Brake Interrupt Flag (Write Protect)** 0 = PWM channel5 level-detect brake event do not happened. 1 = When PWM channel5 level-detect brake event happened, this bit is set to 1, writing 1 to clear. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[12]	**BRKLIF4**	**PWM Channel4 Level-detect Brake Interrupt Flag (Write Protect)** 0 = PWM channel4 level-detect brake event do not happened. 1 = When PWM channel4 level-detect brake event happened, this bit is set to 1, writing 1 to clear. **Note:** This register is write protected. Refer to SYS_REGLCTL register.

[11]	**BRKLIF3**	**PWM Channel3 Level-detect Brake Interrupt Flag (Write Protect)** 0 = PWM channel3 level-detect brake event do not happened. 1 = When PWM channel3 level-detect brake event happened, this bit is set to 1, writing 1 to clear. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[10]	**BRKLIF2**	**PWM Channel2 Level-detect Brake Interrupt Flag (Write Protect)** 0 = PWM channel2 level-detect brake event do not happened. 1 = When PWM channel2 level-detect brake event happened, this bit is set to 1, writing 1 to clear. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[9]	**BRKLIF1**	**PWM Channel1 Level-detect Brake Interrupt Flag (Write Protect)** 0 = PWM channel1 level-detect brake event do not happened. 1 = When PWM channel1 level-detect brake event happened, this bit is set to 1, writing 1 to clear. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[8]	**BRKLIF0**	**PWM Channel0 Level-detect Brake Interrupt Flag (Write Protect)** 0 = PWM channel0 level-detect brake event do not happened. 1 = When PWM channel0 level-detect brake event happened, this bit is set to 1, writing 1 to clear. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[7:6]	**Reserved**	Reserved.
[5]	**BRKEIF5**	**PWM Channel5 Edge-detect Brake Interrupt Flag (Write Protect)** 0 = PWM channel5 edge-detect brake event do not happened. 1 = When PWM channel5 edge-detect brake event happened, this bit is set to 1, writing 1 to clear. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[4]	**BRKEIF4**	**PWM Channel4 Edge-detect Brake Interrupt Flag (Write Protect)** 0 = PWM channel4 edge-detect brake event do not happened. 1 = When PWM channel4 edge-detect brake event happened, this bit is set to 1, writing 1 to clear. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[3]	**BRKEIF3**	**PWM Channel3 Edge-detect Brake Interrupt Flag (Write Protect)** 0 = PWM channel3 edge-detect brake event do not happened. 1 = When PWM channel3 edge-detect brake event happened, this bit is set to 1, writing 1 to clear. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[2]	**BRKEIF2**	**PWM Channel2 Edge-detect Brake Interrupt Flag (Write Protect)** 0 = PWM channel2 edge-detect brake event do not happened. 1 = When PWM channel2 edge-detect brake event happened, this bit is set to 1, writing 1 to clear. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[1]	**BRKEIF1**	**PWM Channel1 Edge-detect Brake Interrupt Flag (Write Protect)** 0 = PWM channel1 edge-detect brake event do not happened. 1 = When PWM channel1 edge-detect brake event happened, this bit is set to 1, writing 1 to clear. **Note:** This register is write protected. Refer to SYS_REGLCTL register.
[0]	**BRKEIF0**	**PWM Channel0 Edge-detect Brake Interrupt Flag (Write Protect)** 0 = PWM channel0 edge-detect brake event do not happened.

		1 = When PWM channel0 edge-detect brake event happened, this bit is set to 1, writing 1 to clear. **Note:** This register is write protected. Refer to SYS_REGLCTL register.

PWM Interrupt Flag Accumulator Register (PWM_IFA)

Register	Offset	R/W	Description	Reset Value
PWM_IFA	PWMx_BA+0xF0	R/W	PWM Interrupt Flag Accumulator Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
IFAEN4_5	IFSEL4_5			IFCNT4_5			
15	14	13	12	11	10	9	8
IFAEN2_3	IFSEL2_3			IFCNT2_3			
7	6	5	4	3	2	1	0
IFAEN0_1	IFSEL0_1			IFCNT0_1			

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23]	**IFAEN4_5**	**PWM_CH4 and PWM_CH5 Interrupt Flag Accumulator Enable Bit** 0 = PWM_CH4 and PWM_CH5 interrupt flag accumulator Disabled. 1 = PWM_CH4 and PWM_CH5 interrupt flag accumulator Enabled.
[22:20]	**IFSEL4_5**	**PWM_CH4 and PWM_CH5 Interrupt Flag Accumulator Source Select** 000 = CNT equal to Zero in channel 4. 001 = CNT equal to PERIOD in channel 4. 010 = CNT equal to CMPU in channel 4. 011 = CNT equal to CMPD in channel 4. 100 = CNT equal to Zero in channel 5. 101 = CNT equal to PERIOD in channel 5. 110 = CNT equal to CMPU in channel 5. 111 = CNT equal to CMPD in channel 5.
[19:16]	**IFCNT4_5**	**PWM_CH4 and PWM_CH5 Interrupt Flag Counter** The register sets the count number which defines how many times of PWM_CH4 and PWM_CH5 period occurs to set bit IFAIF4_5 to request the PWM period interrupt. IFAIF4_5 (PWM_INTSTS0[23]) will be set in every IFCNT4_5+1 times of PWM period.
[15]	**IFAEN2_3**	**PWM_CH2 and PWM_CH3 Interrupt Flag Accumulator Enable Bit** 0 = PWM_CH2 and PWM_CH3 interrupt flag accumulator Disabled. 1 = PWM_CH2 and PWM_CH3 interrupt flag accumulator Enabled.
[14:12]	**IFSEL2_3**	**PWM_CH2 and PWM_CH3 Interrupt Flag Accumulator Source Select** 000 = CNT equal to Zero in channel 2. 001 = CNT equal to PERIOD in channel 2. 010 = CNT equal to CMPU in channel 2. 011 = CNT equal to CMPD in channel 2. 100 = CNT equal to Zero in channel 3. 101 = CNT equal to PERIOD in channel 3.

		110 = CNT equal to CMPU in channel 3. 111 = CNT equal to CMPD in channel 3.
[11:8]	**IFCNT2_3**	**PWM_CH2 and PWM_CH3 Interrupt Flag Counter** The register sets the count number which defines how many times of PWM_CH2 and PWM_CH3 period occurs to set bit IFAIF2_3 to request the PWM period interrupt. IFAIF2_3 (PWM_INTSTS0[15]) will be set in every IFCNT2_3+1 times of PWM period.
[7]	**IFAEN0_1**	**PWM_CH0 and PWM_CH1 Interrupt Flag Accumulator Enable Bit** 0 = PWM_CH0 and PWM_CH1 interrupt flag accumulator Disabled. 1 = PWM_CH0 and PWM_CH1 interrupt flag accumulator Enabled.
[6:4]	**IFSEL0_1**	**PWM_CH0 and PWM_CH1 Interrupt Flag Accumulator Source Select** 000 = CNT equal to Zero in channel 0. 001 = CNT equal to PERIOD in channel 0. 010 = CNT equal to CMPU in channel 0. 011 = CNT equal to CMPD in channel 0. 100 = CNT equal to Zero in channel 1. 101 = CNT equal to PERIOD in channel 1. 110 = CNT equal to CMPU in channel 1. 111 = CNT equal to CMPD in channel 1.
[3:0]	**IFCNT0_1**	**PWM_CH0 and PWM_CH1 Interrupt Flag Counter** The register sets the count number which defines how many times of PWM_CH0 and PWM_CH1 period occurs to set bit IFAIF0_1 to request the PWM period interrupt. IFAIF0_1 (PWM_INTSTS0[7]) will be set in every IFCNT0_1+1 times of PWM period.

PWM Trigger DAC Enable Register (PWM_DACTRGEN)

Register	Offset	R/W	Description	Reset Value
PWM_DACTRGEN	PWMx_BA+0xF4	R/W	PWM Trigger DAC Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CDTRGEN5	CDTRGEN4	CDTRGEN3	CDTRGEN2	CDTRGEN1	CDTRGEN0
23	22	21	20	19	18	17	16
Reserved		CUTRGEN5	CUTRGEN4	CUTRGEN3	CUTRGEN2	CUTRGEN1	CUTRGEN0
15	14	13	12	11	10	9	8
Reserved		PTE5	PTE4	PTE3	PTE2	PTE1	PTE0
7	6	5	4	3	2	1	0
Reserved		ZTE5	ZTE4	ZTE3	ZTE2	ZTE1	ZTE0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:24]	**CDTRGEn**	**PWM Compare Down Count Point Trigger DAC Enable Bit** 0 = PWM Compare Down count point trigger DAC function Disabled. 1 = PWM Compare Down count point trigger DAC function Enabled. PWM can trigger DAC to start action when PWM counter down count to CMPDAT if this bit is set to1. Each bit n controls the corresponding PWM channel n. **Note1:** This bit should keep at 0 when PWM counter operating in up counter type. **Note2:** In complementary mode, CDTRGE1, 3, 5 use as another CDTRGE for channel 0, 2, 4.
[23:22]	**Reserved**	Reserved.
[21:16]	**CUTRGEn**	**PWM Compare Up Count Point Trigger DAC Enable Bit** 0 = PWM Compare Up point trigger DAC function Disabled. 1 = PWM Compare Up point trigger DAC function Enabled. PWM can trigger DAC to start action when PWM counter up count to CMPDAT if this bit is set to1. Each bit n controls the corresponding PWM channel n. **Note1:** This bit should keep at 0 when PWM counter operating in down counter type. **Note2:** In complementary mode, CUTRGE1, 3, 5 use as another CUTRGE for channel 0, 2, 4.
[15:14]	**Reserved**	Reserved.
[13:8]	**PTEn**	**PWM Period Point Trigger DAC Enable Bit** 0 = PWM period point trigger DAC function Disabled. 1 = PWM period point trigger DAC function Enabled. PWM can trigger DAC to start action when PWM counter up count to (PERIODn+1) if this bit is set to1. Each bit n controls the corresponding PWM channel n.
[7:6]	**Reserved**	Reserved.
[5:0]	**ZTEn**	**PWM Zero Point Trigger DAC Enable Bit** 0 = PWM period point trigger DAC function Disabled.

		1 = PWM period point trigger DAC function Enabled. PWM can trigger EADC/DAC/DMA to start action when PWM counter down count to zero if this bit is set to1. Each bit n controls the corresponding PWM channel n.

PWM Trigger EADC Source Select Register 0 (PWM_EADCTS0)

Register	Offset	R/W	Description	Reset Value
PWM_EADCTS0	PWMx_BA+0xF8	R/W	PWM Trigger EADC Source Select Register 0	0x0000_0000

31	30	29	28	27	26	25	24
TRGEN3	Reserved			TRGSEL3			
23	22	21	20	19	18	17	16
TRGEN2	Reserved			TRGSEL2			
15	14	13	12	11	10	9	8
TRGEN1	Reserved			TRGSEL1			
7	6	5	4	3	2	1	0
TRGEN0	Reserved			TRGSEL0			

Bits	Description	
[31]	**TRGEN3**	**PWM_CH3 Trigger EADC Enable Bit** 0 = PWM_CH3 Trigger EADC Disabled. 1 = PWM_CH3 Trigger EADC Enabled.
[30:28]	**Reserved**	Reserved.
[27:24]	**TRGSEL3**	**PWM_CH3 Trigger EADC Source Select** 0000 = PWM_CH2 zero point. 0001 = PWM_CH2 period point. 0010 = PWM_CH2 zero or period point. 0011 = PWM_CH2 up-count CMPDAT point. 0100 = PWM_CH2 down-count CMPDAT point. 0101 = PWM_CH3 zero point. 0110 = PWM_CH3 period point. 0111 = PWM_CH3 zero or period point. 1000 = PWM_CH3 up-count CMPDAT point. 1001 = PWM_CH3 down-count CMPDAT point. 1010 = PWM_CH0 up-count free CMPDAT point. 1011 = PWM_CH0 down-count free CMPDAT point. 1100 = PWM_CH2 up-count free CMPDAT point. 1101 = PWM_CH2 down-count free CMPDAT point. 1110 = PWM_CH4 up-count free CMPDAT point. 1111 = PWM_CH4 down-count free CMPDAT point.
[23]	**TRGEN2**	**PWM_CH2 Trigger EADC Enable Bit** 0 = PWM_CH2 Trigger EADC Disabled. 1 = PWM_CH2 Trigger EADC Enabled.
[22:20]	**Reserved**	Reserved.
[19:16]	**TRGSEL2**	**PWM_CH2 Trigger EADC Source Select**

		0000 = PWM_CH2 zero point. 0001 = PWM_CH2 period point. 0010 = PWM_CH2 zero or period point. 0011 = PWM_CH2 up-count CMPDAT point. 0100 = PWM_CH2 down-count CMPDAT point. 0101 = PWM_CH3 zero point. 0110 = PWM_CH3 period point. 0111 = PWM_CH3 zero or period point. 1000 = PWM_CH3 up-count CMPDAT point. 1001 = PWM_CH3 down-count CMPDAT point. 1010 = PWM_CH0 up-count free CMPDAT point. 1011 = PWM_CH0 down-count free CMPDAT point. 1100 = PWM_CH2 up-count free CMPDAT point. 1101 = PWM_CH2 down-count free CMPDAT point. 1110 = PWM_CH4 up-count free CMPDAT point. 1111 = PWM_CH4 down-count free CMPDAT point.
[15]	**TRGEN1**	**PWM_CH1 Trigger EADC Enable Bit** 0 = PWM_CH1 Trigger EADC Disabled. 1 = PWM_CH1 Trigger EADC Enabled.
[14:12]	**Reserved**	Reserved.
[11:8]	**TRGSEL1**	**PWM_CH1 Trigger EADC Source Select** 0000 = PWM_CH0 zero point. 0001 = PWM_CH0 period point. 0010 = PWM_CH0 zero or period point. 0011 = PWM_CH0 up-count CMPDAT point. 0100 = PWM_CH0 down-count CMPDAT point. 0101 = PWM_CH1 zero point. 0110 = PWM_CH1 period point. 0111 = PWM_CH1 zero or period point. 1000 = PWM_CH1 up-count CMPDAT point. 1001 = PWM_CH1 down-count CMPDAT point. 1010 = PWM_CH0 up-count free CMPDAT point. 1011 = PWM_CH0 down-count free CMPDAT point. 1100 = PWM_CH2 up-count free CMPDAT point. 1101 = PWM_CH2 down-count free CMPDAT point. 1110 = PWM_CH4 up-count free CMPDAT point. 1111 = PWM_CH4 down-count free CMPDAT point.
[7]	**TRGEN0**	**PWM_CH0 Trigger EADC Enable Bit** 0 = PWM_CH0 Trigger EADC Disabled. 1 = PWM_CH0 Trigger EADC Enabled.
[6:4]	**Reserved**	Reserved.
[3:0]	**TRGSEL0**	**PWM_CH0 Trigger EADC Source Select** 0000 = PWM_CH0 zero point. 0001 = PWM_CH0 period point. 0010 = PWM_CH0 zero or period point. 0011 = PWM_CH0 up-count CMPDAT point. 0100 = PWM_CH0 down-count CMPDAT point.

		0101 = PWM_CH1 zero point. 0110 = PWM_CH1 period point. 0111 = PWM_CH1 zero or period point. 1000 = PWM_CH1 up-count CMPDAT point. 1001 = PWM_CH1 down-count CMPDAT point. 1010 = PWM_CH0 up-count free CMPDAT point. 1011 = PWM_CH0 down-count free CMPDAT point. 1100 = PWM_CH2 up-count free CMPDAT point. 1101 = PWM_CH2 down-count free CMPDAT point. 1110 = PWM_CH4 up-count free CMPDAT point. 1111 = PWM_CH4 down-count free CMPDAT point.

PWM Trigger EADC Source Select Register 1 (PWM_EADCTS1)

Register	Offset	R/W	Description	Reset Value
PWM_EADCTS1	PWMx_BA+0xFC	R/W	PWM Trigger EADC Source Select Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
TRGEN5	Reserved			TRGSEL5			
7	6	5	4	3	2	1	0
TRGEN4	Reserved			TRGSEL4			

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**TRGEN5**	**PWM_CH5 Trigger EADC Enable Bit** 0 = PWM_CH5 Trigger EADC Disabled. 1 = PWM_CH5 Trigger EADC Enabled.
[14:12]	**Reserved**	Reserved.
[11:8]	**TRGSEL5**	**PWM_CH5 Trigger EADC Source Select** 0000 = PWM_CH4 zero point. 0001 = PWM_CH4 period point. 0010 = PWM_CH4 zero or period point. 0011 = PWM_CH4 up-count CMPDAT point. 0100 = PWM_CH4 down-count CMPDAT point. 0101 = PWM_CH5 zero point. 0110 = PWM_CH5 period point. 0111 = PWM_CH5 zero or period point. 1000 = PWM_CH5 up-count CMPDAT point. 1001 = PWM_CH5 down-count CMPDAT point. 1010 = PWM_CH0 up-count free CMPDAT point. 1011 = PWM_CH0 down-count free CMPDAT point. 1100 = PWM_CH2 up-count free CMPDAT point. 1101 = PWM_CH2 down-count free CMPDAT point. 1110 = PWM_CH4 up-count free CMPDAT point. 1111 = PWM_CH4 down-count free CMPDAT point.
[7]	**TRGEN4**	**PWM_CH4 Trigger EADC Enable Bit** 0 = PWM_CH4 Trigger EADC Disabled. 1 = PWM_CH4 Trigger EADC Enabled.
[6:4]	**Reserved**	Reserved.

[3:0]	**TRGSEL4**	**PWM_CH4 Trigger EADC Source Select** 0000 = PWM_CH4 zero point. 0001 = PWM_CH4 period point. 0010 = PWM_CH4 zero or period point. 0011 = PWM_CH4 up-count CMPDAT point. 0100 = PWM_CH4 down-count CMPDAT point. 0101 = PWM_CH5 zero point. 0110 = PWM_CH5 period point. 0111 = PWM_CH5 zero or period point. 1000 = PWM_CH5 up-count CMPDAT point. 1001 = PWM_CH5 down-count CMPDAT point. 1010 = PWM_CH0 up-count free CMPDAT point. 1011 = PWM_CH0 down-count free CMPDAT point. 1100 = PWM_CH2 up-count free CMPDAT point. 1101 = PWM_CH2 down-count free CMPDAT point. 1110 = PWM_CH4 up-count free CMPDAT point. 1111 = PWM_CH4 down-count free CMPDAT point.

PWM Free Trigger Compare Register 0_1, 2_3, 4_5 (PWM_FTCMPDAT0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
PWM_FTCMPDAT0_1	PWMx_BA+0x100	R/W	PWM Free Trigger Compare Register 0	0x0000_0000
PWM_FTCMPDAT2_3	PWMx_BA+0x104	R/W	PWM Free Trigger Compare Register 2	0x0000_0000
PWM_FTCMPDAT4_5	PWMx_BA+0x108	R/W	PWM Free Trigger Compare Register 4	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FTCMP							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FTCMP							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**FTCMP**	**PWM Free Trigger Compare Register** FTCMP use to compare with even CNTR to trigger EADC. FTCMPDAT0, 2, 4 corresponding complementary pairs PWM_CH0and PWM_CH1, PWM_CH2 and PWM_CH3, PWM_CH4 and PWM_CH5.

PWM Synchronous Start Control Register (PWM_SSCTL)

Register	Offset	R/W	Description	Reset Value
PWM_SSCTL	PWMx_BA+0x110	R/W	PWM Synchronous Start Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**SSEN5**	**SSEN4**	**SSEN3**	**SSEN2**	**SSEN1**	**SSEN0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**SSENn**	**PWM Synchronous Start Function Enable Bits** When synchronous start function is enabled, the PWM counter enable register (PWM_CNTEN) can be enabled by writing PWM synchronous start trigger bit (CNTSEN). Each bit n controls the corresponding PWM channel n. 0 = PWM synchronous start function Disabled. 1 = PWM synchronous start function Enabled.

PWM Synchronous Start Trigger Register (PWM_SSTRG)

Register	Offset	R/W	Description	Reset Value
PWM_SSTRG	PWMx_BA+0x114	W	PWM Synchronous Start Trigger Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**CNTSEN**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**CNTSEN**	**PWM Counter Synchronous Start Enable Bit (Write Only)** PMW counter synchronous enable function is used to make selected PWM channels (include PWM0_CHx and PWM1_CHx) start counting at the same time. Writing this bit to 1 will also set the counter enable bit (CNTENn, n denotes channel 0 to 5) if correlated PWM channel counter synchronous start function is enabled. **Note:** This bit only present in PWM0_BA.

PWM Status Register (PWM_STATUS)

Register	Offset	R/W	Description	Reset Value
PWM_STATUS	PWMx_BA+0x120	R/W	PWM Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							DACTRGF
23	22	21	20	19	18	17	16
Reserved		ADCTRGF5	ADCTRGF4	ADCTRGF3	ADCTRGF2	ADCTRGF1	ADCTRGF0
15	14	13	12	11	10	9	8
Reserved					SYNCINF4	SYNCINF2	SYNCINF0
7	6	5	4	3	2	1	0
Reserved		CNTMAXF5	CNTMAXF4	CNTMAXF3	CNTMAXF2	CNTMAXF1	CNTMAXF0

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**DACTRGF**	**DAC Start of Conversion Flag** 0 = Indicates no DAC start of conversion trigger event has occurred. 1 = Indicates an DAC start of conversion trigger event has occurred, software can write 1 to clear this bit.
[23:22]	**Reserved**	Reserved.
[21:16]	**ADCTRGFn**	**EADC Start of Conversion Flag** Each bit n controls the corresponding PWM channel n. 0 = Indicates no EADC start of conversion trigger event has occurred. 1 = Indicates an EADC start of conversion trigger event has occurred, software can write 1 to clear this bit.
[15:11]	**Reserved**	Reserved.
[10:8]	**SYNCINFn**	**Input Synchronization Latched Flag** Each bit n controls the corresponding PWM channel n. 0 = Indicates no SYNC_IN event has occurred. 1 = Indicates an SYNC_IN event has occurred, software can write 1 to clear this bit.
[7:6]	**Reserved**	Reserved.
[5:0]	**CNTMAXFn**	**Time-base Counter Equal to 0xFFFF Latched Flag** Each bit n controls the corresponding PWM channel n. 0 = indicates the time-base counter never reached its maximum value 0xFFFF. 1 = indicates the time-base counter reached its maximum value, software can write 1 to clear this bit.

PWM Capture Input Enable Register (PWM_CAPINEN)

Register	Offset	R/W	Description	Reset Value
PWM_CAPINEN	PWMx_BA+0x200	R/W	PWM Capture Input Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	CAPINEN5		CAPINEN4	CAPINEN3	CAPINEN2	CAPINEN1	CAPINEN0

Bits	Description	
[31:6]	Reserved	Reserved.
[5:0]	CAPINENn	**Capture Input Enable Bits** Each bit n controls the corresponding PWM channel n. 0 = PWM Channel capture input path Disabled. The input of PWM channel capture function is always regarded as 0. 1 = PWM Channel capture input path Enabled. The input of PWM channel capture function comes from correlative multifunction pin.

PWM Capture Control Register (PWM_CAPCTL)

Register	Offset	R/W	Description	Reset Value
PWM_CAPCTL	PWMx_BA+0x204	R/W	PWM Capture Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		FCRLDEN5	FCRLDEN4	FCRLDEN3	FCRLDEN2	FCRLDEN1	FCRLDEN0
23	22	21	20	19	18	17	16
Reserved		RCRLDEN5	RCRLDEN4	RCRLDEN3	RCRLDEN2	RCRLDEN1	RCRLDEN0
15	14	13	12	11	10	9	8
Reserved		CAPINV5	CAPINV4	CAPINV3	CAPINV2	CAPINV1	CAPINV0
7	6	5	4	3	2	1	0
Reserved	CAPEN5		CAPEN4	CAPEN3	CAPEN2	CAPEN1	CAPEN0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:24]	**FCRLDENn**	**Falling Capture Reload Enable Bits** Each bit n controls the corresponding PWM channel n. 0 = Falling capture reload counter Disabled. 1 = Falling capture reload counter Enabled.
[23:22]	**Reserved**	Reserved.
[21:16]	**RCRLDENn**	**Rising Capture Reload Enable Bits** Each bit n controls the corresponding PWM channel n. 0 = Rising capture reload counter Disabled. 1 = Rising capture reload counter Enabled.
[15:14]	**Reserved**	Reserved.
[13:8]	**CAPINVn**	**Capture Inverter Enable Bits** Each bit n controls the corresponding PWM channel n. 0 = Capture source inverter Disabled. 1 = Capture source inverter Enabled. Reverse the input signal from GPIO.
[7:6]	**Reserved**	Reserved.
[5:0]	**CAPENn**	**Capture Function Enable Bits** Each bit n controls the corresponding PWM channel n. 0 = Capture function Disabled. RCAPDAT/FCAPDAT register will not be updated. 1 = Capture function Enabled. Capture latched the PWM counter value when detected rising or falling edge of input signal and saved to RCAPDAT (Rising latch) and FCAPDAT (Falling latch).

PWM Capture Status Register (PWM_CAPSTS)

Register	Offset	R/W	Description	Reset Value
PWM_CAPSTS	PWMx_BA+0x208	R	PWM Capture Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**CFLIFOV5**	**CFLIFOV4**	**CFLIFOV3**	**CFLIFOV2**	**CFLIFOV1**	**CFLIFOV0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CRLIFOV5**	**CRLIFOV4**	**CRLIFOV3**	**CRLIFOV2**	**CRLIFOV1**	**CRLIFOV0**

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13:8]	**CFLIFOVn**	**Capture Falling Latch Interrupt Flag Overrun Status (Read Only)** This flag indicates if falling latch happened when the corresponding CFLIF is 1. Each bit n controls the corresponding PWM channel n. **Note:** This bit will be cleared automatically when user clear corresponding CFLIF.
[7:6]	**Reserved**	Reserved.
[5:0]	**CRLIFOVn**	**Capture Rising Latch Interrupt Flag Overrun Status (Read Only)** This flag indicates if rising latch happened when the corresponding CRLIF is 1. Each bit n controls the corresponding PWM channel n. **Note:** This bit will be cleared automatically when user clear corresponding CRLIF.

PWM Rising Capture Data Register 0~5 (PWM_RCAPDAT 0~5)

Register	Offset	R/W	Description	Reset Value
PWM_RCAPDAT0	PWMx_BA+0x20C	R	PWM Rising Capture Data Register 0	0x0000_0000
PWM_RCAPDAT1	PWMx_BA+0x214	R	PWM Rising Capture Data Register 1	0x0000_0000
PWM_RCAPDAT2	PWMx_BA+0x21C	R	PWM Rising Capture Data Register 2	0x0000_0000
PWM_RCAPDAT3	PWMx_BA+0x224	R	PWM Rising Capture Data Register 3	0x0000_0000
PWM_RCAPDAT4	PWMx_BA+0x22C	R	PWM Rising Capture Data Register 4	0x0000_0000
PWM_RCAPDAT5	PWMx_BA+0x234	R	PWM Rising Capture Data Register 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
RCAPDAT							
7	6	5	4	3	2	1	0
RCAPDAT							

Bits	Description	
[31:16]	Reserved	Reserved.
[15:0]	RCAPDAT	**PWM Rising Capture Data Register (Read Only)** When rising capture condition happened, the PWM counter value will be saved in this register.

PWM Falling Capture Data Register 0~5 (PWM_FCAPDAT 0~5)

Register	Offset	R/W	Description	Reset Value
PWM_FCAPDAT0	PWMx_BA+0x210	R	PWM Falling Capture Data Register 0	0x0000_0000
PWM_FCAPDAT1	PWMx_BA+0x218	R	PWM Falling Capture Data Register 1	0x0000_0000
PWM_FCAPDAT2	PWMx_BA+0x220	R	PWM Falling Capture Data Register 2	0x0000_0000
PWM_FCAPDAT3	PWMx_BA+0x228	R	PWM Falling Capture Data Register 3	0x0000_0000
PWM_FCAPDAT4	PWMx_BA+0x230	R	PWM Falling Capture Data Register 4	0x0000_0000
PWM_FCAPDAT5	PWMx_BA+0x238	R	PWM Falling Capture Data Register 5	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
FCAPDAT							
7	6	5	4	3	2	1	0
FCAPDAT							

Bits	Description	
[31:16]	Reserved	Reserved.
[15:0]	FCAPDAT	**PWM Falling Capture Data Register (Read Only)** When falling capture condition happened, the PWM counter value will be saved in this register.

PWM PDMA Control Register (PWM_PDMACTL)

Register	Offset	R/W	Description	Reset Value
PWM_PDMACTL	PWMx_BA+0x23C	R/W	PWM PDMA Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			CHSEL4_5	CAPORD4_5	CAPMOD4_5		CHEN4_5
15	14	13	12	11	10	9	8
Reserved			CHSEL2_3	CAPORD2_3	CAPMOD2_3		CHEN2_3
7	6	5	4	3	2	1	0
Reserved			CHSEL0_1	CAPORD0_1	CAPMOD0_1		CHEN0_1

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20]	**CHSEL4_5**	**Select Channel 4/5 to Do PDMA Transfer** 0 = Channel4. 1 = Channel5.
[19]	**CAPORD4_5**	**Capture Channel 4/5 Rising/Falling Order** Set this bit to determine whether the PWM_RCAPDAT4/5 or PWM_FCAPDAT4/5 is the first captured data transferred to memory through PDMA when CAPMOD4_5 =11. 0 = PWM_FCAPDAT4/5 is the first captured data to memory. 1 = PWM_RCAPDAT4/5 is the first captured data to memory.
[18:17]	**CAPMOD4_5**	**Select PWM_RCAPDAT4/5 or PWM_FCAPDAT4/5 to Do PDMA Transfer** 00 = Reserved. 01 = PWM_RCAPDAT4/5. 10 = PWM_FCAPDAT4/5. 11 = Both PWM_RCAPDAT4/5 and PWM_FCAPDAT4/5.
[16]	**CHEN4_5**	**Channel 4/5 PDMA Enable Bit** 0 = Channel 4/5 PDMA function Disabled. 1 = Channel 4/5 PDMA function Enabled for the channel 4/5 captured data and transfer to memory.
[15:13]	**Reserved**	Reserved.
[12]	**CHSEL2_3**	**Select Channel 2/3 to Do PDMA Transfer** 0 = Channel2. 1 = Channel3.
[11]	**CAPORD2_3**	**Capture Channel 2/3 Rising/Falling Order** Set this bit to determine whether the PWM_RCAPDAT2/3 or PWM_FCAPDAT2/3 is the first captured data transferred to memory through PDMA when CAPMOD2_3 =11.

		0 = PWM_FCAPDAT2/3 is the first captured data to memory. 1 = PWM_RCAPDAT2/3 is the first captured data to memory.
[10:9]	**CAPMOD2_3**	**Select PWM_RCAPDAT2/3 or PWM_FCAODAT2/3 to Do PDMA Transfer** 00 = Reserved. 01 = PWM_RCAPDAT2/3. 10 = PWM_FCAPDAT2/3. 11 = Both PWM_RCAPDAT2/3 and PWM_FCAPDAT2/3.
[8]	**CHEN2_3**	**Channel 2/3 PDMA Enable Bit** 0 = Channel 2/3 PDMA function Disabled. 1 = Channel 2/3 PDMA function Enabled for the channel 2/3 captured data and transfer to memory.
[7:5]	**Reserved**	Reserved.
[4]	**CHSEL0_1**	**Select Channel 0/1 to Do PDMA Transfer** 0 = Channel0. 1 = Channel1.
[3]	**CAPORD0_1**	**Capture Channel 0/1 Rising/Falling Order** Set this bit to determine whether the PWM_RCAPDAT0/1 or PWM_FCAPDAT0/1 is the first captured data transferred to memory through PDMA when CAPMOD0_1 =11. 0 = PWM_FCAPDAT0/1 is the first captured data to memory. 1 = PWM_RCAPDAT0/1 is the first captured data to memory.
[2:1]	**CAPMOD0_1**	**Select PWM_RCAPDAT0/1 or PWM_FCAPDAT0/1 to Do PDMA Transfer** 00 = Reserved. 01 = PWM_RCAPDAT0/1. 10 = PWM_FCAPDAT0/1. 11 = Both PWM_RCAPDAT0/1 and PWM_FCAPDAT0/1.
[0]	**CHEN0_1**	**Channel 0/1 PDMA Enable Bit** 0 = Channel 0/1 PDMA function Disabled. 1 = Channel 0/1 PDMA function Enabled for the channel 0/1 captured data and transfer to memory.

PWM Capture Channel 0_1, 2_3, 4_5 PDMA Register (PWM_PDMACAP 0_1, 2_3, 4_5)

Register	Offset	R/W	Description	Reset Value
PWM_PDMACAP0_1	PWMx_BA+0x240	R	PWM Capture Channel 01 PDMA Register	0x0000_0000
PWM_PDMACAP2_3	PWMx_BA+0x244	R	PWM Capture Channel 23 PDMA Register	0x0000_0000
PWM_PDMACAP4_5	PWMx_BA+0x248	R	PWM Capture Channel 45 PDMA Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CAPBUF							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CAPBUF							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CAPBUF**	**PWM Capture PDMA Register (Read Only)** This register is use as a buffer to transfer PWM capture rising or falling data to memory by PDMA.

PWM Capture Interrupt Enable Register (PWM_CAPIEN)

Register	Offset	R/W	Description	Reset Value
PWM_CAPIEN	PWMx_BA+0x250	R/W	PWM Capture Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**CAPFIEN5**	**CAPFIEN4**	**CAPFIEN3**	**CAPFIEN2**	**CAPFIEN1**	**CAPFIEN0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CAPRIEN5**	**CAPRIEN4**	**CAPRIEN3**	**CAPRIEN2**	**CAPRIEN1**	**CAPRIEN0**

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13:8]	**CAPFIENn**	**PWM Capture Falling Latch Interrupt Enable Bit** Each bit n controls the corresponding PWM channel n. 0 = Capture falling edge latch interrupt Disabled. 1 = Capture falling edge latch interrupt Enabled. **Note:** When Capture with PDMA operating, CINTENR corresponding channel CAPFIEN must be disabled.
[7:6]	**Reserved**	Reserved.
[5:0]	**CAPRIENn**	**PWM Capture Rising Latch Interrupt Enable Bit** Each bit n controls the corresponding PWM channel n. 0 = Capture rising edge latch interrupt Disabled. 1 = Capture rising edge latch interrupt Enabled. **Note:** When Capture with PDMA operating, CINTENR corresponding channel CAPRIEN must be disabled.

PWM Capture Interrupt Flag Register (PWM_CAPIF)

Register	Offset	R/W	Description	Reset Value
PWM_CAPIF	PWMx_BA+0x254	R/W	PWM Capture Interrupt Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**CFLIF5**	**CFLIF4**	**CFLIF3**	**CFLIF2**	**CFLIF1**	**CFLIF0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CRLIF5**	**CRLIF4**	**CRLIF3**	**CRLIF2**	**CRLIF1**	**CRLIF0**

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13:8]	**CFLIFn**	**PWM Capture Falling Latch Interrupt Flag** This bit is writing 1 to clear. Each bit n controls the corresponding PWM channel n. 0 = No capture falling latch condition happened. 1 = Capture falling latch condition happened, this flag will be set to high. **Note:** When Capture with PDMA operating, CIFR corresponding channel CFLIF will cleared by hardware after PDMA transfer data.
[7:6]	**Reserved**	Reserved.
[5:0]	**CRLIFn**	**PWM Capture Rising Latch Interrupt Flag** This bit is writing 1 to clear. Each bit n controls the corresponding PWM channel n. 0 = No capture rising latch condition happened. 1 = Capture rising latch condition happened, this flag will be set to high. **Note:** When Capture with PDMA operating, CIFR corresponding channel CRLIF will cleared by hardware after PDMA transfer data.

PWM Period Register Buffer 0~5 (PWM_PBUF0~5)

Register	Offset	R/W	Description	Reset Value
PWM_PBUF0	PWMx_BA+0x304	R	PWM PERIOD0 Buffer	0x0000_0000
PWM_PBUF1	PWMx_BA+0x308	R	PWM PERIOD1 Buffer	0x0000_0000
PWM_PBUF2	PWMx_BA+0x30C	R	PWM PERIOD2 Buffer	0x0000_0000
PWM_PBUF3	PWMx_BA+0x310	R	PWM PERIOD3 Buffer	0x0000_0000
PWM_PBUF4	PWMx_BA+0x314	R	PWM PERIOD4 Buffer	0x0000_0000
PWM_PBUF5	PWMx_BA+0x318	R	PWM PERIOD5 Buffer	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PBUF							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PBUF							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PBUF**	**PWM Period Register Buffer (Read Only)** Used as PERIOD active register.

PWM Comparator Register Buffer 0~5 (PWM_CMPBUF0~5)

Register	Offset	R/W	Description	Reset Value
PWM_CMPBUF0	PWMx_BA+0x31C	R	PWM CMPDAT0 Buffer	0x0000_0000
PWM_CMPBUF1	PWMx_BA+0x320	R	PWM CMPDAT1 Buffer	0x0000_0000
PWM_CMPBUF2	PWMx_BA+0x324	R	PWM CMPDAT2 Buffer	0x0000_0000
PWM_CMPBUF3	PWMx_BA+0x328	R	PWM CMPDAT3 Buffer	0x0000_0000
PWM_CMPBUF4	PWMx_BA+0x32C	R	PWM CMPDAT4 Buffer	0x0000_0000
PWM_CMPBUF5	PWMx_BA+0x330	R	PWM CMPDAT5 Buffer	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
CMPBUF							
7	6	5	4	3	2	1	0
CMPBUF							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CMPBUF**	**PWM Comparator Register Buffer (Read Only)** Used as CMP active register.

PWM_FTCMPDAT Buffer (PWM_FTCBUF0_1,2_3,4_5)

Register	Offset	R/W	Description	Reset Value
PWM_FTCBUF0_1	PWMx_BA+0x340	R	PWM FTCMPDAT0_1 Buffer	0x0000_0000
PWM_FTCBUF2_3	PWMx_BA+0x344	R	PWM FTCMPDAT2_3 Buffer	0x0000_0000
PWM_FTCBUF4_5	PWMx_BA+0x348	R	PWM FTCMPDAT4_5 Buffer	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FTCMPBUF							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FTCMPBUF							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**FTCMPBUF**	**PWM FTCMPDAT Buffer (Read Only)** Used as FTCMPDAT active register.

PWM FTCMPDAT Indicator Register (PWM_FTCI)

Register	Offset	R/W	Description	Reset Value
PWM_FTCI	PWMx_BA+0x34C	R/W	PWM FTCMPDAT Indicator Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**FTCMD4**	**FTCMD2**	**FTCMD0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**FTCMU4**	**FTCMU2**	**FTCMU0**

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:8]	**FTCMDn**	**PWM FTCMPDAT Down Indicator** Indicator will be set to high when FTCMPDATn equal to PERIODn and DIRF=0, software can write 1 to clear this bit. Each bit n controls the corresponding PWM channel n.
[7:3]	**Reserved**	Reserved.
[2:0]	**FTCMUn**	**PWM FTCMPDAT Up Indicator** Indicator will be set to high when FTCMPDATn equal to PERIODn and DIRF=1, software can write 1 to clear this bit. Each bit n controls the corresponding PWM channel n.

6.10 Watchdog Timer (WDT)

6.10.1 Overview

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

6.10.2 Features

- 18-bit free running up counter for WDT time-out interval
- Selectable time-out interval (2^4 ~ 2^{18}) and the time-out interval is 1.6 ms ~ 26.214s if WDT_CLK = 10 kHz.
- System kept in reset state for a period of (1 / WDT_CLK) * 63
- Supports selectable WDT reset delay period, including 1026、130、18 or 3 WDT_CLK reset delay period
- Supports to force WDT enabled after chip powered on or reset by setting CWDTEN[2:0] in Config0 register
- Supports WDT time-out wake-up function only if WDT clock source is selected as LIRC or LXT.

6.10.3 Block Diagram

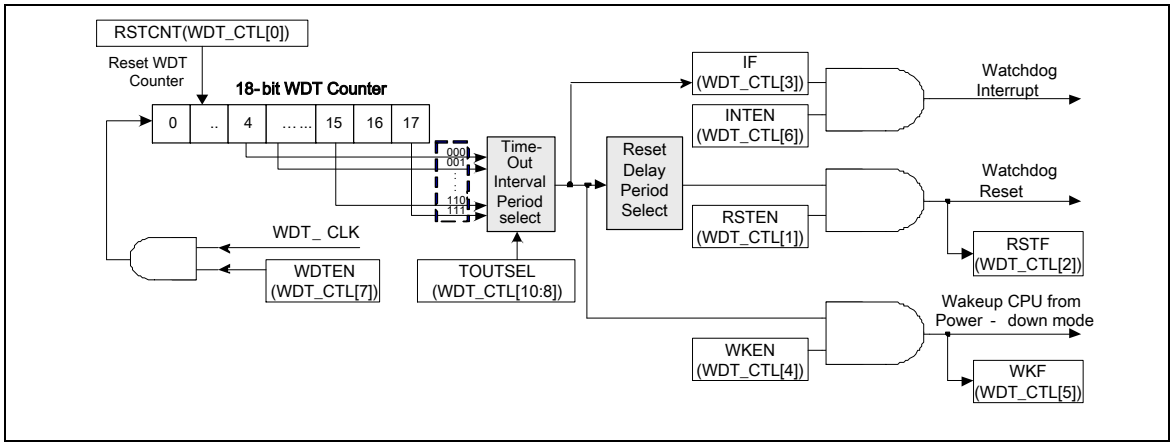


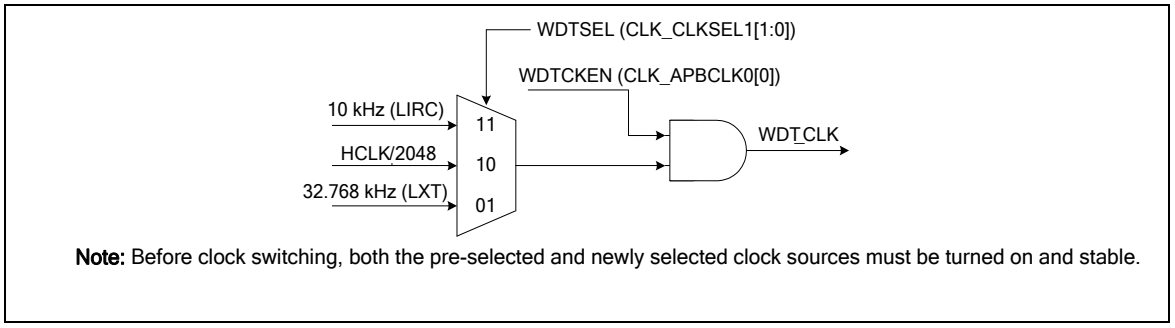
Figure 6.10-1 Watchdog Timer Block Diagram

Note1: WDT resets CPU and lasts 63 WDT_CLK.

Note2: The WDT reset delay period can be selected as 3/18/130/1026 WDT_CLK.

6.10.4 Clock Control

The WDT clock control are shown as follows.



Note: Before clock switching, both the pre-selected and newly selected clock sources must be turned on and stable.

Figure 6.10-2 Watchdog Timer Clock Control

6.10.5 Basic Configuration

The WDT peripheral clock is enabled in WDTCKEN (CLK_APBCLK0[0]) and clock source can be selected in WDTSEL (CLK_CLKSEL1[1:0]).

WDT controller also can be forced enabled and active in 10 kHz after chip powered on or reset while CWDTEN[2:0] (CWDTEN[2] is Config0[31], CWDTEN[1:0] is Config0[4:3]) is not configure to 111.

6.10.6 Functional Description

The WDT includes an 18-bit free running up counter with programmable time-out intervals. Table 6-16 shows the WDT time-out interval period selection and Figure 6.10-3 shows the WDT time-out interval and reset period timing.

6.10.6.1 WDT Time-out Interrupt

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]) settings, WDT time-out interrupt will occur then WDT time-out interrupt flag IF (WDT_CTL[3]) will be set to 1 immediately.

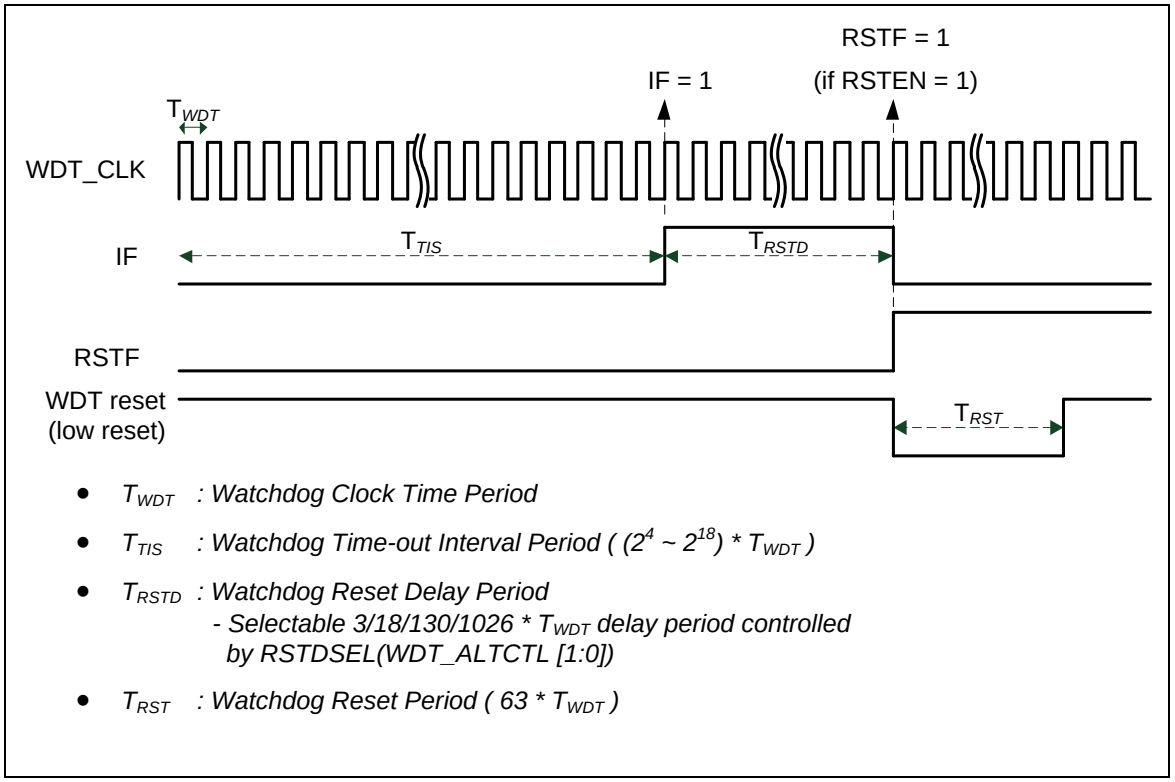
6.10.6.2 WDT Reset Delay Period and Reset System

There is a specified T_{RSTD} reset delay period follows the IF (WDT_CTL[3]) is setting to 1. User should set RSTCNT (WDT_CTL[0]) to reset the 18-bit WDT up counter value to avoid generate WDT time-out reset signal before the T_{RSTD} 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 specific T_{RSTD} delay period expires, the WDT control will set RSTF (WDT_CTL[2]) to 1 if RSTEN (WDT_CTL[1]) bit is enabled, then chip enters to reset state immediately. Refer to , T_{RST} reset period will keep last 63 WDT clocks then chip restart executing program from reset vector (0x0000_0000). The RSTF (WDT_CTL[2]) will keep 1 after WDT time-out reset the chip, user can check RSTF (WDT_CTL[2]) by software to recognize the system has been reset by WDT time-out reset or not.

TOUTSEL	Time-Out Interval Period T_{TIS}	Reset Delay Period T_{RSTD}
000	$2^4 * T_{WDT}$	(3/18/130/1026) * T_{WDT}
001	$2^6 * T_{WDT}$	(3/18/130/1026) * T_{WDT}

010	$2^8 * T_{WDT}$	(3/18/130/1026) * T_{WDT}
011	$2^{10} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
100	$2^{12} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
101	$2^{14} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
110	$2^{16} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
111	$2^{18} * T_{WDT}$	(3/18/130/1026) * T_{WDT}

Table 6-16 Watchdog Timer Time-out Interval Period Selection



- T_{WDT} : *Watchdog Clock Time Period*
- T_{TIS} : *Watchdog Time-out Interval Period ((2^4 ~ 2^{18}) * T_{WDT})*
- T_{RSTD} : *Watchdog Reset Delay Period*
  - *Selectable 3/18/130/1026 * T_{WDT} delay period controlled by RSTDSEL(WDT_ALTCTL [1:0])*
- T_{RST} : *Watchdog Reset Period (63 * T_{WDT})*

Figure 6.10-3 Watchdog Timer Time-out Interval and Reset Period Timing

6.10.6.3 WDT Wake-up

If WDT clock source is selected to LIRC or LXT, system can be waken-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 set to 1 automatically, user can check WKF (WDT_CTL[5]) status by software to recognize the system has been waken-up by WDT time-out interrupt or not.

6.10.7 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 = 0x4004_0000**				
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

6.10.8 Register Description

WDT Control Register (WDT_CTL)

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

31	30	29	28	27	26	25	24
ICEDEBUG	**Reserved**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**TOUTSEL**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WDTEN	**INTEN**	**WKF**	**WKEN**	**IF**	**RSTF**	**RSTEN**	**RSTCNT**

Bits	Description	
[31]	**ICEDEBUG**	**ICE Debug Mode Acknowledge Disable Control (Write Protect)** 0 = ICE debug mode acknowledgement affects WDT counting. WDT up counter will be held while CPU is held by ICE. 1 = ICE debug mode acknowledgement 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.
[30:11]	**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 Control (Write Protect)** 0 = WDT Disabled (This action will reset the internal up counter value). 1 = WDT Enabled. **Note1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note2:** If CWDTEN[2:0] (combined by Config0[31] and Config0[4:3]) bits is not configure to 111, this bit is forced as 1 and user cannot change this bit to 0.
[6]	**INTEN**	**WDT Time-out Interrupt Enable Control (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 Idle or Power-down 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. **Note1:** This bit is write protected. Refer to the SYS_REGLCTL register. **Note2:** Chip can be woken-up by WDT time-out interrupt signal generated only if WDT clock source is selected to LIRC or LXT.
[3]	**IF**	**WDT Time-out Interrupt Flag** This bit will 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 Control (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.

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

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**RSTDSEL**	

Bits	Description	
[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.

6.11 Window Watchdog Timer (WWDT)

6.11.1 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.

6.11.2 Features

- 6-bit down counter value (CNTDAT) and 6-bit compare value (CMPDAT) to make the WWDT time-out window period flexible
- Supports 4-bit value (PSCSEL) to programmable maximum 11-bit prescale counter period of WWDT counter

6.11.3 Block Diagram

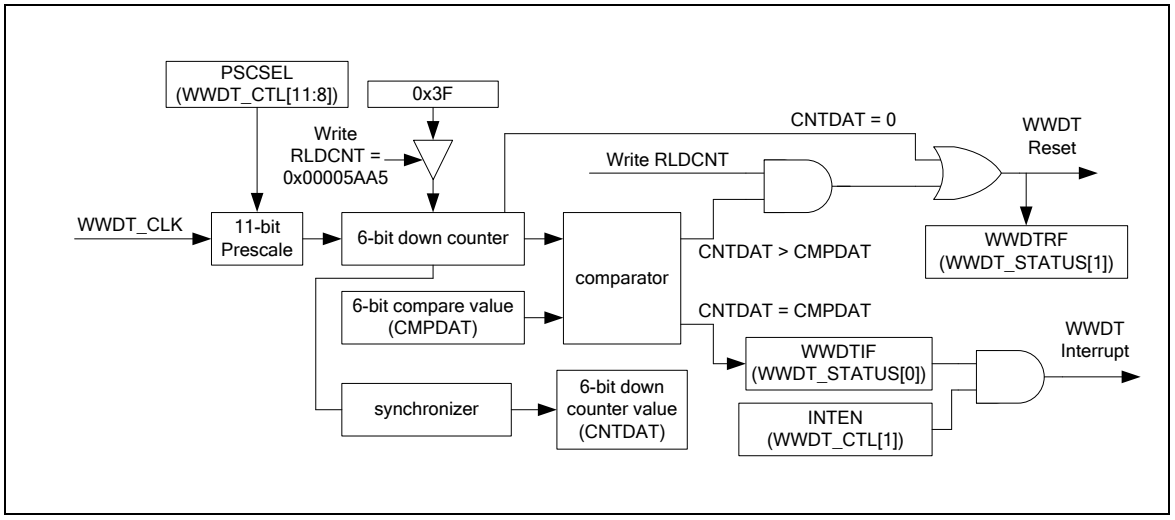
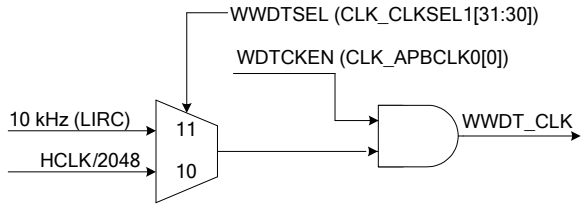


Figure 6.11-1 WWDT Block Diagram

6.11.4 Clock Control

The WWDT clock control are shown as Figure 6.11-2.



Note: Before clock switching, both the pre-selected and newly selected clock sources must be turned on and stable.

Figure 6.11-2 WWDT Clock Control

6.11.5 Basic Configuration

The WWDT peripheral clock is enabled in WDTCKEN (CLK_APBCLK0[0]) and clock source can be selected in WWDTSEL[1:0] (CLK_CLKSEL1[31:30]).

6.11.6 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 10 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 the Table 6-17.

PSCSEL	Prescaler Value	Max. Time-Out Period	Max. Time-Out Interval (WWDT_CLK=10 KHz)
0000	1	1 * 64 * T_{WWDT}	6.4 ms
0001	2	2 * 64 * T_{WWDT}	12.8 ms
0010	4	4 * 64 * T_{WWDT}	25.6 ms
0011	8	8 * 64 * T_{WWDT}	51.2 ms
0100	16	16 * 64 * T_{WWDT}	102.4 ms
0101	32	32 * 64 * T_{WWDT}	204.8 ms
0110	64	64 * 64 * T_{WWDT}	409.6 ms
0111	128	128 * 64 * T_{WWDT}	819.2 ms
1000	192	192 * 64 * T_{WWDT}	1.2288 s
1001	256	256 * 64 * T_{WWDT}	1.6384 s
1010	384	384 * 64 * T_{WWDT}	2.4576 s
1011	512	512 * 64 * T_{WWDT}	3.2768 s
1100	768	768 * 64 * T_{WWDT}	4.9152 s

1101	1024	1024 * 64 * T_{WWDT}	6.5536 s
1110	1536	1536 * 64 * T_{WWDT}	9.8304 s
1111	2048	2048 * 64 * T_{WWDT}	13.1072 s

Table 6-17 WWDT Prescaler Value Selection

6.11.6.1 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.

6.11.6.2 WWDT Compare Match Interrupt

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 WWDTIF can be cleared by user; if INTEN (WWDT_CTL[1]) is also set to 1 by user, the WWDT compare match interrupt signal is generated also while WWDTIF is set to 1 by hardware.

6.11.6.3 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 info 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.

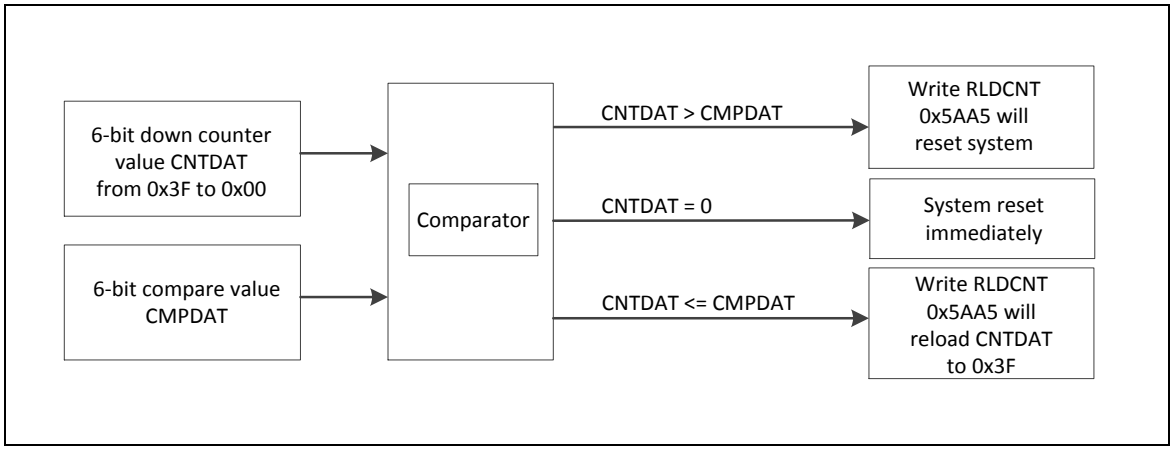


Figure 6.11-3 WWDT Reset and Reload Behavior

6.11.6.4 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.

PSCSEL	Prescale Value	Valid CMPDAT Value
0000	1	0x3 ~ 0x3F
0001	2	0x2 ~ 0x3F
Others	Others	0x0 ~ 0x3F

Table 6-18 CMPDAT Setting Limitation

6.11.7 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 = 0x4004_0100**				
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

6.11.8 Register Description

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

31	30	29	28	27	26	25	24
RLDCNT							
23	22	21	20	19	18	17	16
RLDCNT							
15	14	13	12	11	10	9	8
RLDCNT							
7	6	5	4	3	2	1	0
RLDCNT							

Bits	Description	
[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.

WWDT Control Register (WWDT_CTL)

Register	Offset	R/W	Description	Reset Value
WWDT_CTL	WWDT_BA+0x04	R/W	WWDT Control Register	0x003F_0800

Note: This register can be write only one time after chip is powered on or reset.

31	30	29	28	27	26	25	24
ICEDEBUG	**Reserved**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved		**CMPDAT**					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**PSCSEL**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**INTEN**	**WWDTEN**

Bits	Description	
[31]	**ICEDEBUG**	**ICE Debug Mode Acknowledge Disable Control** 0 = ICE debug mode acknowledgement effects WWDT counting. WWDT down counter will be held while CPU is held by ICE. 1 = ICE debug mode acknowledgement 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 Register** 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 Selection** 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 Control 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 Control Bit** Set this bit to enable WWDT counter counting. 0 = WWDT counter is stopped. 1 = WWDT counter is starting counting.

WWDT Status Register (WWDT_STATUS)

Register	Offset	R/W	Description	Reset Value
WWDT_STATUS	WWDT_BA+0x08	R/W	WWDT Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						WWDTRF	WWDTIF

Bits	Description	
[31:2]	**Reserved**	Reserved.
[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.

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

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**CNTDAT**					

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**CNTDAT**	**WWDT Counter Value** CNTDAT will be updated continuously to monitor 6-bit WWDT down counter value.

6.12 Real Time Clock (RTC)

6.12.1 Overview

The Real Time Clock (RTC) controller provides the real time and calendar message. The RTC offers programmable time tick and alarm match interrupts. The data format of time and calendar messages are expressed in BCD format. A digital frequency compensation feature is available to compensate external crystal oscillator frequency accuracy.

The RTC controller also offers 80 bytes spare registers to store user's important information. The spare registers content is cleared when specified event on tamper pin is detected.

6.12.2 Features

- Supports real time counter in RTC_TIME (hour, minute, second) and calendar counter in RTC_CAL (year, month, day) for RTC time and calendar check
- Supports alarm time (hour, minute, second) and calendar (year, month, day) settings in RTC_TALM and RTC_CALM
- Supports alarm time (hour, minute, second) and calendar (year, month, day) mask enable in RTC_TAMSK and RTC_CAMSK
- Selectable 12-hour or 24-hour time scale in RTC_CLKFMT register
- Supports Leap Year indication in RTC_LEAPYEAR register
- Supports Day of the Week counter in RTC_WEEKDAY register
- Frequency of RTC clock source compensate by RTC_FREQADJ register
- All time and calendar message expressed in BCD format
- Supports periodic RTC Time Tick interrupt with 8 period interval options 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second
- Supports RTC Time Tick and Alarm Match interrupt
- Supports chip wake-up from Idle or Power-down mode while a RTC interrupt signal is generated
- Supports 80 bytes spare registers and a snoop pin detection to clear the content of these spare registers

6.12.3 Block Diagram

The RTC block diagram is shown below.

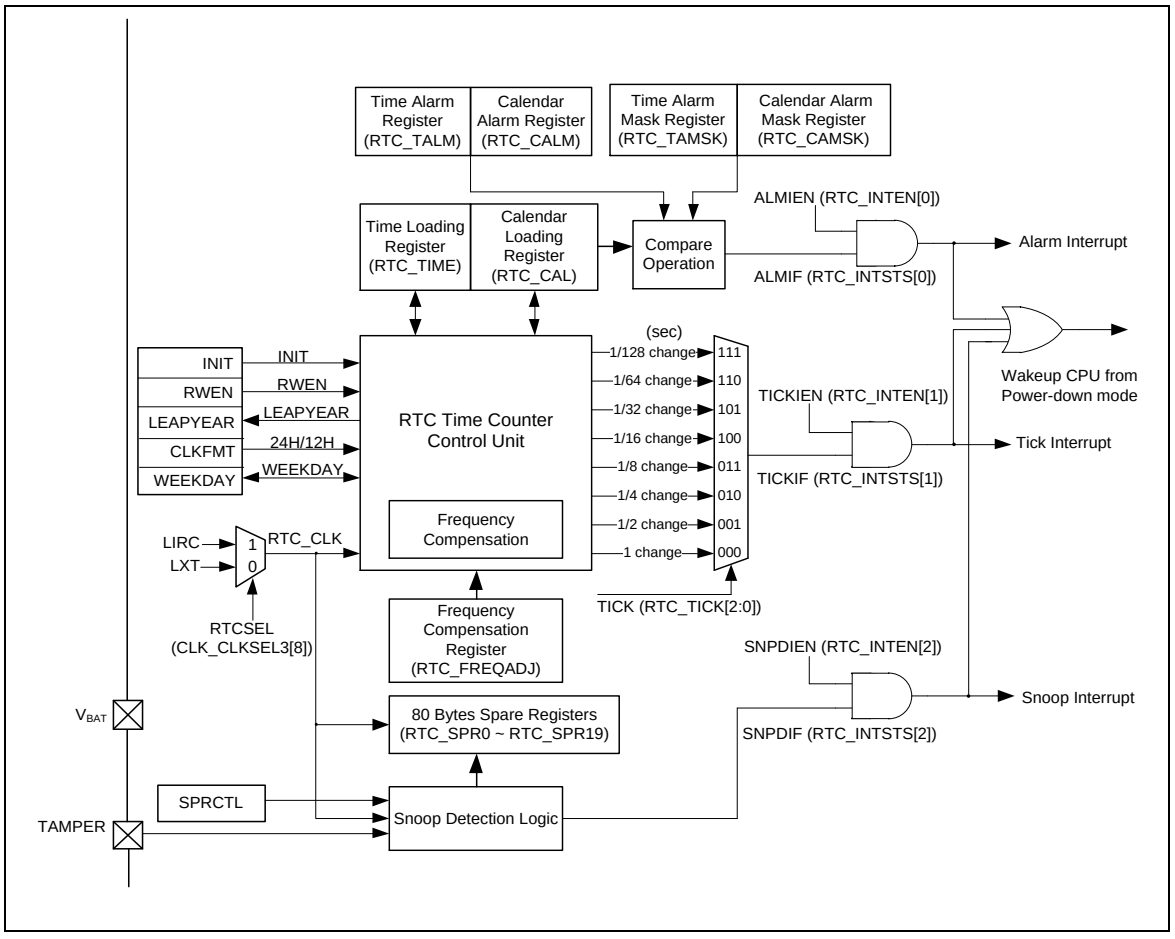


Figure 6.12-1 RTC Block Diagram

6.12.4 Basic Configuration

The RTC controller clock source is enabled by RTCCKEN (APBCLK[1]) and low speed 32 kHz oscillator is enabled by LXTEN (CLK_PWRCTL[1]).

6.12.5 Functional Description

6.12.5.1 RTC Initiation

When a RTC block is powered on, RTC is at reset state. User has to write a number 0xa5eb1357 to RTC initial register RTC_INIT (INIT[31:0]) to make RTC leaving reset state. Once the RTC_INIT register is written as 0xa5eb1357, the RTC will be in normal active state permanently. User can read Active bit (INIT[0]) to check the RTC is at normal active state or reset state.

6.12.5.2 Access to RTC register

Due to clock frequency difference between RTC clock and system clock, when user writes new data to any one of the RTC registers, the data will not be updated until 2 RTC clocks later (about 60us).

In addition, user must be aware that RTC controller does not check whether loaded data is out of bounds or not in RTC_TIME, RTC_CAL, RTC_TALM and RTC_CALM registers. RTC does not

check rationality between RTC_WEEKDAY and RTC_CAL either.

6.12.5.3 RTC Read/Write Enable

The RWEN bits (RTC_RWEN[15:0]) is served as read/write access of RTC registers to unlock register read/write protection function. If RTC_RWEN[15:0] is written to 0xa965, user can read register access enable flag RWENF (RTC_RWEN[16]) to check the RTC registers are read/write accessible or locked. Once RWENF bit is enabled, RTC access enable function will keep effect at least 1024 RTC clocks (about 30ms) and RWENF bit will be cleared automatically after 1024 RTC clocks. The RTC control registers access attribute when RWENF is 1 and 0 are shown in the Table 6-19.

Register	RWENF=1	RWENF=0
RTC_INIT	R/W	R/W
RTC_RWEN	R/W	R/W
RTC_FREQADJ	R/W	Not available
RTC_TIME	R/W	R
RTC_CAL	R/W	R
RTC_CLKFMT	R/W	R/W
RTC_WEEKDAY	R/W	R
RTC_TALM	R/W	Not available
RTC_CALM	R/W	Not available
MRTC_TALM	R/W	R
RTC_CAMSK	R/W	R
RTC_LEAPYEAR	R	R
RTC_INTEN	R/W	R/W
RTC_INTSTS	R/W	R/W
RTC_TICK	R/W	Not available
RTC_SPRCTL	R/W	Not available
RTC_SPR0-RTC_SPR19	R/W	Not available
RTC_LXTCTL	R/W	Not available
RTC_LXTOCTL	R/W	Not available
RTC_LXTICTL	R/W	Not available
RTC_TAMPCTL	R/W	Not available

Table 6-19 RTC control registers access attribute

6.12.5.4 Frequency Compensation

The RTC source clock may not precise to exactly 32768 Hz and the RTC_FREQADJ register allows user to make digital compensation to the RTC source clock only if the frequency of RTC source clock is in the range from 32761 Hz to 32776 Hz.

Integer Part Of Detected Value	INTEGER (RTC_FREQADJ[11:8])	Integer Part Of Detected Value	INTEGER (RTC_FREQADJ[11:8])

32776	1111	32768	0111
32775	1110	32767	0110
32774	1101	32766	0101
32773	1100	32765	0100
32772	1011	32764	0011
32771	1010	32763	0010
32770	1001	32762	0001
32769	1000	32761	0000

Following are the compensation examples for the real RTC source clock is higher or lower than 32768 Hz.

Example 1: (RTC Source Clock > 32768 Hz) **RTC Source Clock Measured: 32773.65 Hz (> 32768 Hz)Integer Part: 32773 => 0x8005** **If Integer Part Of Detected Value Is 32768, RTC_FREQADJ[11:8] Is Assigned To Be 0111b.** **Now That Integer Part Of Detected Value Is 32773, The Corresponding RTC_FREQADJ[11:8] Can Be Calculated By:** **0111 (RTC_FREQADJ[11:8] For 32768, Binary) + 5 (32773 – 32768, Decimal) = 0xC (Hexadecimal).** **Fraction Part: 0.65** **FRACTION (RTC_FREQADJ [5:0] Fraction Part) = 0.65 X 60 = 39 = 0x27** **RTC_FREQADJ Register Should Be As 0xC27**
Example 2: (RTC source clock ≦ 32768 Hz) RTC source clock measured: 32765.27 Hz (≦ 32768 Hz) Integer part: 32765 => 0x7FFD If integer part of detected value is 32768, RTC_FREQADJ[11:8] is assigned to be 0111b. Now that integer part of detected value is 32765, the corresponding RTC_FREQADJ[11:8] can be calculated by: 0111 (RTC_FREQADJ[11:8] for 32768, binary) + (-3) (32765 – 32768, decimal) = 0x4 (hexadecimal). Fraction part: 0.27 FRACTION (RTC_FREQADJ [5:0] Fraction Part) = 0.27 x 60 = 16.2 = 0x10 RTC_FREQADJ register should be as 0x410

6.12.5.5 Time and Calendar counter

RTC_TIME and RTC_CAL are used to load the real time and calendar. RTC_TALM and RTC_CALM are used for setup alarm time and calendar.

6.12.5.6 12/24 hour Time Scale Selection

The 12/24 hour time scale selection depends on 24HEN (RTC_CLKFMT[0]).

When RTC runs as 12-hour time scale mode, RTC_TIME[21] (the high bit of TENHR[1:0]) means AM/PM indication. (If RTC_TIME[21] is 1, it indicates PM time message.)

Note: The Hour Value Write Into RTC_TIME[21:16], Messages Are Expressed In BCD Format.			
24-Hour Time Scale (24HEN = 1)	**24-Hour Time Scale (24HEN = 1)**	**12-Hour Time Scale (24HEN = 0)**	**12-Hour Time Scale (24HEN = 0) (PM Time + 20)**
0x00	0x12	0x12 (AM12)	0x32 (PM12)

0x01	0x13	0x01 (AM01)	0x21 (PM01)
0x02	0x14	0x02 (AM02)	0x22 (PM02)
0x03	0x15	0x03 (AM03)	0x23 (PM03)
0x04	0x16	0x04 (AM04)	0x24 (PM04)
0x05	0x17	0x05 (AM05)	0x25 (PM05)
0x06	0x18	0x06 (AM06)	0x26 (PM06)
0x07	0x19	0x07 (AM07)	0x27 (PM07)
0x08	0x20	0x08 (AM08)	0x28 (PM08)
0x09	0x21	0x09 (AM09)	0x29 (PM09)
0x10	0x22	0x10 (AM10)	0x30 (PM10)
0x11	0x23	0x11 (AM11)	0x31 (PM11)

6.12.5.7 Day of the Week Counter

The RTC controller provides day of week in WEEKDAY bits (RTC_WEEKDAY[2:0]). The value is defined from 0 to 6 to represent Sunday to Saturday respectively.

6.12.5.8 Periodic Time Tick Interrupt

The Periodic Time Tick interrupt has 8 period interval options 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second that are selected by TICK bits (RTC_TICK[2:0]). When Periodic Time Tick interrupt is enabled by setting TICKIEN (RTC_INTEN[1]) to 1, the Periodic Time Tick interrupt is requested periodically in the period selected by RTC_TICK[2:0] settings.

6.12.5.9 Alarm Interrupt

When the real time and calendar message in RTC_TIME and RTC_CAL registers are equal to alarm time and calendar values in RTC_TALM and RTC_CALM registers, the RTC alarm interrupt flag ALMIF (RTC_INTSTS[0]) is set to 1 and the RTC alarm interrupt signal assert if the alarm interrupt enable ALMIEN (RTC_INTEN[0]) is enabled.

The RTC controller provides Time Alarm Mask Register (RTC_TAMSK register) and Calendar Alarm Mask Register (RTC_CAMSK register) to mask the specified digit and generate periodic interrupt without changing the alarm match condition in RTC_TALM and RTC_CALM registers in each alarm interrupt service routine.

6.12.5.10 Application Note

1. All data in RTC_TALM, RTC_CALM, RTC_TIME and RTC_CAL registers are all expressed in BCD format.
2. User has to make sure that the loaded values are reasonable. For example, Load RTC_CAL as 201a (year), 13 (month), 00 (day), or RTC_CAL does not match with RTC_WEEKDAY, etc.
3. Registers value after powered on:

Register	Reset State
RTC_RWEN	0
RTC_CAL	05/1/1 (year/month/day)
RTC_TIME	00:00:00 (hour : minute : second)

RTC_CALM	00/00/00 (year/month/day)
RTC_TALM	00:00:00 (hour : minute : second)
RTC_CLKFMT	1 (24-hour mode)
RTC_WEEKDAY	6 (Saturday)
RTC_INTEN	0
RTC_INTSTS	0
RTC_LEAPYEAR	0
RTC_TICK	0

4. In RTC_CAL and RTC_CALM, only 2 BCD digits are used to express “year”. The 2 BCD digits of xy means 20xy, rather than 19xy or 21xy.
5. Example of 12-Hour Time Setting
   If current RTC time is PM12:59:30 in 12-Hour Time Scale mode, the RTC_TIME setting as:
   RTC_TIME[21:16]: 0x32 (0x12+0x20)
   TENHR (RTC_TIME[21:20]) is 0x3, HR (RTC_TIME[19:16]) is 0x2.
   RTC_TIME[14:8]: 0x59
   TENMIN (RTC_TIME[14:12]) is 0x5, MIN (RTC_TIME[11:8]) is 0x9.
   RTC_TIME[6:0]: 0x30
   TENSEC (RTC_TIME[6:4]) is 0x3, SEC (RTC_TIME[3:0]) is 0x0.
6. Which power domain the register is in:

Register	Power Domain
RTC_FREQADJ	Battery Power Domain
RTC_TIME	Core Power Domain
RTC_CAL	Core Power Domain
RTC_CLKFMT	Core Power Domain
RTC_WEEKDAY	Core Power Domain
RTC_TALM	Battery Power Domain
RTC_CALM	Battery Power Domain
RTC_LEAPYEAR	Core Power Domain
RTC_INTEN	Core Power Domain
RTC_INTSTS	Core Power Domain
RTC_TICK	Battery Power Domain
RTC_TAMSK	Battery Power Domain
RTC_CAMSK	Battery Power Domain
RTC_SPRCTL	Battery Power Domain
RTC_SPR0	Battery Power Domain
RTC_SPR1	Battery Power Domain
RTC_SPR2	Battery Power Domain
RTC_SPR3	Battery Power Domain
RTC_SPR4	Battery Power Domain

RTC_SPR5	Battery Power Domain
RTC_SPR6	Battery Power Domain
RTC_SPR7	Battery Power Domain
RTC_SPR8	Battery Power Domain
RTC_SPR9	Battery Power Domain
RTC_SPR10	Battery Power Domain
RTC_SPR11	Battery Power Domain
RTC_SPR12	Battery Power Domain
RTC_SPR13	Battery Power Domain
RTC_SPR14	Battery Power Domain
RTC_SPR15	Battery Power Domain
RTC_SPR16	Battery Power Domain
RTC_SPR17	Battery Power Domain
RTC_SPR18	Battery Power Domain
RTC_SPR19	Battery Power Domain
RTC_LXTCTL	Battery Power Domain
RTC_LXTOCTL	Battery Power Domain
RTC_LXTICTL	Battery Power Domain
RTC_TAMPCTL	Battery Power Domain

6.12.5.11 Spare registers and snoop event detect

The RTC module is equipped with 80 bytes spare registers to store user's important information. These spare registers are located in RTC clock domain, user needs to enable SPRRWEN (RTC_SPRCTL[2]) before writing one of 20 spare registers (RTC_SPR0 ~ RTC_SPR19). User could read SPRRWRDY (RTC_SPRCTL[7]) to check if data has been written into registers or not. User could only access the spare registers again once SPRRWRDY is 1. Any access to spare registers is available if SPRRWRDY is 0.

If external 32 kHz clock (LXT) is not available in system design, user can choose RTCSEL (CLK_CLKSEL3[8]) to 1 to use on chip 10 kHz internal low speed RC oscillator (LIRC) instead of for spare registers read and write operations.

Snoop detection function to detect the transition of TAMPER pin. When the transition condition defined in SNPTYPE1 (RTC_SPRCTL[3]) and SNPTYPE0 (RTC_SPRCTL[1]) is detected then 80 bytes spare registers (RTC_SPR0 ~ RTC_SPR19) content will be cleared by hardware automatically to prevent the security data be disclosure. Snoop detected interrupt flag SNPDIF (RTC_INTSTS[2]) is set to 1 and interrupt is generated to NVIC if snoop detect interrupt enable SNPDIEN (RTC_INTEN[2]) is enabled. Snoop detection condition is listed below:

SNPTYPE1	SNPTYPE0	Detect Event Condition
1	1	Rising edge

1	0	Falling edge
0	1	High Level
0	0	Low level

6.12.5.12 Backup Domain GPIO Function

When PF.0/X32O and PF.1/X32I pins are not used as low speed 32K oscillator function, they can be used as GPIO pin function. The CTLSEL (RTC_LXTOCTL[3]) is used to select the PF.0/X32O pin is controlled by RTC or GPIO module. The PF.1/X32I and PF.2/TAMPER pins are controlled by CTLSEL in RTC_LXTICTL[3] and RTC_TAMPCTL[3] respectively.

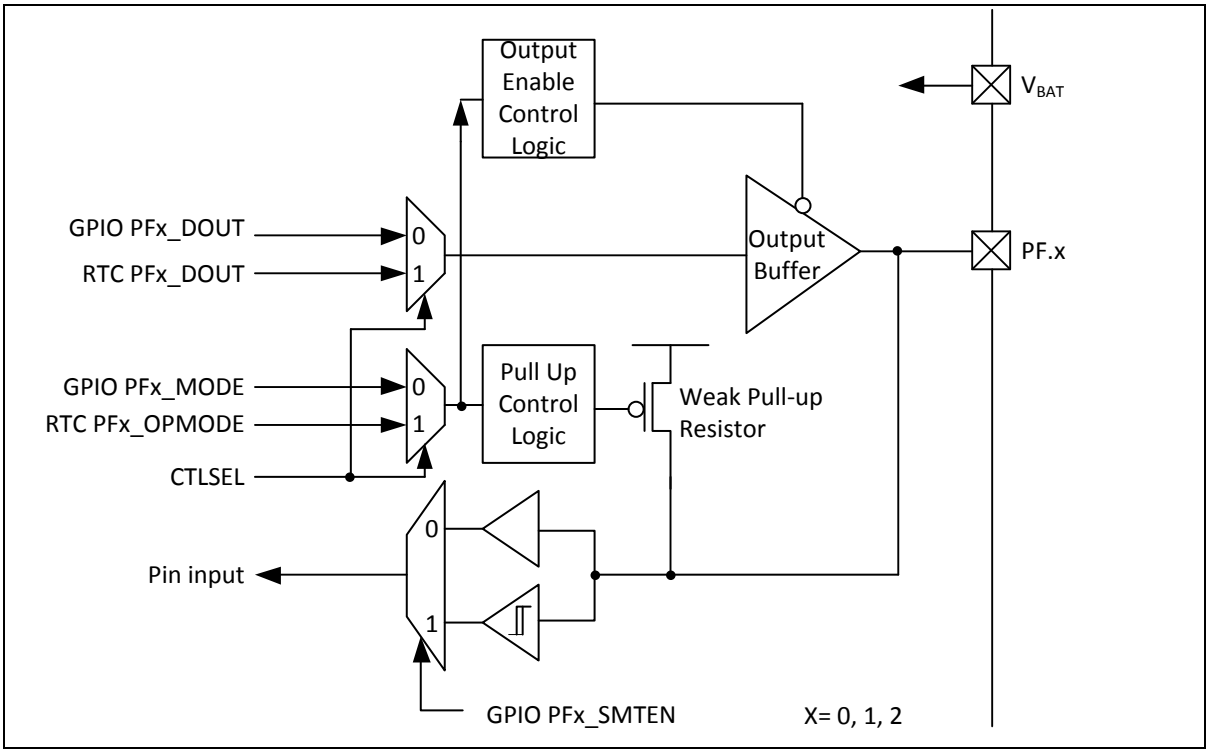


Figure 6.12-2 Backup I/O Control Diagram

In the figure, GPIO PFx_MODE means the GPIO register PF_MODE. While, RTC PFx_OPMODE means the RTC registers RTC_LXTOCTL[1:0] and RTC_LXTICTL[1:0]. i.e. OPMODE.

6.12.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
RTC Base Address: **RTC_BA = 0x4004_1000**				
RTC_INIT	RTC_BA+0x00	R/W	RTC Initiation Register	0x0000_0000
RTC_RWEN	RTC_BA+0x04	R/W	RTC Access Enable Register	0x0000_0000
RTC_FREQADJ	RTC_BA+0x08	R/W	RTC Frequency Compensation Register	0x0000_0700
RTC_TIME	RTC_BA+0x0C	R/W	RTC Time Loading Register	0x0000_0000
RTC_CAL	RTC_BA+0x10	R/W	RTC Calendar Loading Register	0x0005_0101
RTC_CLKFMT	RTC_BA+0x14	R/W	RTC Time Scale Selection Register	0x0000_0001
RTC_WEEKDAY	RTC_BA+0x18	R/W	RTC Day of the Week Register	0x0000_0006
RTC_TALM	RTC_BA+0x1C	R/W	RTC Time Alarm Register	0x0000_0000
RTC_CALM	RTC_BA+0x20	R/W	RTC Calendar Alarm Register	0x0000_0000
RTC_LEAPYEAR	RTC_BA+0x24	R	RTC Leap Year Indicator Register	0x0000_0000
RTC_INTEN	RTC_BA+0x28	R/W	RTC Interrupt Enable Register	0x0000_0000
RTC_INTSTS	RTC_BA+0x2C	R/W	RTC Interrupt Indicator Register	0x0000_0000
RTC_TICK	RTC_BA+0x30	R/W	RTC Time Tick Register	0x0000_0000
RTC_TAMSK	RTC_BA+0x34	R/W	RTC Time Alarm Mask Register	0x0000_0000
RTC_CAMSK	RTC_BA+0x38	R/W	RTC Calendar Alarm Mask Register	0x0000_0000
RTC_SPRCTL	RTC_BA+0x3C	R/W	RTC Spare Functional Control Register	0x0000_0080
RTC_SPR0	RTC_BA+0x40	R/W	RTC Spare Register 0	0x0000_0000
RTC_SPR1	RTC_BA+0x44	R/W	RTC Spare Register 1	0x0000_0000
RTC_SPR2	RTC_BA+0x48	R/W	RTC Spare Register 2	0x0000_0000
RTC_SPR3	RTC_BA+0x4C	R/W	RTC Spare Register 3	0x0000_0000
RTC_SPR4	RTC_BA+0x50	R/W	RTC Spare Register 4	0x0000_0000
RTC_SPR5	RTC_BA+0x54	R/W	RTC Spare Register 5	0x0000_0000
RTC_SPR6	RTC_BA+0x58	R/W	RTC Spare Register 6	0x0000_0000
RTC_SPR7	RTC_BA+0x5C	R/W	RTC Spare Register 7	0x0000_0000
RTC_SPR8	RTC_BA+0x60	R/W	RTC Spare Register 8	0x0000_0000
RTC_SPR9	RTC_BA+0x64	R/W	RTC Spare Register 9	0x0000_0000

RTC_SPR10	RTC_BA+0x68	R/W	RTC Spare Register 10	0x0000_0000
RTC_SPR11	RTC_BA+0x6C	R/W	RTC Spare Register 11	0x0000_0000
RTC_SPR12	RTC_BA+0x70	R/W	RTC Spare Register 12	0x0000_0000
RTC_SPR13	RTC_BA+0x74	R/W	RTC Spare Register 13	0x0000_0000
RTC_SPR14	RTC_BA+0x78	R/W	RTC Spare Register 14	0x0000_0000
RTC_SPR15	RTC_BA+0x7C	R/W	RTC Spare Register 15	0x0000_0000
RTC_SPR16	RTC_BA+0x80	R/W	RTC Spare Register 16	0x0000_0000
RTC_SPR17	RTC_BA+0x84	R/W	RTC Spare Register 17	0x0000_0000
RTC_SPR18	RTC_BA+0x88	R/W	RTC Spare Register 18	0x0000_0000
RTC_SPR19	RTC_BA+0x8C	R/W	RTC Spare Register 19	0x0000_0000
RTC_LXTCTL	RTC_BA+0x100	R/W	RTC 32.768 kHz Oscillator Control Register	0x0000_000E
RTC_LXTOCTL	RTC_BA+0x104	R/W	X32KO Pin Control Register	0x0000_0000
RTC_LXTICTL	RTC_BA+0x108	R/W	X32KI Pin Control Register	0x0000_0000
RTC_TAMPCTL	RTC_BA+0x10C	R/W	TAMPER Pin Control Register	0x0000_0000

6.12.7 Register Description

RTC Initiation Register (RTC_INIT)

Register	Offset	R/W	Description	Reset Value
RTC_INIT	RTC_BA+0x00	R/W	RTC Initiation Register	0x0000_0000

31	30	29	28	27	26	25	24
INIT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
INIT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
INIT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INIT							**INIT[0]/ACTIVE**

Bits	Description	
[31:1]	**INIT[31:1]**	**RTC Initiation** When RTC block is powered on, RTC is at reset state. User has to write a number (0x a5eb1357) to INIT to make RTC leaving reset state. Once the INIT is written as 0xa5eb1357, the RTC will be in un-reset state permanently. The INIT is a write-only field and read value will be always 0.
[0]	**INIT[0]/ACTIVE**	**RTC Active Status (Read Only)** 0 = RTC is at reset state. 1 = RTC is at normal active state.

RTC Access Enable Register (RTC_RWEN)

Register	Offset	R/W	Description	Reset Value
RTC_RWEN	RTC_BA+0x04	R/W	RTC Access Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							RWENF
15	14	13	12	11	10	9	8
RWEN							
7	6	5	4	3	2	1	0
RWEN							

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**RWENF**	**RTC Register Access Enable Flag (Read Only)** 0 = RTC register read/write Disabled. 1 = RTC register read/write Enabled. This bit will be set after RTC_RWEN[15:0] register is load a 0xA965, and be cleared automatically after 1024 RTC clock.
[15:0]	**RWEN**	**RTC Register Access Enable Password (Write Only)** Writing 0xA965 to this register will enable RTC access and keep 1024 RTC clock.

RTC Frequency Compensation Register (RTC_FREQADJ)

Register	Offset	R/W	Description	Reset Value
RTC_FREQADJ	RTC_BA+0x08	R/W	RTC Frequency Compensation Register	0x0000_0700

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				INTEGER			
7	6	5	4	3	2	1	0
Reserved		FRACTION					

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:8]	**INTEGER**	**Integer Part**
[5:0]	**FRACTION**	**Fraction Part** Formula = (fraction part of detected value) x 60. **Note:** Digit in RTC_FREQADJ must be expressed as hexadecimal number.

Note: This register can be read back after the RTC register access enable bit RWENF (RTC_RWEN[16]) is active.

RTC Time Loading Register (RTC_TIME)

Register	Offset	R/W	Description	Reset Value
RTC_TIME	RTC_BA+0x0C	R/W	RTC Time Loading Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		TENHR		HR			
15	14	13	12	11	10	9	8
Reserved	TENMIN			MIN			
7	6	5	4	3	2	1	0
Reserved	TENSEC			SEC			

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21:20]	**TENHR**	**10-hour Time Digit (0~2)** When RTC runs as 12-hour time scale mode, RTC_TIME[21] (the high bit of TENHR[1:0]) means AM/PM indication. (If RTC_TIME[21] is 1, it indicates PM time message.)
[19:16]	**HR**	1-Hour Time Digit (0~9)
[15]	**Reserved**	Reserved.
[14:12]	**TENMIN**	10-Min Time Digit (0~5)
[11:8]	**MIN**	1-Min Time Digit (0~9)
[7]	**Reserved**	Reserved.
[6:4]	**TENSEC**	10-Sec Time Digit (0~5)
[3:0]	**SEC**	1-Sec Time Digit (0~9)

Note:

1. RTC_TIME is a BCD digit counter and RTC will not check loaded data.
2. The reasonable value range is listed in the parenthesis.

RTC Calendar Loading Register (RTC_CAL)

Register	Offset	R/W	Description	Reset Value
RTC_CAL	RTC_BA+0x10	R/W	RTC Calendar Loading Register	0x0005_0101

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TENYEAR				**YEAR**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved			**TENMON**	**MON**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**TENDAY**		**DAY**			

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:20]	**TENYEAR**	10-Year Calendar Digit (0~9)
[19:16]	**YEAR**	1-Year Calendar Digit (0~9)
[15:13]	**Reserved**	Reserved.
[12]	**TENMON**	10-Month Calendar Digit (0~1)
[11:8]	**MON**	1-Month Calendar Digit (0~9)
[7:6]	**Reserved**	Reserved.
[5:4]	**TENDAY**	10-Day Calendar Digit (0~3)
[3:0]	**DAY**	1-Day Calendar Digit (0~9)

Note:

1. RTC_CAL is a BCD digit counter and RTC will not check loaded data.
2. The reasonable value range is listed in the parenthesis.

RTC Time Scale Selection Register (RTC_CLKFMT)

Register	Offset	R/W	Description	Reset Value
RTC_CLKFMT	RTC_BA+0x14	R/W	RTC Time Scale Selection Register	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**24HEN**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**24HEN**	**24-hour / 12-hour Time Scale Selection** Indicates that RTC_TIME and RTC_TALM are in 24-hour time scale or 12-hour time scale 0 = 12-hour time scale with AM and PM indication selected. 1 = 24-hour time scale selected.

RTC Day of the Week Register (RTC_WEEKDAY)

Register	Offset	R/W	Description	Reset Value
RTC_WEEKDAY	RTC_BA+0x18	R/W	RTC Day of the Week Register	0x0000_0006

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					WEEKDAY		

Bits	Description	
[31:3]	Reserved	Reserved.
[2:0]	WEEKDAY	Day of the Week Register 000 = Sunday. 001 = Monday. 010 = Tuesday. 011 = Wednesday. 100 = Thursday. 101 = Friday. 110 = Saturday. 111 = Reserved.

RTC Time Alarm Register (RTC_TALM)

Register	Offset	R/W	Description	Reset Value
RTC_TALM	RTC_BA+0x1C	R/W	RTC Time Alarm Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved		TENHR		HR			
15	14	13	12	11	10	9	8
Reserved	TENMIN			MIN			
7	6	5	4	3	2	1	0
Reserved	TENSEC			SEC			

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21:20]	**TENHR**	**10-hour Time Digit of Alarm Setting (0~2)** When RTC runs as 12-hour time scale mode, the high bit of TENHR (RTC_TIME[21]) means AM/PM indication.
[19:16]	**HR**	1-Hour Time Digit of Alarm Setting (0~9)
[15]	**Reserved**	Reserved.
[14:12]	**TENMIN**	10-Min Time Digit of Alarm Setting (0~5)
[11:8]	**MIN**	1-Min Time Digit of Alarm Setting (0~9)
[7]	**Reserved**	Reserved.
[6:4]	**TENSEC**	10-Sec Time Digit of Alarm Setting (0~5)
[3:0]	**SEC**	1-Sec Time Digit of Alarm Setting (0~9)

Note:

1. RTC_TALM is a BCD digit counter and RTC will not check loaded data.
2. The reasonable value range is listed in the parenthesis.
3. This register can be read back after the RTC register access enable bit RWENF (RTC_RWEN[16]) is active.

RTC Calendar Alarm Register (RTC_CALM)

Register	Offset	R/W	Description	Reset Value
RTC_CALM	RTC_BA+0x20	R/W	RTC Calendar Alarm Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
TENYEAR				YEAR			
15	14	13	12	11	10	9	8
Reserved			TENMON	MON			
7	6	5	4	3	2	1	0
Reserved		TENDAY		DAY			

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:20]	**TENYEAR**	10-Year Calendar Digit of Alarm Setting (0~9)
[19:16]	**YEAR**	1-Year Calendar Digit of Alarm Setting (0~9)
[15:13]	**Reserved**	Reserved.
[12]	**TENMON**	10-Month Calendar Digit of Alarm Setting (0~1)
[11:8]	**MON**	1-Month Calendar Digit of Alarm Setting (0~9)
[7:6]	**Reserved**	Reserved.
[5:4]	**TENDAY**	10-Day Calendar Digit of Alarm Setting (0~3)
[3:0]	**DAY**	1-Day Calendar Digit of Alarm Setting (0~9)

Note:

1. RTC_CALM is a BCD digit counter and RTC will not check loaded data.
2. The reasonable value range is listed in the parenthesis.
3. This register can be read back after the RTC register access enable bit RWENF (RTC_RWEN[16]) is active.

RTC Leap Year Indication Register (RTC_LEAPYEAR)

Register	Offset	R/W	Description	Reset Value
RTC_LEAPYEAR	RTC_BA+0x24	R	RTC Leap Year Indicator Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							LEAPYEAR

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	LEAPYEAR	Leap Year Indication Register (Read Only) 0 = This year is not a leap year. 1 = This year is leap year.

RTC Interrupt Enable Register (RTC_INTEN)

Register	Offset	R/W	Description	Reset Value
RTC_INTEN	RTC_BA+0x28	R/W	RTC Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					SNPDIEN	TICKIEN	ALMIEN

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**SNPDIEN**	**Snoop Detection Interrupt Enable Bit** 0 = Snoop detected interrupt Disabled. 1 = Snoop detected interrupt Enabled.
[1]	**TICKIEN**	**Time Tick Interrupt Enable Bit** 0 = RTC Time Tick interrupt Disabled. 1 = RTC Time Tick interrupt Enabled.
[0]	**ALMIEN**	**Alarm Interrupt Enable Bit** 0 = RTC Alarm interrupt Disabled. 1 = RTC Alarm interrupt Enabled.

RTC Interrupt Indication Register (RTC_INTSTS)

Register	Offset	R/W	Description	Reset Value
RTC_INTSTS	RTC_BA+0x2C	R/W	RTC Interrupt Indicator Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					SNPDIF	TICKIF	ALMIF

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**SNPDIF**	**Snoop Detect Interrupt Flag** When tamper pin transition event is detected, this bit is set to 1 and an interrupt is generated if Snoop Detection Interrupt enabled SNPDIEN (RTC_INTEN[2]) is set to1. Chip will be waken up from Power-down mode if spare register snooper detect interrupt is enabled. 0 = No snoop event is detected. 1 = Snoop event is detected. **Note:** Write 1 to clear this bit.
[1]	**TICKIF**	**RTC Time Tick Interrupt Flag** When RTC time tick happened, this bit will be set to 1 and an interrupt will be generated if RTC Tick Interrupt enabled TICKIEN (RTC_INTEN[1]) is set to 1. Chip will also be waken up if RTC Tick Interrupt is enabled and this bit is set to 1 when chip is running at Power-down mode. 0 = Tick condition does not occur. 1 = Tick condition occur. **Note:** Write 1 to clear to clear this bit.
[0]	**ALMIF**	**RTC Alarm Interrupt Flag** When RTC time counters RTC_TIME and RTC_CAL match the alarm setting time registers RTC_TALM and RTC_CALM, this bit will be set to 1 and an interrupt will be generated if RTC Alarm Interrupt enabled ALMIEN (RTC_INTEN[0]) is set to 1. Chip will be waken up if RTC Alarm Interrupt is enabled when chip is at Power-down mode. 0 = Alarm condition is not matched. 1 = Alarm condition is matched. **Note:** Write 1 to clear this bit.

RTC Time Tick Register (RTC_TICK)

Register	Offset	R/W	Description	Reset Value
RTC_TICK	RTC_BA+0x30	R/W	RTC Time Tick Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**TICK**		

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2:0]	**TICK**	**Time Tick Register** These bits are used to select RTC time tick period for Periodic Time Tick Interrupt request. 000 = Time tick is 1 second. 001 = Time tick is 1/2 second. 010 = Time tick is 1/4 second. 011 = Time tick is 1/8 second. 100 = Time tick is 1/16 second. 101 = Time tick is 1/32 second. 110 = Time tick is 1/64 second. 111 = Time tick is 1/28 second. **Note:** This register can be read back after the RTC register access enable bit RWENF (RTC_RWEN[16]) is active.

Note: This register can be read back after the RTC register access enable bit RWENF (RTC_RWEN[16]) is active.

RTC Time Alarm MASK Register (RTC_TAMSK)

Register	Offset	R/W	Description	Reset Value
RTC_TAMSK	RTC_BA+0x34	R/W	RTC Time Alarm Mask Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		MTENHR	MHR	MTENMIN	MMIN	MTENSEC	MSEC

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**MTENHR**	Mask 10-Hour Time Digit of Alarm Setting (0~2)
[4]	**MHR**	Mask 1-Hour Time Digit of Alarm Setting (0~9)
[3]	**MTENMIN**	Mask 10-Min Time Digit of Alarm Setting (0~5)
[2]	**MMIN**	Mask 1-Min Time Digit of Alarm Setting (0~9)
[1]	**MTENSEC**	Mask 10-Sec Time Digit of Alarm Setting (0~5)
[0]	**MSEC**	Mask 1-Sec Time Digit of Alarm Setting (0~9)

Note:

1. RTC_TALM is a BCD digit counter and RTC will not check loaded data.
2. The reasonable value range is listed in the parenthesis.

RTC Calendar Alarm MASK Register (RTC_CAMSK)

Register	Offset	R/W	Description	Reset Value
RTC_CAMSK	RTC_BA+0x38	R/W	RTC Calendar Alarm Mask Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		MTENYEAR	MYEAR	MTENMON	MMON	MTENDAY	MDAY

Bits	Description	
	Reserved	Reserved.
[5]	**MTENYEAR**	Mask 10-Year Calendar Digit of Alarm Setting (0~9)
[4]	**MYEAR**	Mask 1-Year Calendar Digit of Alarm Setting (0~9)
[3]	**MTENMON**	Mask 10-Month Calendar Digit of Alarm Setting (0~1)
[2]	**MMON**	Mask 1-Month Calendar Digit of Alarm Setting (0~9)
[1]	**MTENDAY**	Mask 10-Day Calendar Digit of Alarm Setting (0~3)
[0]	**MDAY**	Mask 1-Day Calendar Digit of Alarm Setting (0~9)

Note:

1. RTC_CALM is a BCD digit counter and RTC will not check loaded data.
2. The reasonable value range is listed in the parenthesis.

RTC Spare Functional Control Register (RTC_SPRCTL)

Register	Offset	R/W	Description	Reset Value
RTC_SPRCTL	RTC_BA+0x3C	R/W	RTC Spare Functional Control Register	0x0000_0080

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
SPRRWRDY	Reserved	SPRCSTS	Reserved	SNPTYPE1	SPRRWEN	SNPTYPE0	SNPDEN

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**SPRRWRDY**	**SPR Register Ready** This bit indicates if the registers RTC_SPRCTL, RTC_SPR0 ~ RTC_SPR19 are ready to be accessed. After user writing registers RTC_SPRCTL, RTC_SPR0 ~ RTC_SPR19, read this bit to check if these registers are updated done is necessary. 0 = RTC_SPRCTL, RTC_SPR0 ~ RTC_SPR19 updating is in progress. 1 = RTC_SPRCTL, RTC_SPR0 ~ RTC_SPR19 are updated done and ready to be accessed. **Note:** This bit is read only and any write to it won't take any effect.
[6:3]	**Reserved**	Reserved.
[5]	**SPRCSTS**	**SPR Clear Flag** This bit indicates if the RTC_SPR0 ~RTC_SPR19 content is cleared when specify snoop event is detected. 0 = Spare register content is not cleared. 1 = Spare register content is cleared. Writes 1 to clear this bit.
[4]	**Reserved**	Reserved.
[3]	**SNPTYPE1**	**Snoop Detection Mode** This bit controls TAMPER pin is edge or level detection 0 = Level detection. 1 = Edge detection.
[2]	**SPRRWEN**	**Spare Register Enable Bit** 0 = Spare register is Disabled. 1 = Spare register is Enabled. **Note:** When spare register is disabled, RTC_SPR0 ~ RTC_SPR19 cannot be accessed.
[1]	**SNPTYPE0**	**Snoop Detection Level**

		This bit controls TAMPER detect event is high level/rising edge or low level/falling edge. 0 = Low level/Falling edge detection. 1 = High level/Rising edge detection..
[0]	**SNPDEN**	**Snoop Detection Enable Bit** 0 = TAMPER pin detection is Disabled. 1 = TAMPER pin detection is Enabled.

RTC Spare Register (RTC_SPRx)

Register	Offset	R/W	Description	Reset Value
RTC_SPR0	RTC_BA+0x40	R/W	RTC Spare Register 0	0x0000_0000
RTC_SPR1	RTC_BA+0x44	R/W	RTC Spare Register 1	0x0000_0000
RTC_SPR2	RTC_BA+0x48	R/W	RTC Spare Register 2	0x0000_0000
RTC_SPR3	RTC_BA+0x4C	R/W	RTC Spare Register 3	0x0000_0000
RTC_SPR4	RTC_BA+0x50	R/W	RTC Spare Register 4	0x0000_0000
RTC_SPR5	RTC_BA+0x54	R/W	RTC Spare Register 5	0x0000_0000
RTC_SPR6	RTC_BA+0x58	R/W	RTC Spare Register 6	0x0000_0000
RTC_SPR7	RTC_BA+0x5C	R/W	RTC Spare Register 7	0x0000_0000
RTC_SPR8	RTC_BA+0x60	R/W	RTC Spare Register 8	0x0000_0000
RTC_SPR9	RTC_BA+0x64	R/W	RTC Spare Register 9	0x0000_0000
RTC_SPR10	RTC_BA+0x68	R/W	RTC Spare Register 10	0x0000_0000
RTC_SPR11	RTC_BA+0x6C	R/W	RTC Spare Register 11	0x0000_0000
RTC_SPR12	RTC_BA+0x70	R/W	RTC Spare Register 12	0x0000_0000
RTC_SPR13	RTC_BA+0x74	R/W	RTC Spare Register 13	0x0000_0000
RTC_SPR14	RTC_BA+0x78	R/W	RTC Spare Register 14	0x0000_0000
RTC_SPR15	RTC_BA+0x7C	R/W	RTC Spare Register 15	0x0000_0000
RTC_SPR16	RTC_BA+0x80	R/W	RTC Spare Register 16	0x0000_0000
RTC_SPR17	RTC_BA+0x84	R/W	RTC Spare Register 17	0x0000_0000
RTC_SPR18	RTC_BA+0x88	R/W	RTC Spare Register 18	0x0000_0000
RTC_SPR19	RTC_BA+0x8C	R/W	RTC Spare Register 19	0x0000_0000

31	30	29	28	27	26	25	24
SPARE							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SPARE							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SPARE							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SPARE							

Bits	Description	
[31:0]	**SPARE**	**Spare Register** This field is used to store back-up information defined by user. This field will be cleared by hardware automatically once a snooper pin event is detected. Before storing back-up information in to RTC_SPRx register, user should write 0xA965 to RTC_RWEN[15:0] to make sure register read/write enable bit REWNF (RTC_RWEN[16]) is enabled.

RTC 32K Oscillator Control Register (RTC_LXTCTL)

Register	Offset	R/W	Description	Reset Value
RTC_LXTCTL	RTC_BA+0x100	R/W	RTC 32.768 kHz Oscillator Control Register	0x0000_000E

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				GAIN			LXTEN

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3:1]	**GAIN**	**Oscillator Gain Option** User can select oscillator gain according to crystal external loading and operating temperature range. The larger gain value corresponding to stronger driving capability and higher power consumption. 000 = L0 mode. 001 = L1 mode. 010 = L2 mode. 011 = L3 mode. 100 = L4 mode. 101 = L5 mode. 110 = L6 mode. 111 = L7 mode (Default).
[0]	**LXTEN**	**Backup Domain 32K Oscillator Enable Bit** 0 = Oscillator is Disabled. 1 = Oscillator is Enabled. This bit controls 32 kHz oscillator on/off. User can set either LXTEN in RTC battery power domain or system manager control register CLK_PWRCTL[1] (LXTEN) to enable 32 kHz oscillator. If this bit is set 1, X32 kHz oscillator keep running after system core power is turned off, if this bit is clear to 0, oscillator is turned off when system core power is turned off.

X32KO Control Register (RTC_LXTOCTL)

Register	Offset	R/W	Description	Reset Value
RTC_LXTOCTL	RTC_BA+0x104	R/W	X32KO Pin Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				CTLSEL	DOUT	OPMODE	

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**CTLSEL**	**IO Pin State Backup Selection** When low speed 32 kHz oscillator is disabled, X32KO (PF.0) pin can be used as GPIO function. User can program CTLSEL bit to decide X32KO (PF.0) I/O function is controlled by system power domain GPIO module or V_{BAT} power domain RTC_LXTOCTL control register. 0 = X32KO (PF.0) pin I/O function is controlled by GPIO module. It becomes floating when system power is turned off. 1 = X32KO (PF.0) pin I/O function is controlled by V_{BAT} power domain, X32KO (PF.0) pin function and I/O status are controlled by OPMODE[1:0] and DOUT after CTLSEL it set to 1. I/O pin keeps the previous state after system power is turned off.
[2]	**DOUT**	**IO Output Data** 0 = X32KO (PF.0) output low. 1 = X32KO (PF.0) output high.
[1:0]	**OPMODE**	**GPF0 Operation Mode** 00 = X32KO (PF.0) is input only mode, without pull-up resistor. 01 = X32KO (PF.0) is output push pull mode. 10 = X32KO (PF.0) is open drain mode. 11 = X32KO (PF.0) is input only mode with internal pull up.

X32KI Control Register (RTC_LXTICTL)

Register	Offset	R/W	Description	Reset Value
RTC_LXTICTL	RTC_BA+0x108	R/W	X32KI Pin Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				CTLSEL	DOUT	OPMODE	

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**CTLSEL**	**IO Pin State Backup Selection** When low speed 32 kHz oscillator is disabled, X32KI (PF.1) pin can be used as GPIO function. User can program CTLSEL bit to decide X32KI (PF.1) I/O function is controlled by system power domain GPIO module or V_{BAT} power domain RTC_LXTICTL control register. 0 = X32KI (PF.1) pin I/O function is controlled by GPIO module. It becomes floating state when system power is turned off. 1 = X32KI (PF.1) pin I/O function is controlled by V_{BAT} power domain, X32KI (PF.1) pin function and I/O status are controlled by OPMODE[1:0] and DOUT after CTLSEL it set to 1. I/O pin keeps the previous state after system power is turned off.
[2]	**DOUT**	**IO Output Data** 0 = X32KI (PF.1) output low. 1 = X32KI (PF.1) output high.
[1:0]	**OPMODE**	**IO Operation Mode** 00 = X32KI (PF.1) is input only mode, without pull-up resistor. 01 = X32KI (PF.1) is output push pull mode. 10 = X32KI (PF.1) is open drain mode. 11 = X32KI (PF.1) is input only mode with internal pull up.

TAMPER Control Register (RTC_TAMPCTL)

Register	Offset	R/W	Description	Reset Value
RTC_TAMPCTL	RTC_BA+0x10C	R/W	TAMPER Pin Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				CTLSEL	DOUT	OPMODE	

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**CTLSEL**	**IO Pin State Backup Selection** When tamper function is disabled, TAMPER pin can be used as GPIO function. User can program CTLSEL bit to decide PF.2 I/O function is controlled by system power domain GPIO module or V_{BAT} power domain RTC_TAMPCTL control register. 0 =TAMPER (PF.2) I/O function is controlled by GPIO module. It becomes floating state when system power is turned off. 1 =TAMPER (PF.2) I/O function is controlled by V_{BAT} power domain. PF.2 function and I/O status are controlled by OPMODE[1:0] and DOUT after CTLSEL it set to 1. I/O pin state keeps previous state after system power is turned off.
[2]	**DOUT**	**IO Output Data** 0 = TAMPER (PF.2) output low. 1 = TAMPER (PF.2) output high.
[1:0]	**OPMODE**	**IO Operation Mode** 00 = TAMPER (PF.2) is input only mode, without pull-up resistor. 01 = TAMPER (PF.2) is output push pull mode. 10 = TAMPER (PF.2) is open drain mode. 11 = TAMPER (PF.2) is input only mode with internal pull up.

6.13 UART Interface Controller (UART)

6.13.1 Overview

The NuMicro® M4TK series provides four channels of Universal Asynchronous Receiver/Transmitters (UART). UART Controller performs Normal Speed UART and supports flow control function. The UART Controller performs a serial-to-parallel conversion on data received from the peripheral and a parallel-to-serial conversion on data transmitted from the CPU. Each UART Controller channel supports ten types of interrupts. The UART controller also supports IrDA SIR, RS-485 and auto-baud rate measuring function.

6.13.2 Features

- Full-duplex asynchronous communications
- Separates receive and transmit 16/16 bytes entry FIFO for data payloads
- Supports hardware auto-flow control
- Programmable receiver buffer trigger level
- Supports programmable baud rate generator for each channel individually
- Supports nCTS and RX data wake-up function
- Supports 8-bit receiver buffer time-out detection function
- Programmable transmitting data delay time between the last stop and the next start bit by setting DLY (UART_TOUT [15:8])
- Supports Auto-Baud Rate measurement
- Supports break error, frame error, parity error and receive/transmit buffer overflow detection function
- Fully programmable serial-interface characteristics
  - Programmable number of data bit, 5-, 6-, 7-, 8- bit character
  - Programmable parity bit, even, odd, no parity or stick parity bit generation and detection
  - Programmable stop bit, 1, 1.5, or 2 stop bit generation
- Supports IrDA SIR function mode
  - Supports for 3/16 bit duration for normal mode
- Supports LIN function mode (Only UART0 /UART1 with LIN function)
  - Supports LIN master/slave mode
  - Supports programmable break generation function for transmitter
  - Supports break detection function for receiver
- Supports RS-485 function mode
  - Supports RS-485 9-bit mode
  - Supports hardware or software enables to program nRTS pin to control RS-485 transmission direction

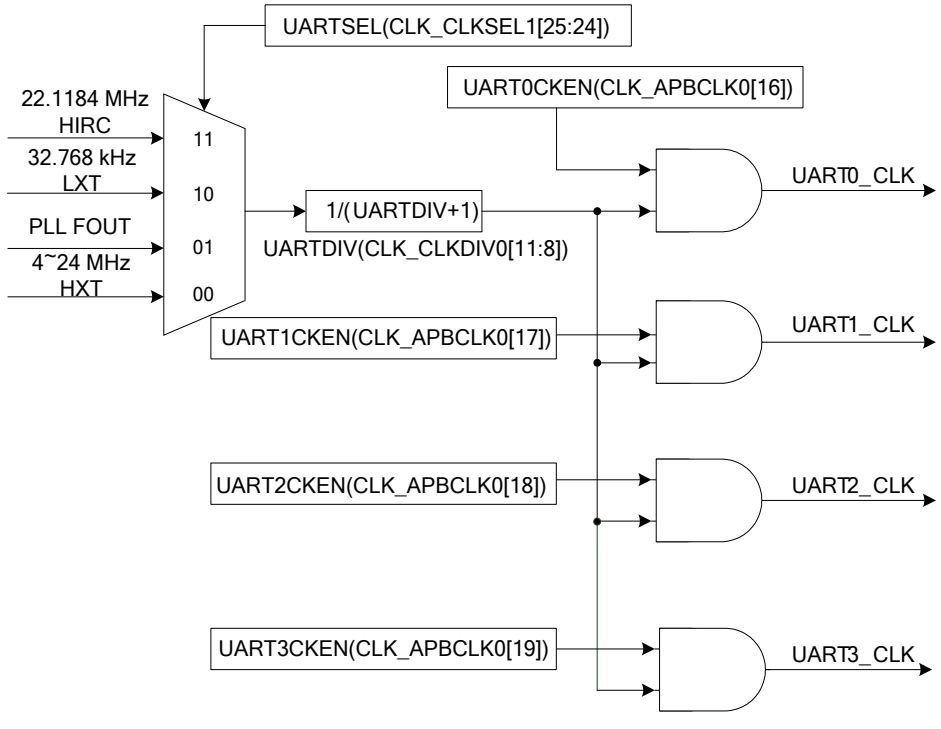
UART Feature	UART0 / UART1	UART2 / UART3	SC_UART
FIFO	16 Bytes	16 Bytes	4 Bytes

Auto Flow Control (CTS/RTS)	√	√	-
IrDA	√	√	-
LIN	√	-	-
RS-485 Function Mode	√	√	-
Auto-Flow Control	√	√	-
nCTS Wake-up	√	√	-
RX Data Wake-up	√	√	-
Auto-Baud Rate Measurement	√	√	-
STOP Bit Length	1, 1.5, 2 bit	1, 1.5, 2 bit	1, 2 bit
Word Length 5, 6,7, 8 bits	√	√	√
Even / Odd Parity	√	√	√
Stick Bit	√	√	-
√= Supported			

Table 6-20 NuMicro® M4TK Series UART Feature

6.13.3 Block Diagram

The UART clock control and block diagram are shown in Figure 6.13-1 and Figure 6.13-2 respectively.



Note: Before clock switching, both the pre-selected and newly selected clock sources must be turned on and stable.

Figure 6.13-1 UART Clock Control Diagram

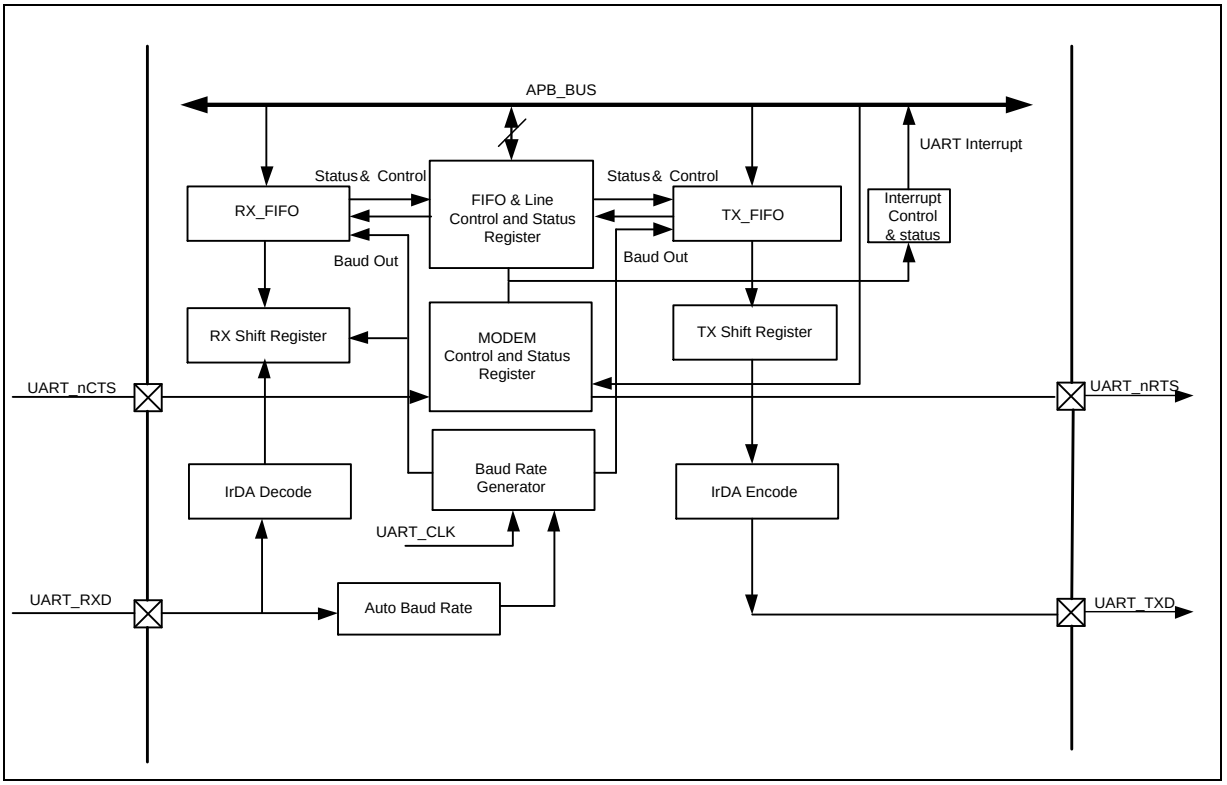


Figure 6.13-2 UART Block Diagram

Each block is described in detail as follows:

TX_FIFO
The transmitter is buffered with a 16 bytes FIFO to reduce the number of interrupts presented to the CPU.

RX_FIFO
The receiver is buffered with a 16 bytes FIFO (plus three error bits, BIF (UART_FIFOSTS[6]), FEF (UART_FIFOSTS[5]), PEF (UART_FIFOSTS[4])) to reduce the number of interrupts presented to the CPU.

TX Shift Register
This block is responsible for shifting out the transmitting data serially.

RX Shift Register
This block is responsible for shifting in the receiving data serially.

Modem Control and Status Register
This register controls the interface to the MODEM or data set (or a peripheral device emulating a MODEM).

Baud Rate Generator
Divide the external clock by the divisor to get the desired baud rate clock. Refer to baud rate equation.

IrDA Encode
This block is IrDA encoding control block.

IrDA Decode
This block is IrDA decoding control block.

FIFO & Line Control and Status Register

This field is register set that including the FIFO control registers (UART_FIFO), FIFO status registers (UART_FIFOSTS), and line control register (UART_LINE) for transmitter and receiver. The time-out control register (UART_TOUT) identifies the condition of time-out interrupt.

Auto-Baud Rate Measurement

This block is responsible for auto-baud rate measurement.

Interrupt Control and Status Register

There are ten types of interrupts, transmitter FIFO empty interrupt (THERIF), receiver threshold level reaching interrupt (RDAIF), receive line status interrupt (parity error or framing error or break interrupt) (RLSIF), time-out interrupt (RXTOINT), Buffer error interrupt (BUFERRINT), LIN bus interrupt (LININT), data wake-up interrupt, nCTS wake-up interrupt and auto-baud rate detection finish or auto-baud rate detection counter overflow interrupt. Interrupt enable register (UART_INTEN) enable or disable the responding interrupt and interrupt status register (UART_INTSTS) identifying the occurrence of the responding interrupt.

6.13.4 Basic Configuration

The UART Controller function pins are configured in SYS_GPA_MFPL, SYS_GPA_MFPH, SYS_GPB_MFPL, SYS_GPB_MFPH, SYS_GPC_MFPL, SYS_GPD_MFPL, SYS_GPD_MFPH and SYS_GPE_MFPH Multiple Function Registers.

The UART Controller clock are enabled in UAR0TCKEN (CLK_APBCLK0[16]) for UART0, UART1CKEN(CLK_APBCLK0[17]) for UART1, UART2CKEN(CLK_APBCLK0[18] for UART2 and UART3CKEN(CLK_APBCLK0[19] for UART3.

The UART Controller clock source is selected by UARTSEL (CLK_CLKSEL1[25:24]).

The UART Controller clock pre-scale is determined by UARTDIV (CLK_CLKDIV0[11:8]).

UART Interface Controller Pin description is shown as Table 6-21:

Pin	Type	Description
UART_TXD	Output	UART transmit
UART_RXD	Input	UART receive
UART_nCTS	Input	UART modem clear to send
UART_nRTS	Output	UART modem request to send

Table 6-21 UART Interface Controller Pin

6.13.5 Functional Description

The UART Controller supports four function modes including UART, IrDA, LIN and RS-485 mode. User can select a function by setting the UART_FUNCSEL register. The four function modes will be described in following section.

6.13.5.1 UART Controller Baud Rate Generator

The UART Controller includes a programmable baud rate generator capable of dividing clock input by divisors to produce the serial clock that transmitter and receiver need. The Table 6-22 list the UART baud rate equations in the various conditions and UART baud rate parameter settings. There is no error for the baud rate results calculated through the baud rate parameter and register setting below. In IrDA function mode, the baud rate generator must be set in mode 0. More detail register description is shown in UART_BAUD register. There are three setting mode. Mode 0 is set by UART_BAUD[29:28] with 00. Mode 1 is set by UART_BAUD[29:28] with 10. Mode 2 is set by UART_BAUD[29:28] with 11.

Mode	BAUDM1	BAUDM0	Baud Rate Equation
Mode 0	0	0	UART_CLK / [16 * (BRD+2)]
Mode 1	1	0	UART_CLK / [(EDIVM1+1) * (BRD+2)], EDIVM1 must >= 8
Mode 2	1	1	UART_CLK / (BRD+2) .If UART_CLK <= 3*HCLK, BRD must >= 9. If UART_CLK > 3*HCLK, BRD must >= 3*N – 1. N is the smallest integer larger than or equal to the ratio of UART_CLK /HCLK. For example, if 3*HCLK < UART_CLK =< 4*HCLK, BRD must >=11. if 4*HCLK < UART_CLK =< 5*HCLK, BRD must >=14.

Table 6-22 UART Controller Baud Rate Equation Table

UART Peripheral Clock = 22.1184 MHz			
Baud Rate	**Mode 0**	**Mode 1**	**Mode 2**
921600	Not support	BRD=0, EDIVM1=11	BRD=22
460800	BRD=1	BRD=1, EDIVM1 =15 BRD=2, EDIVM1 =11	BRD=46
230400	BRD =4	BRD =4, EDIVM1 =15 BRD =6, EDIVM1 =11	BRD =94
115200	BRD =10	BRD =10, EDIVM1 =15 BRD =14, EDIVM1 =11	BRD =190
57600	BRD =22	BRD =22, EDIVM1 =15 BRD =30, EDIVM1 =11	BRD =382
38400	BRD =34	BRD =62, EDIVM1 =8 BRD =46, EDIVM1 =11 BRD =34, E EDIVM1 =15	BRD =574
19200	BRD =70	BRD =126, EDIVM1 =8 BRD =94, EDIVM1 =11 BRD =70, EDIVM1 =15	BRD =1150
9600	BRD =142	BRD =254, EDIVM1 =8 BRD =190, EDIVM1 =11 BRD =142, EDIVM1 =15	BRD =2302
4800	BRD =286	BRD =510, EDIVM1 =8 BRD =382, EDIVM1 =11 BRD =286, EDIVM1 =15	BRD =4606

Table 6-23 UART Controller Baud Rate Parameter Setting Example Table

UART Peripheral Clock = 22.1184 MHz			
Baud Rate	**UART_BAUD Vaule**		
	Mode 0	**Mode 1**	**Mode 2**
921600	Not support	0x2B00_0000	0x3000_0016
460800	0x0000_0001	0x2F00_0001 0x2B00_0002	0x3000_002E
230400	0x0000_0004	0x2F00_0004 0x2B00_0006	0x3000_005E
115200	0x0000_000A	0x2F00_000A 0x2B00_000E	0x3000_00BE
57600	0x0000_0016	0x2F00_0016 0x2B00_001E	0x3000_017E
38400	0x0000_0022	0x2800_003E 0x2B00_002E 0x2F00_0022	0x3000_023E
19200	0x0000_0046	0x2800_007E 0x2B00_005E 0x2F00_0046	0x3000_047E

9600	0x0000_008E	0x2800_00FE 0x2B00_00BE 0x2F00_008E	0x3000_08FE
4800	0x0000_011E	0x2800_01FE 0x2B00_017E 0x2F00_011E	0x3000_11FE

Table 6-24 UART Controller Baud Rate Register Setting Example Table

6.13.5.2 UART Controller Auto-Baud Rate Function Mode

Auto-Baud Rate function can measure baud rate of receiving data from UART RX pin automatically. When the Auto-Baud Rate measurement is finished, the measuring baud rate is loaded to BRD (UART_BAUD[15:0]). Both of the BAUDM1 (UART_BAUD[29]) and BAUDM0 (UART_BAUD[28]) are set to 1 automatically. UART RX data from Start bit to 1st rising edge time is set by $2^{ABRDBITS}$ bit time in Auto-Baud Rate function detection frame.

$2^{ABRDBITS}$ bit time from Start bit to the 1st rising edge is calculated by setting ABRDBITS (UART_ALTCTL[20:19]) . Setting ABRDEN (UART_ALTCTL[18]) is to enable auto-baud rate function. In beginning stage, the UART RX is kept at 1. Once falling edge is detected, START bit is received. The auto-baud rate counter is reset and starts counting. The auto-baud rate counter will be stop when the 1st rising edge is detected. Then, auto-baud rate counter value divided by ABRDBITS (UART_ALTCTL[20:19]) is loaded to BRD(UART_BAUD[15:0]) automatically. ABRDEN (UART_ALTCTL[18]) is cleared. Once the auto-baud rate measurement is finished, the ABRDIF (UART_FIFOSTS[1]) is set. When auto-baud rate counter is overflow, ABRTOIF (UART_FIFOSTS[2]) is set. If the ABRIEN (UART_INTEN[18]) is enabled, ABRDIF(UART_FIFOSTS[1]) or ABRDTOIF (UART_FIFOSTS[2]) cause the auto-baud rate interrupt ABRIF(UART_ALTCTL[17]) is generated.

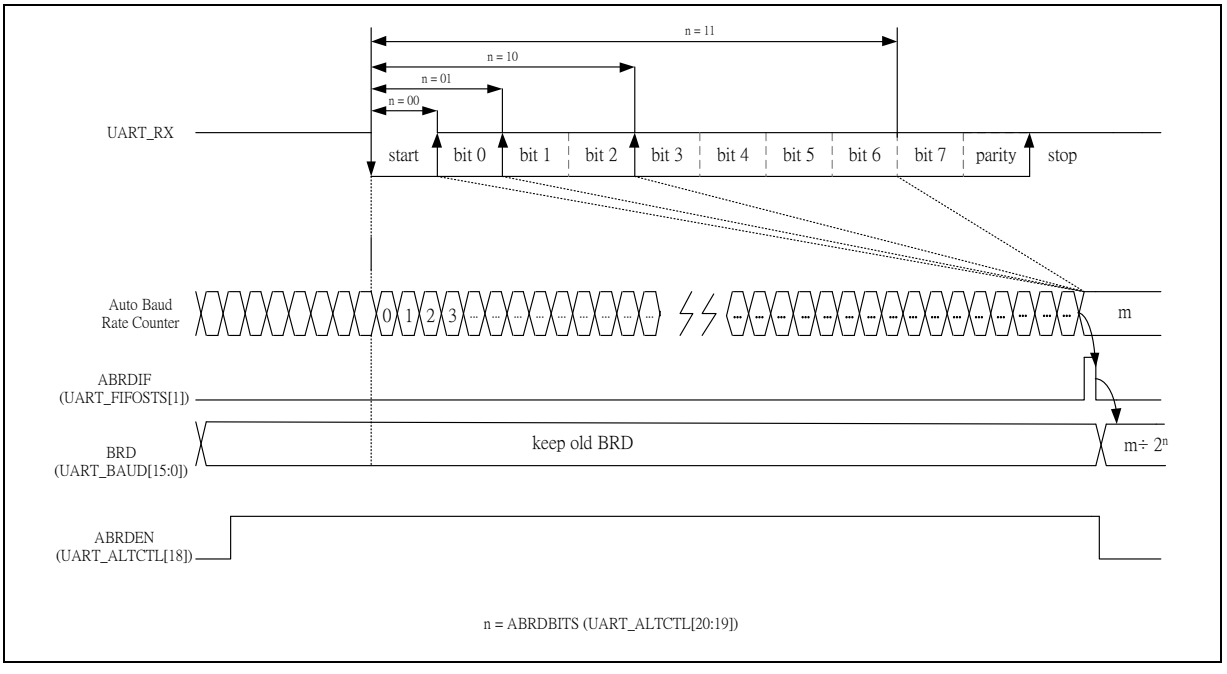


Figure 6.13-3 Auto-Baud Rate Measurement

Programming Sequence Example:

1. Program ABRDBITS (UART_ALTCTL[20:19]) to determines UART RX data 1st rising edge time from Start by 2 ABRDBITS bit time.
2. Set ABRIEN (UART_INTEN[18]) to enable auto-baud rate function interrupt.
3. Set ABRDEN (UART_ALTCTL[18]) to enable auto-baud rate function.
4. ABRDIF (UART_FIFOSTS[1]) is set, the auto-baud rate measurement is finished.
5. Operate UART transmit and receive action.
6. ABRDTOIF (UART_FIFOSTS[2]) is set, if auto-baud rate counter is overflow.
7. Go to Step 2.

6.13.5.3 UART Controller Transmit Delay Time Value

The UART Controller programs DLY (UART_TOUT [15:8]) to control the transfer delay time between the last stop bit and next start bit in transmission. The unit is baud. The operation is shown in Figure 6.13-4

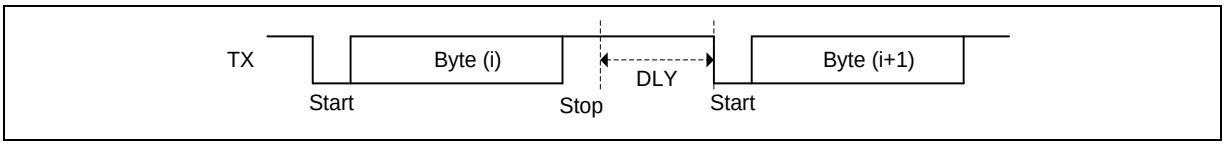


Figure 6.13-4 Transmit Delay Time Operation

6.13.5.4 UART Controller FIFO Control and Status

The UART Controller is built-in with a 16 bytes transmitter FIFO (TX_FIFO) and a 16 bytes receiver FIFO (RX_FIFO) that reduces the number of interrupts presented to the CPU. The CPU can read the status of the UART at any time during operation. The reported status information includes condition of the transfer operations being performed by the UART, as well as 3 error conditions (parity error, framing error, break interrupt) occur if receiving data has parity, frame or break error. UART, IrDA, LIN and RS-485 mode support FIFO control and status function.

6.13.5.5 UART Controller Wake-up Function

The UART controller supports wake-up system function. The wake-up function includes nCTS and data wake-up function. When the system is in power-down, the UART can wake-up system by nCTS pin or incoming data. When incoming data wakes system up, the incoming data will be received and stored in FIFO and controller will clear the WKDATIEN (UART_INTEN [10]) automatically. However, the first byte of receiving data is lost. The data is received after second bytes. The Figure 6.13-5 demonstrates the wake-up function.

nCTS Wake-Up Case 1 (nCTS transition from low to high)

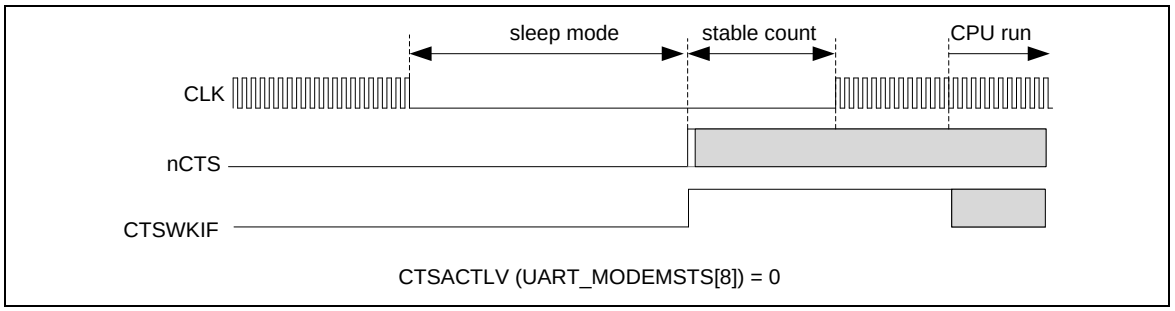


Figure 6.13-5 UART nCTS Wake-UP Case1

nCTS Wake-Up Case 2 (nCTS transition from high to low)

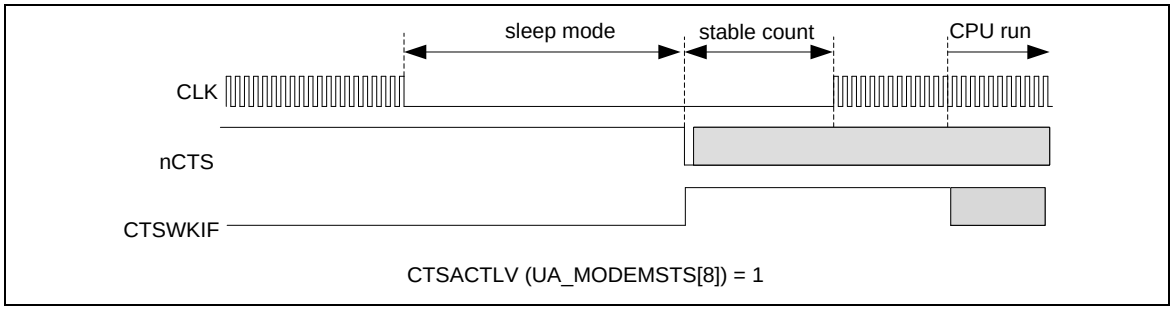


Figure 6.13-6 UART nCTS Wake-UP Case2

RX Data Wake-Up

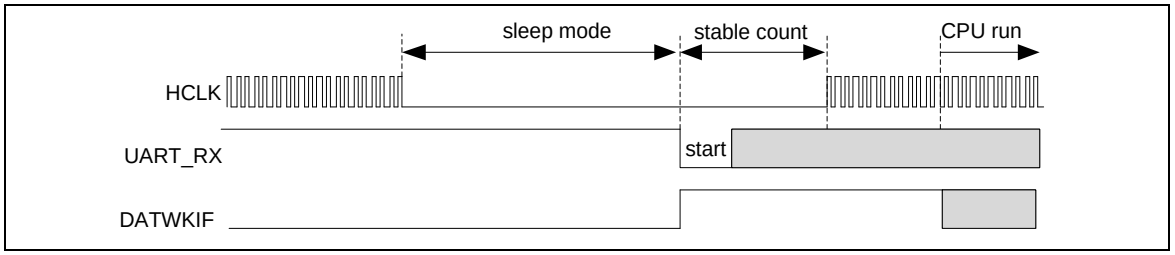


Figure 6.13-7 UART RX Data Wake-Up

6.13.5.6 UART Controller Interrupt and Status

Each UART Controller supports ten types of interrupts including:

- Receiver Data Available Interrupt (RDAINT)
- Transmit Holding Register Empty Interrupt (THERINT)
- Receive Line status Interrupt (parity error, frame error or break error) (RLSINT)
- MODEM Status Interrupt (MODEMINT)
- Receiver Buffer Time-out Interrupt (RXTOINT)
- Buffer Error Interrupt (BUFERRINT)
- LIN Bus Interrupt (LININT)
- nCTS Wake-up Interrupt (CTSWKIF)
- Data Wake-Up Interrupt (DATWKIF)
- Auto-Baud Rate Interrupt (ABRIF)

The Table 6-25 describe the interrupt sources and flags. The interrupt is generated when the interrupt flag is generated and the interrupt enable bit is set. User must clear the interrupt flag after the interrupt is generated.

Interrupt Source	Interrupt Indicator	Interrupt Enable Bit	Interrupt Flag	Flag Cleared By
Receive Data Available Interrupt	RDAINT	RDAIEN	**RDAIF**	Read UART_DAT
Transmit Holding Register Empty Interrupt	THERINT	TJREIEN	**THREIF**	Write UART_DAT
Receive Line Status Interrupt	RLSINT	RLSIEN	**RLSIF** = BIF	**Write '1' to** BIF
			RLSIF = FEF	**Write '1' to** FEF
			RLSIF = PEF	**Write '1' to** PEF
			RLSIF = ADDRDETF	**Write '1' to** ADDRDETF
Modem Status Interrupt	MODEMINT	MODEMIEN	**MODEMIF** = CTSDETF	**Write '1' to** CTSDETF
Receiver Buffer Time-out Interrupt	RXTOINT	RXTOIEN	**RXTOIF**	Read UART_DAT
Buffer Error Interrupt	BUFERRINT	BUFERRIEN	**BUFERRIF** = TXOVIF	**Write '1' to** TXOVIF
			BUFERRIF = RXOVIF	**Write '1' to** RXOVIF
LIN Bus Interrupt	LININT	LIN _IEN	**LINIF =** BRKDETF	**Write '1' to** LINIF **and** **Write '1' to** BRKDETF
			LINIF = BITEF	**Write '1' to** BITEF
			LINIF = SLVIDPEF	**Writing '1' to** SLVIDPEF
			LINIF = SLVHEF	**Wrie '1' to** SLVHEF
			LINIF = SLVHDETF	**Write '1' to** SLVHDETF
nCTS Wake-Up Interrupt	N/A	WKCTSIEN	**CTSWKIF**	**Write '1' to** CTSWKIF
Data Wake-Up Interrupt	N/A	WKDATIEN	**DATWKIF**	**Write '1' to** DATWKIF
Auto-Baud Rate Interrupt	N/A	ABRIEN	**ABRIF =** ABRDIF	**Write '1' to** ABRDIF
			ABRIF = ABRDTOIF	**Write '1' to** ABRDTOIF.

Table 6-25 UART Controller Interrupt Source and Flag List

6.13.5.7 UART Function Mode

The UART Controller provides UART function (Setting FUNCSEL (UART_FUNCSEL [1:0]) to '00' to enable UART function mode). The UART baud rate is up to 1 Mbps.

The UART provides full-duplex and asynchronous communications. The transmitter and receiver contain 16 bytes FIFO for payloads. User can program receiver buffer trigger level and receiver buffer time-out detection for receiver. The transmitting data delay time between the last stop and the next start bit can be programed by setting DLY (UART_TOUT [15:8]) register. The UART supports hardware auto-flow control that provides programmable nRTS flow control trigger level. The number of data bytes in RX FIFO is equal to or greater than RTSTRGLV (UART_FIFO[19:16]), the nRTS is de-asserted.

UART Line Control Function

The UART Controller supports fully programmable serial-interface characteristics by setting the UART_LINE register. User can program UART_LINE register for the word length, stop bit and parity bit setting. The Table 6-26 list the UART word, stop bit length and the parity bit settings.

NSB (UART_LINE[2])	WLS (UART_LINE[1:0])	Word Length (Bit)	Stop Length (Bit)
0	00	5	1
0	01	6	1
0	10	7	1
0	11	8	1
1	00	5	1.5
1	01	6	2
1	10	7	2
1	11	8	2

Table 6-26 UART Line Control of Word and Stop Length Setting

Parity Type	SPE (UART_LINE[5])	EPE (UART_LINE[4])	PBE (UART_LINE[3])	Description
No Parity	x	x	0	No parity bit output.
Odd Parity	0	0	1	Odd Parity is calculated by adding all the "1's" in a data stream and adding a parity bit to the total bits, to make the total count an odd number.
Even Parity	0	1	1	Even Parity is calculated by adding all the "1's" in a data stream and adding a parity bit to the total bits, to make the count an even number.
Forced Mask Parity	1	0	1	Parity bit always logic 1. Parity bit on the serial byte is set to "1" regardless of total number of "1's" (even or odd counts).
Forced Space Parity	1	1	1	Parity bit always logic 0. Parity bit on the serial byte is set to "0" regardless of total number of "1's" (even or odd counts).

Table 6-27 UART Line Control of Parity Bit Setting

UART Auto-Flow Control Function

The UART supports auto-flow control function that uses two signals, nCTS (clear-to-send) and nRTS (request-to-send), to control the flow of data transfer between the UART and external devices (e.g. Modem). When auto-flow is enabled, the UART is not allowed to receive data until the UART asserts nRTS to external device. When the number of bytes stored in the RX FIFO equals the value of RTSTRGLV (UART_FIFO [19:16]), the nRTS is de-asserted. The UART sends data out when UART detects nCTS is asserted from external device. If the valid asserted nCTS is not detected, the UART will not send data out.

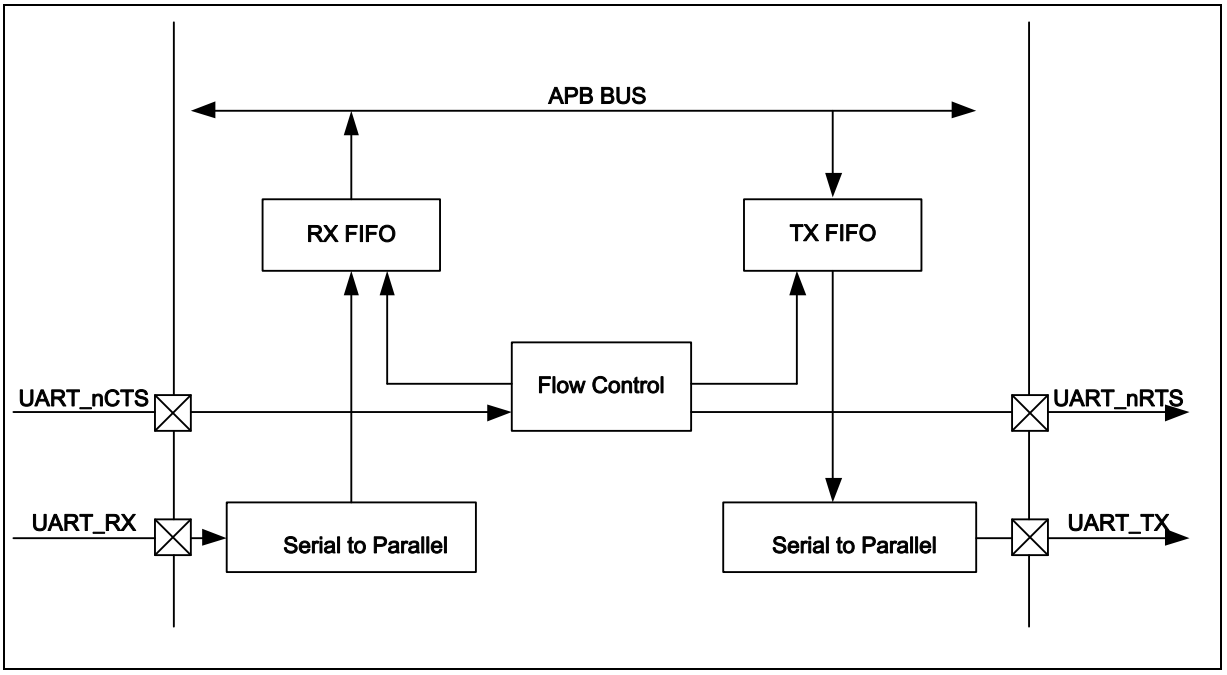


Figure 6.13-8 Auto-Flow Control Block Diagram

The Figure 6.13-9 demonstrates the nCTS auto-flow control of UART function mode. User must set ATOCTSEN (UART_INTEN [13]) to enable nCTS auto-flow control function. The CTSACTLV (UART_MODEMSTS [8]) can set nCTS pin input active state. The CTSDETF (UART_MODEMSTS[0]) is set when any state change of nCTS pin input has occurred, and then TX data will be automatically transmitted from TX FIFO.

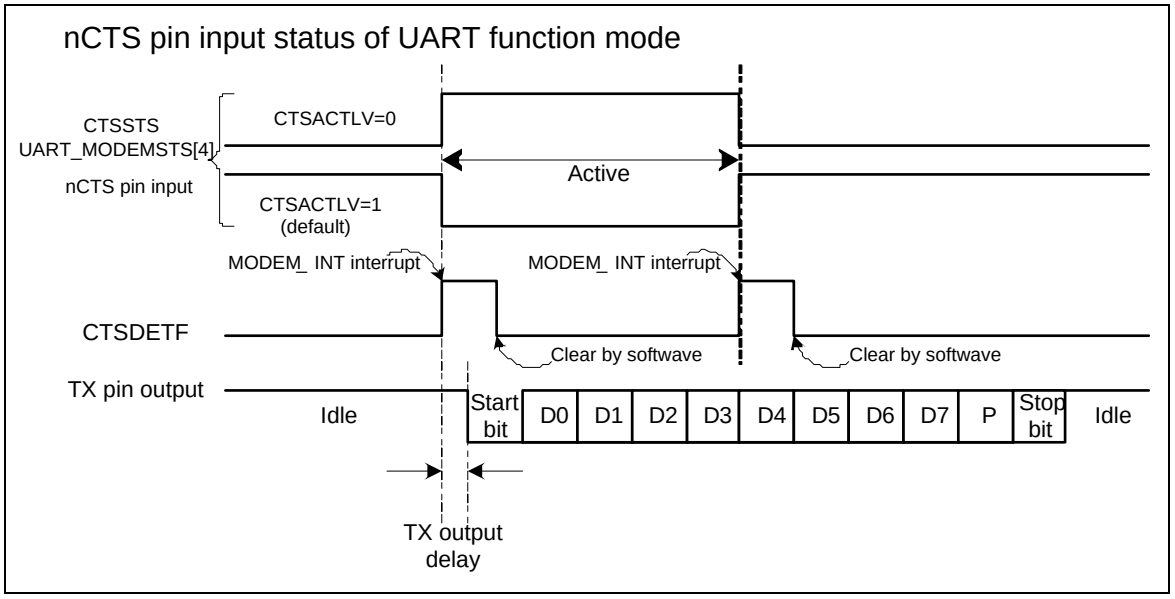


Figure 6.13-9 UART nCTS Auto-Flow Control Enabled

The Figure 6.13-10 demonstrates the nRTS auto-flow control of UART function mode. User must set ATORTSEN (UART_INTEN[12])=1 to enable nRTS auto-flow control function. The nRTS internal signal is controlled by RTSTRGLV (UART_FIFO[19:16]) trigger level, if the number of data bytes in RX FIFO is equal to or greater than RTSTRGLV (UART_FIFO[19:16]), the nRTS is de-asserted. Setting RTSACTLV(UART_MODEM[9]) can control the nRTS pin output is inverse or non-inverse from nRTS signal. User can read the RTSSTS (UART_MODEM[13]) bit to get real nRTS pin output voltage logic status.

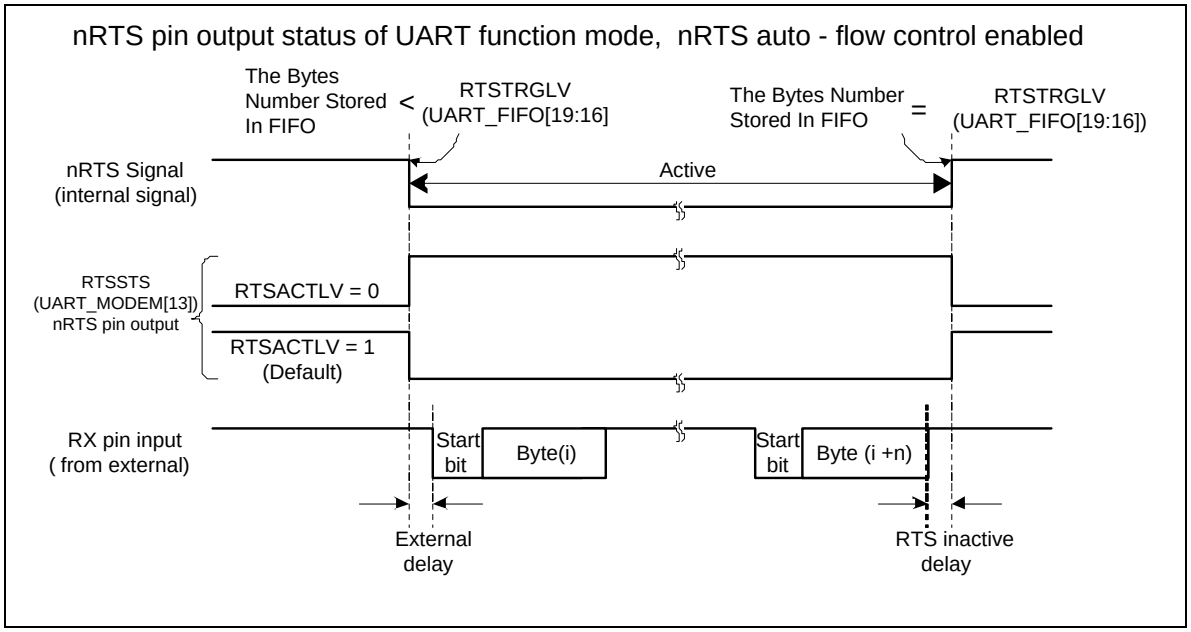


Figure 6.13-10 UART nRTS Auto-Flow Control Enabled

As shown in the Figure 6.13-11, in software mode (ATORTSEN(UART_INTEN[12])=0), the nRTS

flow is directly controlled by software programming of RTS(UART_MODEM[1]) control bit.

Setting RTSACTLV(UART_MODEM[9]) can control the nRTS pin output is inverse or non-inverse from RTS(UART_MODEM[1]) control bit. User can read the RTSSTS(UART_MODEM[13]) bit to get real nRTS pin output voltage logic status.

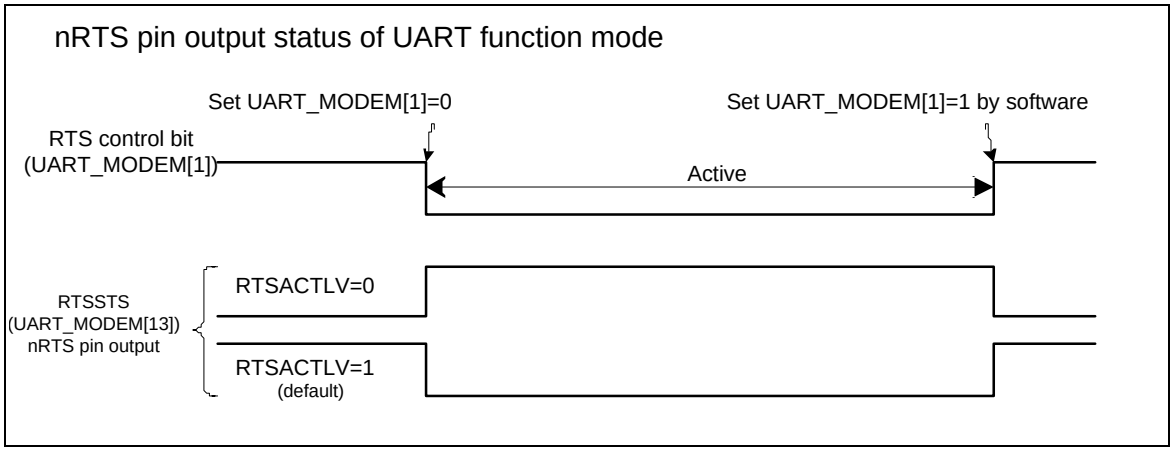


Figure 6.13-11 UART nRTS Auto-Flow with Software Control

6.13.5.8 IrDA Function Mode

The UART Controller also provides Serial IrDA (SIR, Serial Infrared) function (Setting UART_FUNCSEL [1:0] to '10' to enable the IrDA function). The SIR specification defines a short-range infrared asynchronous serial transmission mode with one start bit, 8 data bits, and 1 stop bit. The maximum data rate is 115.2 kbps. The IrDA SIR block contains an IrDA SIR protocol encoder/decoder. The IrDA SIR protocol is half-duplex only. So, it cannot transmit and receive data at the same time. The IrDA SIR physical layer specifies a minimum 10 ms transfer delay between transmission and reception, and this delay feature must be implemented by software.

In IrDA mode, the BAUDM1 (UART_BAUD [29]) must be cleared.

Baud Rate = Clock / (16 * BRD +2), where BRD (UART_BAUD[15:0]) is Baud Rate Divider in UART_BAUD register.

The IrDA control block diagram is shown as Figure 6.13-12.

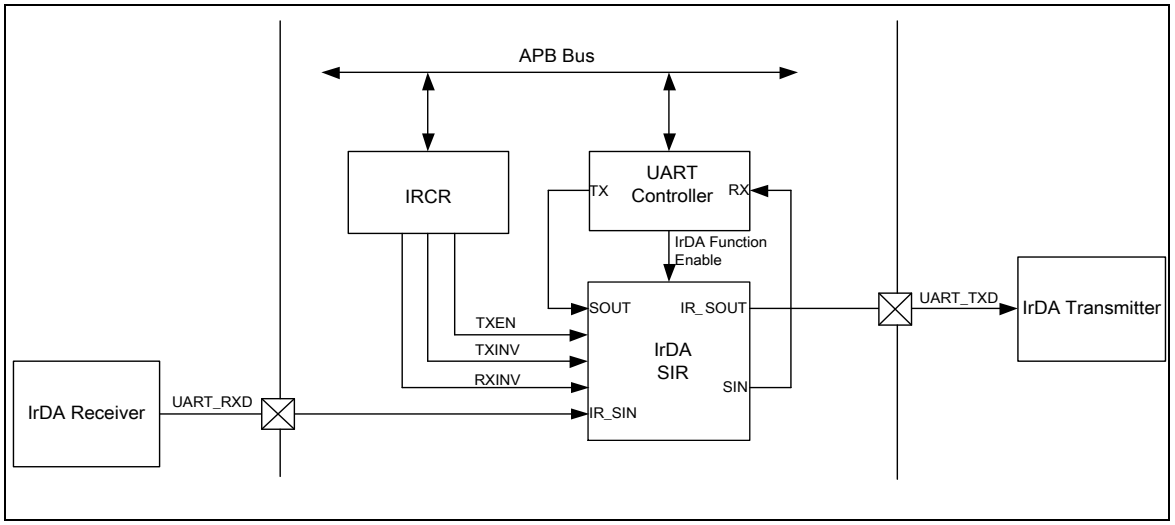


Figure 6.13-12 IrDA Control Block Diagram

IrDA SIR Transmit Encoder

The IrDA SIR Transmit Encoder modulates Non-Return-to-Zero (NRZ) transmit bit stream output from UART. The IrDA SIR physical layer specifies the use of Return-to-Zero, Inverted (RZI) modulation scheme which represents logic 0 as an infra light pulse. The modulated output pulse stream is transmitted to an external output driver and infrared light emitting diode.

The transmitted pulse width is specified as 3/16 period of baud rate.

IrDA SIR Receive Decoder

The IrDA SIR Receive Decoder demodulates the Return-to-Zero bit stream from the input detector and outputs the NRZ serial bits stream to the UART received data input.

A start bit is detected when the decoder input is LOW. In idle state, the decoder input is high. In normal operation, the RXINV (UART_IRDA[6]) is set to '1' and TXINV (UART_IRDA[5]) is set to '0'.

IrDA SIR Operation

The IrDA SIR encoder/decoder provides functionality which converts between UART data stream and half-duplex serial SIR interface. The Figure 6.13-13 is IrDA encoder/decoder waveform.

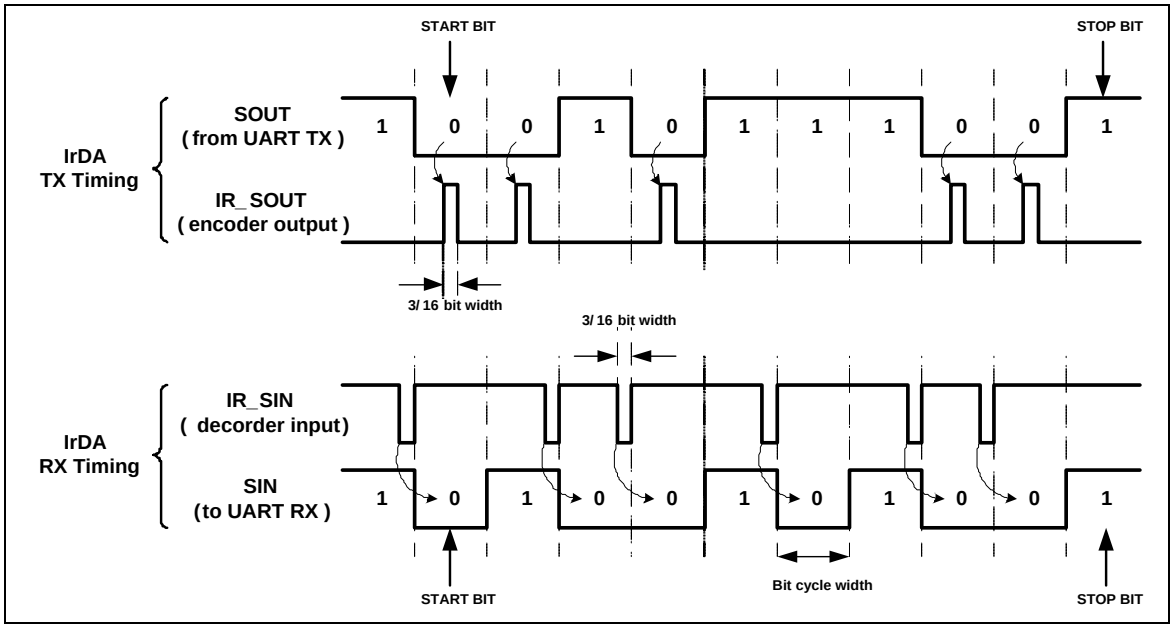


Figure 6.13-13 IrDA TX/RX Timing Diagram

6.13.5.9 LIN Function Mode (Local Interconnection Network)

The UART0/UART1 supports LIN function. Setting FUNCSEL (UART_FUNCSEL[1:0]) to '01' to select LIN mode operation. The UART0/UART1 supports LIN break/delimiter generation and break/delimiter detection in LIN master mode, and supports header detection and automatic resynchronization in LIN Slave mode.

Structure of LIN Frame

According to the LIN protocol, all information transmitted is packed as frames; a frame consists of a header (provided by the master task) and a response (provided by a slave task). The header (provided by the master task) consists of a break field and a sync field followed by a frame identifier (frame ID). The frame identifier uniquely defines the purpose of the frame. The slave task is appointed for providing the response associated with the frame ID. The response consists of a data field and a checksum field. The Figure 6.13-14 is the structure of LIN Frame.

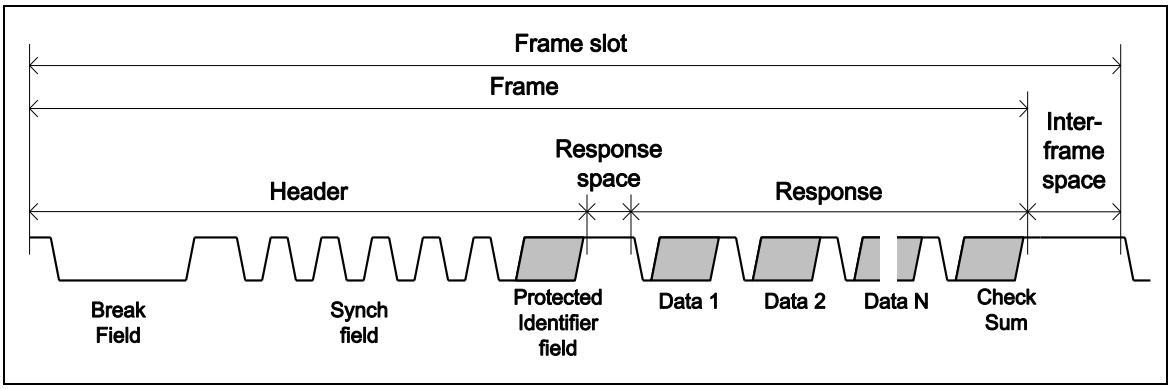


Figure 6.13-14 Structure of LIN Frame

Structure of LIN Byte

In LIN mode, each byte field is initiated by a START bit with value 0 (dominant), followed by 8 data bits and no parity bit, LSB is first and ended by 1 stop bit with value 1 (recessive) in accordance with the LIN standard. The structure of Byte is shown as Figure 6.13-15.

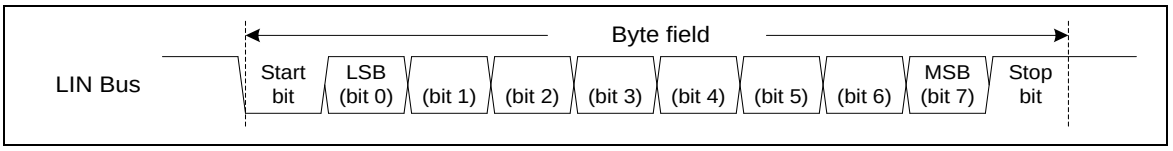


Figure 6.13-15 Structure of LIN Byte

LIN Master Mode

The UART0/UART1 controller supports LIN Master mode. To enable and initialize the LIN Master mode, the following steps are necessary:

1. Setting the UART_BAUD register to select the desired baud rate.
2. Setting WLS (UART_LINE[1:0]) to '11' to configure the word length with 8 bits, clearing PBE (UART_LINE[3]) bit to disable parity check and clearing NSB (UART_LINE[2]) bit to configure with one stop bit.
3. Setting FUNCSEL (UART_FUNCSEL[1:0]) to '01' to select LIN function mode operation.

A complete header consists of a break field and sync field followed by a frame identifier (frame ID). The UART0/UART1 controller can be selected header sending by three header selected modes. The header selected mode can be "break field" or "break field and sync field" or "break field, sync field and frame ID field" by setting HSEL (UART_LINCTL[23:22]). If the selected header is "break field", software must handle the following sequence to send a complete header to bus by filling sync data (0x55) and frame ID data to the UART_DAT register. If the selected header is "break field and sync field", software must handle the sequence to send a complete header to bus by filling the frame ID data to UART_DAT register, and if the selected header is "break field, sync field and frame ID field", hardware will control the header sending sequence automatically but software must filled frame ID data to PID (UART_LINCTL [31:24]). When operating in header selected mode in which the selected header is "break field, sync field and frame ID field", the frame ID parity bit can be calculated by software or hardware depending whether the IDPEN (UART_LINCTL[9]) bit is set or not.

HSEL	Break Field	Sync Field	ID Field
0	Generated by Hardware	Handled by Software	Handled by Software
1	Generated by Hardware	Generated by Hardware	Handled by Software
2	Generated by Hardware	Generated by Hardware	Generated by Hardware (But Software needs to fill ID to PID (UART_LINCTL[31:24]) first

Table 6-28 LIN Header Selection in Master Mode

When UART is operated in LIN data transmission, LIN bus transfer state can be monitored by hardware or software. User can enable hardware monitoring by setting BITERREN (UART_LINCTL [12]) to “1”, if the input pin (UART_RX) state is not equal to the output pin (UART_TX) state in LIN transmitter state that hardware will generate an interrupt to CPU. Software can also monitor the LIN bus transfer state by checking the read back data in UART_DAT register. The following sequence is a program sequence example.

The procedure without software error monitoring in Master mode:

1. Fill Protected Identifier to PID (UART_LINCTL[31:24]).
2. Select the hardware transmission header field including “break field + sync field + protected identifier field” by setting HSEL (UART_LINCTL [23:22]) to “10”.
3. Set SENDH (UART_LINCTL[8]) bit to 1 for requesting header transmission.
4. Wait until SENDH (UART_LINCTL[8]) bit cleared by hardware.
5. Wait until TXEMPTYF (UART_FIFOSTS[28]) set to 1 by hardware.

Note1: The default setting of break field is 12 dominant bits (break field) and 1 recessive bit break/sync delimiter. Setting BRKFL (UART_LINCTL [19:16]) and BSL (UART_LINCTL[21:20]) to change the LIN break field length and break/sync delimiter length.

Note2: The default setting of break/sync delimiter length is 1-bit time and the inter-byte spaces default setting is also 1-bit time. Setting BSL (UART_LINCTL[21:20]) and DLY(UART_TOUT[15:8]) can change break/sync delimiter length and inter-byte spaces.

Note3: If the header includes the “break field, sync field and frame ID field”, software must fill frame ID to PID (UART_LINCTL[31:24]) before trigger header transmission (setting the SENDH (UART_LINCTL[8]). The frame ID parity can be generated by software or hardware depending on IDPEN (UART_LINCTL[9]) setting. If the parity generated by software with IDPEN (UART_LINCTL[9]) is set to ‘0’, software must fill 8 bit data (include 2 bit parity) in this field. If the parity generated by hardware with IDPEN (UART_LINCTL[9]) is set to ‘1’ , software fill ID0~ID5 and hardware calculates P0 and P1.

Procedure with software error monitoring in Master mode:

1. Choose the hardware transmission header field to only include “break field” by setting HSEL (UART_LINCTL [23:22])] to ‘00’.
2. Enable break detection function by setting BRKDETEN (UART_LINCTL[10]).
3. Request break + break/sync delimiter transmission by setting the SENDH (UART_LINCTL[8])
4. Wait until the BRKDETF (UART_LINSTS[8]) flag is set to “1” by hardware..
5. Request sync field transmission by writing 0x55 into UART_DAT register.
6. Wait until the RDAIF (UART_INTSTS[0]) is set to “1” by hardware and then read back the

UART_DAT register.

7. Request header frame ID transmission by writing the protected identifier value to UART_DAT register.
8. Wait until the RDAIF (UART_INTSTS[0]) is set to “1” by hardware and then read back the UART_DAT register.

LIN break and delimiter detection

When software enables the break detection function by setting BRKDETEN (UART_LINCTL[10]), the break detection circuit is activated. The break detection circuit is totally independent from the UART0/UART1 receiver.

When the break detection function is enabled, the circuit looks at the input UART_RX pin for a start signal. If UART LIN controller detects consecutive dominant is greater than 11 bits dominant followed by a recessive bit (delimiter), the BRKDETF (UART_LINSTS[8]) flag is set at the end of break field. If the LINIEN (UART_INTEN[8]) bit is set to 1, an interrupt LININT (UART_INTSTS[15]) will be generated. The behavior of the break detection and break flag are shown in the Figure 6.13-16.

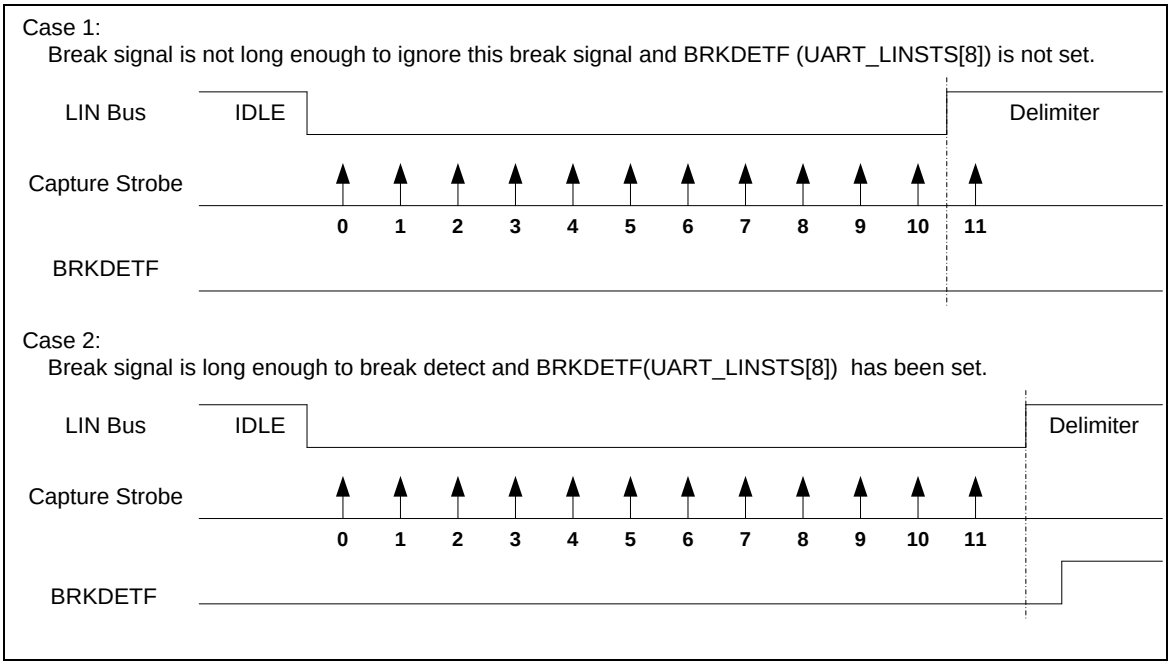


Figure 6.13-16 Break Detection in LIN Mode

LIN Frame ID and Parity Format

The LIN frame ID value in LIN function mode is shown, the frame ID parity can be generated by software or hardware depends on IDPEN (UART_LINCTL[9]) = 1.

If the parity generated by hardware, user fill ID0~ID5, (UART_LINCTL [29:24])hardware will calculate P0 (UART_LINCTL[30]) and P1 (UART_LINCTL[31]), otherwise user must filled frame ID and parity in this field.

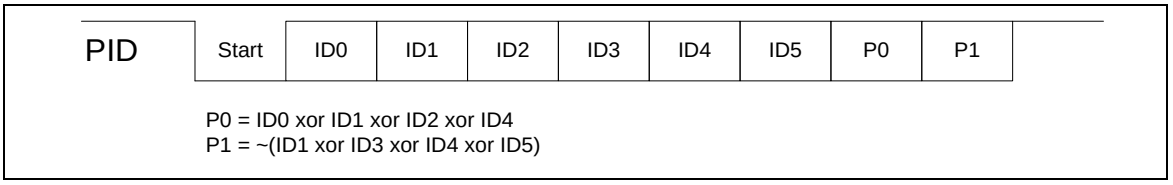


Figure 6.13-17 LIN Frame ID and Parity Format

LIN Slave Mode

The UART0/UART1 controller supports LIN Slave mode. To enable and initialize the LIN Slave mode, the following steps are necessary:

1. Set the UART_BAUD register to select the desired baud rate
2. Configure the data length to 8 bits by setting WLS (UART_LINE[1:0]) to '11' and disable parity check by clearing PBE (UART_LINE[3]) bit and configure with one stop bit by clearing NSB (UART_LINE[2]) bit.
3. Select LIN function mode by setting FUNCSEL (UART_FUNCSEL[1:0]) to '01'
4. Enable LIN slave mode by setting the SLVEN (UART_LINCTL[0]) to 1.

LIN header reception

According to the LIN protocol, a slave node must wait for a valid header which comes from the master node. Next the slave task will take one of following actions (depend on the master header frame ID value)

- Receive the response.
- Transmit the response.
- Ignore the response and wait for next header.

In LIN Slave mode, user can enable the slave header detection function by setting the SLVHDEN (UART_LINCTL[10]) to detect complete frame header (receive "break field", "sync field" and "frame ID field"). When a LIN header is received, the SLVHDETF (UART_LINSTS[0]) flag will be set. If the LINIEN (UART_INTEN[8]) bit is set to 1, an interrupt will be generated. User can enable the frame ID parity check function by setting IDPEN (UART_LINCTL[9]). If only received frame ID parity is not correct (break and sync filed are correct), the SLVIDPEF (UART_LINSTS[2]) flag is set to '1'. If the LINIEN(UART_INTEN[8]) is set to 1, an interrupt will be generated and SLVHDETF (UART_LINSTS[0]) is set to '1'. User can also put LIN in mute mode by setting MUTE (UART_LINCTL[4]) to '1'. This mode allows detection of headers only (break + sync + frame ID) and prevents the reception of any other characters. In order to avoid bit rate tolerance, the controller supports automatic resynchronization function to avoid clock deviation error, user can enable this feature by setting SLVAREN (UART_LINCTL[2]).

LIN response transmission

The LIN slave node can transmit response and receive response. When slave node is the publisher of the response, the slave node sends response by filling data to the UART_DAT register. If the slave node is the subscriber of the response, the slave node receives data from LIN bus.

LIN header time-out error

The LIN slave controller contains a header time-out counter. If the entire header is not received within the maximum time limit of 57 bit times, the header error flag SLVHEF (UART_LINSTS [1]) will be set. The time-out counter is enabled at each break detect edge and stopped in the following conditions.

- A LIN frame ID field has been received.
- The header error flag asserts.
- Writing 1 to the SLVSYNCF (UART_LINSTS[3]) to re-search a new frame header.

Mute mode and LIN exit from mute mode condition

In Mute mode, a LIN slave node will not receive any data until specified condition occurred. It allows header detection only and prevents the reception of any other characters. User can enable Mute mode by setting the MUTE (UART_LINCTL[4]) and exiting from Mute mode condition can be selected by HSEL (UART_LINCTL[23:22]).

Note: It is recommended to set LIN slave node to Mute mode after checksum transmission.

The LIN slave controller exiting from Mute mode is described as follows: If HSEL (UART_LINCTL[23:22]) is set to “break field”, when LIN slave controller detects a valid LIN break + delimiter, the controller will enable the receiver (exit from Mute mode) and subsequent data (sync data, frame ID data, response data) are received in RX-FIFO.

If HSEL (UART_LINCTL[23:22]) is set to “break field and sync field”, when the LIN slave controller detects a valid LIN break + delimiter followed by a valid sync field without frame error, the controller will enable the receiver (exit from mute mode) and subsequent data(ID data, response data) are received in RX-FIFO. If HSEL (UART_LINCTL[23:22]) is set to “break field, sync field and ID field”, when the LIN slave controller detects a valid LIN break + delimiter and valid sync field without frame error followed by ID data without frame error and received ID data matched PID (UART_LINCTL[31:24]) value. The controller will enable the receiver (exit from mute mode) and subsequent data (response data) are received in RX-FIFO.

Slave mode non-automatic resynchronization (NAR)

User can disable the automatic resynchronization function to fix the communication baud rate. When operating in Non-Automatic Resynchronization mode, software needs some initial process, and the initialization process flow of Non-Automatic Resynchronization mode is shown as follows:

1. Select the desired baud rate by setting the UART_BAUD register.
2. Select LIN function mode by setting FUNCSEL (UART_FUNCSEL[1:0]) to ‘01’.
3. Disable automatic resynchronization function by setting SLVAREN (UART_LINCTL[2]) is set to 0.
4. Enable LIN slave mode by setting the SLVEN (UART_LINCTL[0]) is set to 1.

Slave mode with automatic resynchronization (AR)

In Automatic Resynchronization (AR) mode, the controller will adjust the baud rate generator after each sync field reception. The initialization process flow of Automatic Resynchronization mode is shown as follows:

1. Select the desired baud rate by setting the UART_BAUD register.
2. Select LIN function mode by setting UART_FUNCSEL (UART_FUNCSEL[1:0]) to ‘01’
3. Enable automatic resynchronization function by setting SLVAREN (UART_LINCTL[2]) to ‘1’.

4. Enable LIN slave mode by setting the SLVEN (UART_LINCTL[0]) is set to '1'.

When the automatic resynchronization function is enabled, after each LIN break field, the time duration between five falling edges is sampled on peripheral clock and the result of this measurement is stored in an internal 13-bit register and the UART_BAUD register value will be automatically updated at the end of the fifth falling edge. If the measure timer (13-bit) overflows before five falling edges, then the header error flag SLVHEF (UART_LINSTS [1]) will be set.

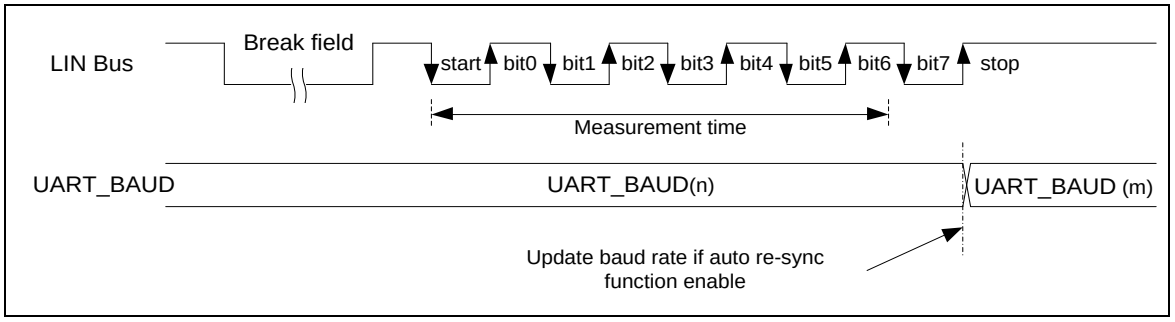


Figure 6.13-18 LIN Sync Field Measurement

When operating in Automatic Resynchronization (AR) mode, software must select the desired baud rate by setting the UART_BAUD register and hardware will store it at internal TEMP_REG register, after each LIN break field, the time duration between five falling edges is sampled on peripheral clock and the result of this measurement is stored in an internal 13-bit register BAUD_LIN and the result will be updated to UART_BAUD register automatically.

To guarantee the transmission baud rate, the baud rate generator must reload the initial value before each new break reception. The initial value is programmed by the application during initialization (TEMP_REG). User can set SLVDUEN (UART_LINCTL [3]) to enable auto reload initial baud rate value function. If the SLVDUEN (UART_LINCTL [3]) is set, when received the next character, hardware will auto reload the initial value to UART_BAUD, and when the UART_BAUD be updated, the SLVDUEN (UART_LINCTL [3]) will be cleared automatically. The behavior of LIN updated method as shown in the Figure 6.13-19 and Figure 6.13-20.

Note1: It is recommended to set the SLVDUEN bit before every checksum reception.

Note2: When a header error is detected, user must write 1 to SLVSYNCF (UART_LINSTS[3]) to re-search new frame header. When writing 1 to it, hardware will reload the initial baud rate TEMP_REG and re-search new frame header.

Note3: When operating in Automatic Resynchronization mode, the baud rate setting must be operated at mode2 (BAUDM1 (UART_BAUD [29]) and BAUDM0 (UART_BAUD[28]) must be 1).

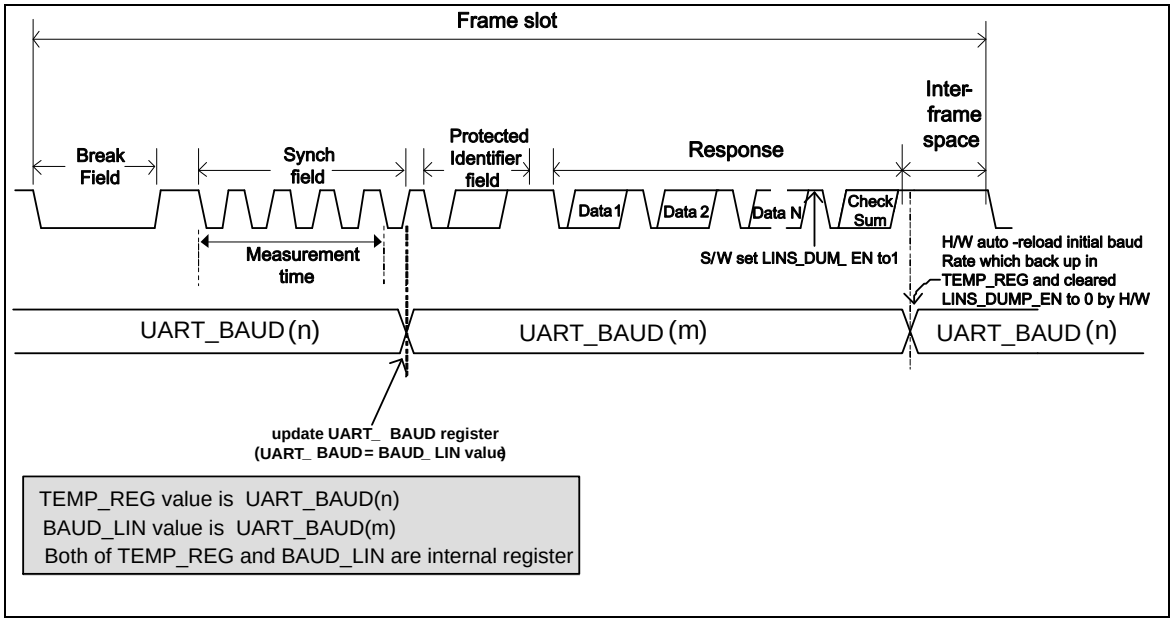


Figure 6.13-19 UART_BAUD Update Sequence in AR mode if SLVDUEN is 1

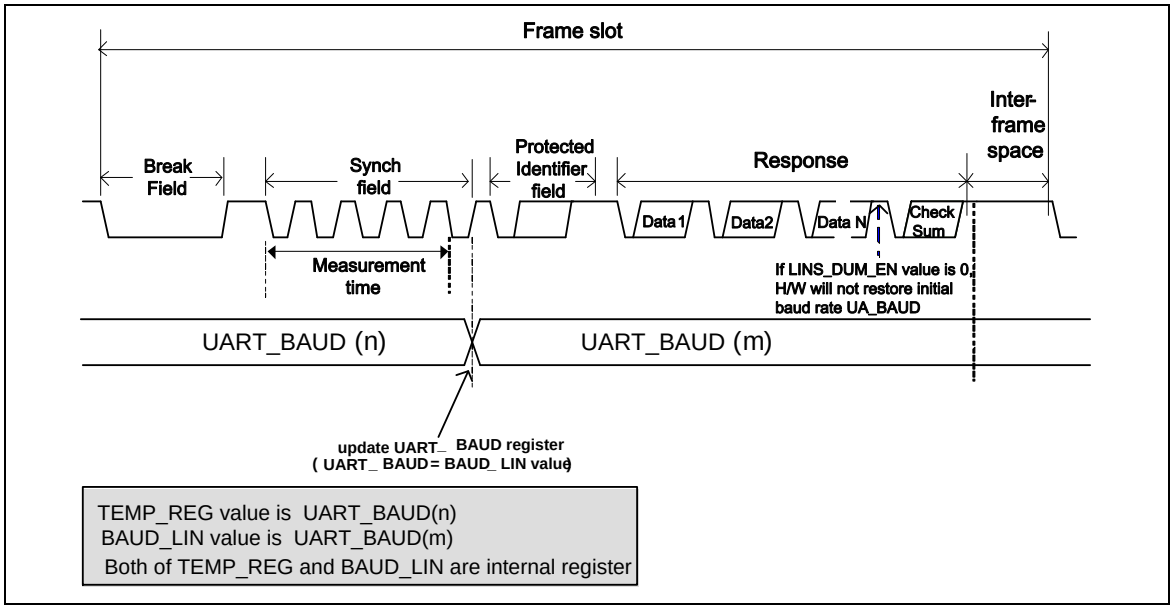


Figure 6.13-20 UART_BAUD Update Sequence in AR mode if SLVDUEN is 0

Deviation error on the sync field

When operating in Automatic Resynchronization mode, the controller will check the deviation error on the sync field. The deviation error is checked by comparing the current baud rate with the received sync field. Two checks are performed in parallel.

Check1: Based on measurement between the first falling edge and the last falling edge of the sync field.

- If the difference is more than 14.84%, the header error flag SLVHEF (UART_LINSTS[1]) will be set.
- If the difference is less than 14.06%, the header error flag SLVHEF (UART_LINSTS[1]) will not be set.

- If the difference is between 14.84% and 14.06%, the header error flag SLVHEF (UART_LINSTS[1]) may either set or not.

Check2: Based on measurement of time between each falling edge of the sync field.

- If the difference is more than 18.75%, the header error flag SLVHEF (UART_LINSTS[1]) will be set.
- If the difference is less than 15.62%, the header error flag SLVHEF (UART_LINSTS[1]) will not be set.
- If the difference is between 18.75% and 15.62%, the header error flag SLVHEF (UART_LINSTS[1]) may either set or not.

Note: The deviation check is based on the current baud rate clock. Therefore, in order to guarantee correct deviation checking, the baud rate must reload the nominal value before each new break reception by setting SLVDUEN (UART_LINCTL[3]) register (It is recommend setting the SLVDUEN (UART_LINCTL[3]) bit before every checksum reception)

LIN header error detection

In LIN Slave function mode, when user enables the header detection function by setting the SLVHDEN (UART_LINCTL[1]), hardware will handle the header detect flow. If the header has an error, the LIN header error flag SLVHEF (UART_LINSTS[1]) will be set and an interrupt is generated if the LINIEN (UART_INTEN[8]) bit is set. When header error is detected, user must reset the detect circuit to re-search a new frame header by writing 1 to SLVSYNCF (UART_LINSTS[3]) to re-search a new frame header.

The LIN header error flag SLVHEF (UART_LINSTS[1]) is set if one of the following conditions occurs:

- Break Delimiter is too short (less than 0.5-bit time).
- Frame error in sync field or Identifier field.
- The sync field data is not 0x55 (Non-Automatic Resynchronization mode).
- The sync field deviation error (With Automatic Resynchronization mode).
- The sync field measure time-out (With Automatic Resynchronization mode).
- LIN header reception time-out.

6.13.5.10 RS-485 Function Mode

Another alternate function of UART Controller is RS-485 function (user must set UART_FUNCSEL [1:0] to ’11’ to enable RS-485 function), and direction control provided by nRTS pin from an asynchronous serial port. The RS-485 transceiver control is implemented by using the nRTS control signal to enable the RS-485 driver. Many characteristics of the RX and TX are same as UART in RS-485 mode.

The UART controller can be configured as an RS-485 addressable slave and the RS-485 master transmitter will identify an address character by setting the parity (9-th bit) to 1. For data characters, the parity is set to 0. Software can use UART_LINE register to control the 9-th bit (When the PBE, EPE and SPE are set, the 9-th bit is transmitted 0 and when PBE and SPE are set and EPE is cleared, the 9-th bit is transmitted 1).

The controller supports three operation modes: RS-485 Normal Multidrop Operation Mode (NMM), RS-485 Auto Address Detection Operation Mode (AAD) and RS-485 Auto Direction

Control Operation Mode (AUD). Software can choose any operation mode by programming the UART_ALTCTL register, and drive the transfer delay time between the last stop bit leaving the TX FIFO and the de-assertion of by setting DLY (UART_TOUT [15:8]) register.

RS-485 Normal Multidrop Operation Mode (NMM)

In RS-485 Normal Multidrop Operation Mode (RS485NMM (UART_ALTCTL[8]) = 1), in first, software must decide the data which before the address byte be detected will be stored in RX FIFO or not. If software wants to ignore any data before address byte detected, the flow is set RXOFF (UART_FIFO [8]) then enable RS485NMM (UART_ALTCTL [8]) and the receiver will ignore any data until an address byte is detected (bit 9 = 1) and the address byte data will be stored in the RX FIFO. If software wants to receive any data before address byte detected, the flow is disables RXOFF (UART_FIFO [8]) then enable RS485NMM (UART_ALTCTL [8]) and the receiver will received any data.

If an address byte is detected (bit 9 = 1), it will generate an interrupt to CPU and RXOFF (UART_FIFO [8]) can decide whether accepting the following data bytes are stored in the RX FIFO. If software disables receiver by setting RXOFF (UART_FIFO [8]) register, when a next address byte is detected, the controller will clear the RXOFF (UART_FIFO [8]) bit and the address byte data will be stored in the RX FIFO.

RS-485 Auto Address Detection Operation Mode (AAD)

In RS-485 Auto Address Detection Operation Mode (RS485AAD (UART_ALTCTL[9]) = 1), the receiver will ignore any data until an address byte is detected (bit 9 = 1) and the address byte data matches the ADDRMV (UART_ALTCTL[31:24]) value. The address byte data will be stored in the RX FIFO. The all received byte data will be accepted and stored in the RX FIFO until an address byte data not match the ADDRMV (UART_ALTCTL[31:24]) value.

RS-485 Auto Direction Function (AUD)

Another option function of RS-485 controllers is RS-485 auto direction control function (RS485AUD (UART_ALTCTL[10) = 1). The RS-485 transceiver control is implemented by using the nRTS control signal from an asynchronous serial port. The nRTS line is connected to the RS-485 transceiver enable pin such that setting the nRTS line to high (logic 1) enables the RS-485 transceiver. Setting the nRTS line to low (logic 0) puts the transceiver into the tri-state condition to disabled. User can set RTSACTLV in UART_MODEM register to change the nRTS driving level.

The Figure 6.13-21 demonstrates the RS-485 nRTS driving level in AUD mode. The nRTS pin will be automatically driven during TX data transmission.

Setting RTSACTLV(UART_MODEM[9]) can control nRTS pin output driving level. User can read the RTSSTS(UART_MODEM[13]) bit to get real nRTS pin output voltage logic status.

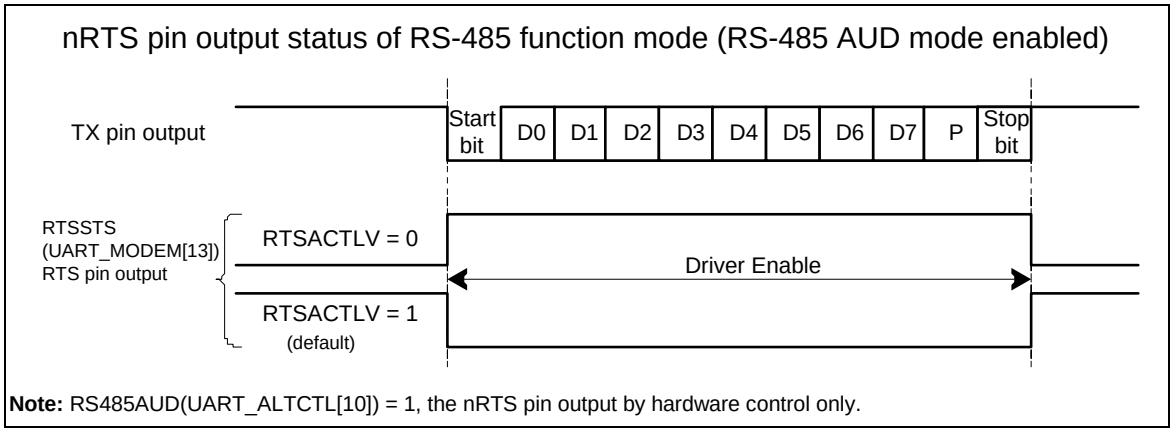


Figure 6.13-21 RS-485 nRTS Driving Level in Auto Direction Mode

The Figure 6.13-22 demonstrates the RS-485 nRTS driving level in software control (RS485AUD (UART_ALTCTL[10])=0). The nRTS driving level is controlled by programing the RTS(UART_MODEM[1]) control bit.

Setting RTSACTLV (UART_MODEM[9]) can control the nRTS pin output is inverse or non-inverse from RTS(UART_MODEM[1]) control bit. User can read the RTSSTS (UART_MODEM[13]) bit to get real nRTS pin output voltage logic status.

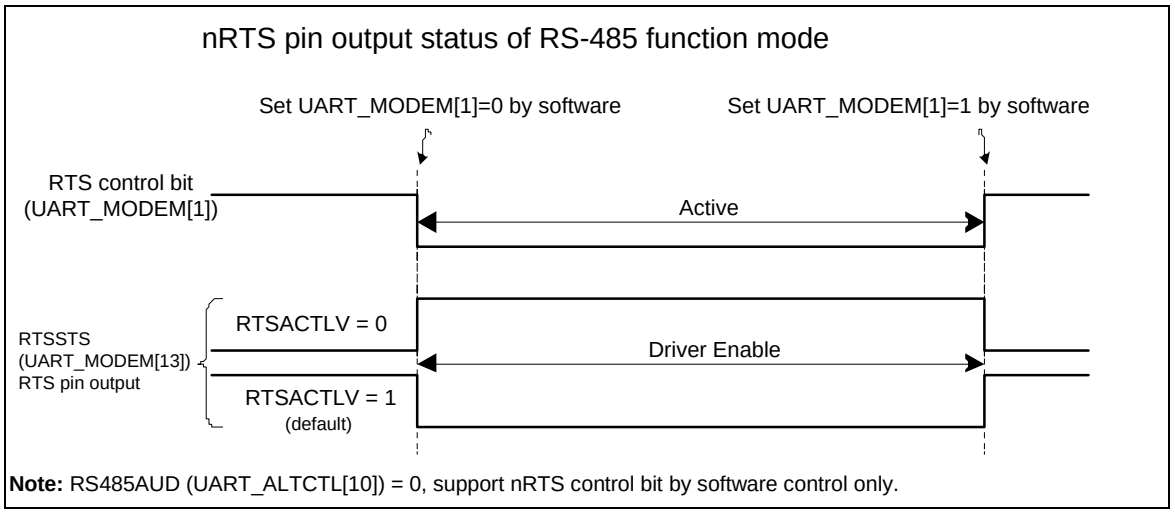
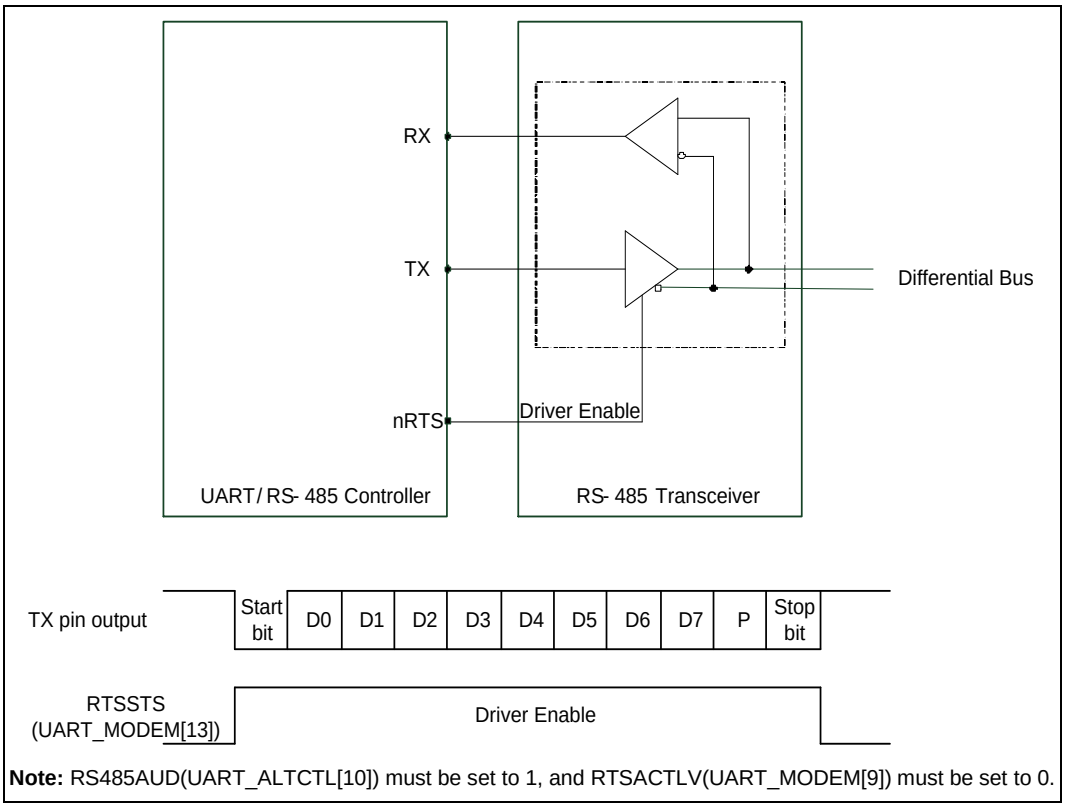


Figure 6.13-22 RS-485 nRTS Driving Level with Software Control

Programming Sequence Example:

1. Program FUNCSEL in UART_FUNCSEL to select RS-485 function.
2. Program the RXOFF (UART_FIFO[8]) to determine enable or disable the receiver RS-485 receiver
3. Program the RS485NMM (UART_ALTCTL[8]) or RS485AAD (UART_ALTCTL[9]) mode.
4. If the RS485AAD (UART_ALTCTL[9]) mode is selected, the ADDRMV (UART_ALTCTL[31:24]) is programmed for auto address match value.
5. Determine auto direction control by programming RS485AUD (UART_ALTCTL[10]).



Note: RS485AUD(UART_ALTCTL[10]) must be set to 1, and RTSACTLV(UART_MODEM[9]) must be set to 0.

Figure 6.13-23 Structure of RS-485 Frame

Programming Sequence Example:

1. Program ABRDBITS (UART_ALTCTL[20:19]) to determines UART RX data 1st rising edge time from Start by 2 ^{ABRDBITS} bit time.
2. Set ABRIEN (UART_INTEN[18]) to enable auto-baud rate function interrupt.
3. Set ABRDEN (UART_ALTCTL[18]) is to enable auto-baud rate function.
4. ABRDIF (UART_FIFOSTS[1]) is set, the auto-baud rate measurement is finished.
5. Operate UART transmit and receive action.
6. ABRDTOIF (UART_FIFOSTS[2]) is set, if auto-baud rate counter is overflow.
7. Go to Step 2.

6.13.6 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 = 0x4007_0000** **UART1_BA = 0x4007_1000** **UART2_BA = 0x4007_2000** **UART3_BA = 0x4007_3000**				
UART_DAT **x=0,1,2,3**	UARTx_BA+0x00	R/W	UART Receive/Transmit Buffer Register	Undefined
UART_INTEN **x=0,1,2,3**	UARTx_BA+0x04	R/W	UART Interrupt Enable Register	0x0000_0000
UART_FIFO **x=0,1,2,3**	UARTx_BA+0x08	R/W	UART FIFO Control Register	0x0000_0101
UART_LINE **x=0,1,2,3**	UARTx_BA+0x0C	R/W	UART Line Control Register	0x0000_0000
UART_MODEM **x=0,1,2,3**	UARTx_BA+0x10	R/W	UART Modem Control Register	0x0000_0200
UART_MODEMSTS **x=0,1,2,3**	UARTx_BA+0x14	R/W	UART Modem Status Register	0x0000_0110
UART_FIFOSTS **x=0,1,2,3**	UARTx_BA+0x18	R/W	UART FIFO Status Register	0x1040_4000
UART_INTSTS **x=0,1,2,3**	UARTx_BA+0x1C	R/W	UART Interrupt Status Register	0x0000_0002
UART_TOUT **x=0,1,2,3**	UARTx_BA+0x20	R/W	UART Time-out Register	0x0000_0000
UART_BAUD **x=0,1,2,3**	UARTx_BA+0x24	R/W	UART Baud Rate Divisor Register	0x0F00_0000
UART_IRDA **x=0,1,2,3**	UARTx_BA+0x28	R/W	UART IrDA Control Register	0x0000_0040
UART_ALTCTL **x=0,1,2,3**	UARTx_BA+0x2C	R/W	UART Alternate Control/Status Register	0x0000_000C
UART_FUNCSEL **x=0,1,2,3**	UARTx_BA+0x30	R/W	UART Function Select Register	0x0000_0000
UART_LINCTL **x=0,1**	UARTx_BA+0x34	R/W	UART LIN Control Register	0x000C_0000

UART_LINSTS **x=0,1**	UARTx_BA+0x38	R/W	UART LIN Status Register	0x0000_0000

6.13.7 Register Description

UART Receive/Transmit Buffer Register (UART_DAT)

Register	Offset	R/W	Description	Reset Value
UART_DAT x=0,1,2,3	UARTx_BA+0x00	R/W	UART Receive/Transmit Buffer Register	Undefined

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DAT							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**DAT**	**Receiving/Transmit Buffer** Write Operation: By writing one byte to this register, the data byte will be stored in transmitter FIFO. The UART Controller will send out the data stored in transmitter FIFO top location through the UART_TXD. Read Operation: By reading this register, the UART will return an 8-bit data received from receiving FIFO.

UART Interrupt Enable Register (UART_INTEN)

Register	Offset	R/W	Description	Reset Value
UART_INTEN x=0,1,2,3	UARTx_BA+0x04	R/W	UART Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**ABRIEN**	**Reserved**	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RXPDMAEN	**TXPDMAEN**	**ATOCTSEN**	**ATORTSEN**	**TOCNTEN**	**WKDATIEN**	**WKCTSIEN**	**LINIEN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**BUFERRIEN**	**RXTOIEN**	**MODEMIEN**	**RLSIEN**	**THREIEN**	**RDAIEN**

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**ABRIEN**	**Auto-baud Rate Interrupt Enable Bit** 0 = Auto-baud rate interrupt Disabled. 1 = Auto-baud rate interrupt Enabled.
[17:16]	**Reserved**	Reserved.
[15]	**RXPDMAEN**	**RX DMA Enable Bit** This bit can enable or disable RX DMA service. 0 = RX DMA Disabled. 1 = RX DMA Enabled.
[14]	**TXPDMAEN**	**TX DMA Enable Bit** This bit can enable or disable TX DMA service. 0 = TX DMA Disabled. 1 = TX DMA Enabled.
[13]	**ATOCTSEN**	**nCTS Auto-flow Control Enable Bit** 0 = nCTS auto-flow control Disabled. 1 = nCTS auto-flow control Enabled. **Note:** When nCTS auto-flow is enabled, the UART will send data to external device if nCTS input assert (UART will not send data to device until nCTS is asserted).
[12]	**ATORTSEN**	**nRTS Auto-flow Control Enable Bit** 0 = nRTS auto-flow control Disabled. 1 = nRTS auto-flow control Enabled. **Note:** When nRTS auto-flow is enabled, if the number of bytes in the RX FIFO equals the RTSTRGLV (UART_FIFO[19:16]), the UART will de-assert nRTS signal.
[11]	**TOCNTEN**	**Time-out Counter Enable Bit** 0 = Time-out counter Disabled. 1 = Time-out counter Enabled.

[10]	**WKDATIEN**	**Incoming Data Wake-up Interrupt Enable Bit** 0 = Incoming data wake-up system function Disabled. 1 = Incoming data wake-up system function Enabled, when the system is in Power-down mode, incoming data will wake-up system from Power-down mode.. **Note:** Hardware will clear this bit when the incoming data wake-up operation finishes and "system clock" work stable.
[9]	**WKCTSIEN**	**nCTS Wake-up Interrupt Enable Bit** 0 = nCTS wake-up system function Disabled. 1 = Wake-up system function Enabled, when the system is in Power-down mode, an external nCTS change will wake-up system from Power-down mode.
[8]	**LINIEN**	**LIN Bus Interrupt Enable Bit (Not Available in UART2/UART3)** 0 = LIN bus interrupt Disabled. 1 = LIN bus interrupt Enabled. **Note:** This bit is used for LIN function mode.
[7:6]	**Reserved**	Reserved.
[5]	**BUFERRIEN**	**Buffer Error Interrupt Enable Bit** 0 = Buffer error interrupt Disabled. 1 = Buffer error interrupt Enabled.
[4]	**RXTOIEN**	**RX Time-out Interrupt Enable Bit** 0 = RX time-out interrupt Disabled. 1 = RX time-out interrupt Enabled.
[3]	**MODEMIEN**	**Modem Status Interrupt Enable Bit** 0 = Modem status interrupt Disabled. 1 = Modem status interrupt Enabled.
[2]	**RLSIEN**	**Receive Line Status Interrupt Enable Bit** 0 = Receive Line Status interrupt Disabled. 1 = Receive Line Status interrupt Enabled.
[1]	**THREIEN**	**Transmit Holding Register Empty Interrupt Enable Bit** 0 = Transmit holding register empty interrupt Disabled. 1 = Transmit holding register empt interrupt Enabled.
[0]	**RDAIEN**	**Receive Data Available Interrupt Enable Bit** 0 = Receive data available interrupt Disabled. 1 = Receive data available interrupt Enabled.

UART FIFO Control Register (UART_FIFO)

Register	Offset	R/W	Description	Reset Value
UART_FIFO x=0,1,2,3	UARTx_BA+0x08	R/W	UART FIFO Control Register	0x0000_0101

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved				**RTSTRGLV**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**RXOFF**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RFITL				**Reserved**	**TXRST**	**RXRST**	**Reserved**

Bits	Description	
[31:20]	**Reserved**	Reserved.
[19:16]	**RTSTRGLV**	**nRTS Trigger Level for Auto-flow Control Use** 0000 = nRTS Trigger Level is 1 byte. 0001 = nRTS Trigger Level is 4 bytes. 0010 = nRTS Trigger Level is 8 bytes. 0011 = nRTS Trigger Level is 14 bytes. Others = Reserved. **Note:** This field is used for auto nRTS flow control.
[15:9]	**Reserved**	Reserved.
[8]	**RXOFF**	**Receiver Disable** The receiver is disabled or not (set 1 to disable receiver) 0 = Receiver Enabled. 1 = Receiver Disabled. **Note:** This bit is used for RS-485 Normal Multi-drop mode. It should be programmed before RS485NMM (UART_ALTCTL [8]) is programmed.
[7:4]	**RFITL**	**RX FIFO Interrupt Trigger Level** When the number of bytes in the receive FIFO equals the RFITL, the RDAIF will be set (if RDAIEN (UART_INTEN [0]) enabled, and an interrupt will be generated). 0000 = RX FIFO Interrupt Trigger Level is 1 byte. 0001 = RX FIFO Interrupt Trigger Level is 4 bytes. 0010 = RX FIFO Interrupt Trigger Level is 8 bytes. 0011 = RX FIFO Interrupt Trigger Level is 14 bytes. Others = Reserved.
[3]	**Reserved**	Reserved.
[2]	**TXRST**	**TX Field Software Reset** When TXRST (UART_FIFO[2]) is set, all the byte in the transmit FIFO and TX internal

		state machine are cleared. 0 = No effect. 1 = Reset the TX internal state machine and pointers. **Note:** This bit will automatically clear at least 3 UART peripheral clock cycles.
[1]	**RXRST**	**RX Field Software Reset** When RXRST (UART_FIFO[1]) is set, all the byte in the receiver FIFO and RX internal state machine are cleared. 0 = No effect. 1 = Reset the RX internal state machine and pointers. **Note:** This bit will automatically clear at least 3 UART peripheral clock cycles.
[0]	**Reserved**	Reserved.

UART Line Control Register (UART_LINE)

Register	Offset	R/W	Description	Reset Value
UART_LINE **x=0,1,2,3**	UARTx_BA+0x0C	R/W	UART Line Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**BCB**	**SPE**	**EPE**	**PBE**	**NSB**	**WLS**	

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**BCB**	**Break Control Bit** 0 = Break Control Disabled. 1 = Break Control Enabled. **Note:** When this bit is set to logic 1, the serial data output (TX) is forced to the Spacing State (logic 0). This bit acts only on TX line and has no effect on the transmitter logic.
[5]	**SPE**	**Stick Parity Enable Bit** 0 = Stick parity Disabled. 1 = Stick parity Enabled. **Note:** If PBE (UART_LINE[3]) and EPE (UART_LINE[4]) are logic 1, the parity bit is transmitted and checked as logic 0. If PBE (UART_LINE[3]) is 1 and EPE (UART_LINE[4]) is 0 then the parity bit is transmitted and checked as 1.
[4]	**EPE**	**Even Parity Enable Bit** 0 = Odd number of logic 1's is transmitted and checked in each word. 1 = Even number of logic 1's is transmitted and checked in each word. **Note:**This bit has effect only when PBE (UART_LINE[3]) is set.
[3]	**PBE**	**Parity Bit Enable Bit** 0 = No parity bit generated Disabled. 1 = Parity bit generated Enabled. Note : Parity bit is generated on each outgoing character and is checked on each incoming data.
[2]	**NSB**	**Number of "STOP Bit"** 0 = One "STOP bit" is generated in the transmitted data. 1 = When select 5-bit word length, 1.5 "STOP bit" is generated in the transmitted data. When select 6-, 7- and 8-bit word length, 2 "STOP bit" is generated in the transmitted data.
[1:0]	**WLS**	**Word Length Selection** This field sets UART word length.

		00 = 5 bits. 01 = 6 bits. 10 = 7 bits. 11 = 8 bits.

UART MODEM Control Register (UART_MODEM)

Register	Offset	R/W	Description	Reset Value
UART_MODEM x=0,1,2,3	UARTx_BA+0x10	R/W	UART Modem Control Register	0x0000_0200

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**RTSSTS**	**Reserved**			**RTSACTLV**	**Reserved**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**RTS**	**Reserved**

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**RTSSTS**	**nRTS Pin Status (Read Only)** This bit mirror from nRTS pin output of voltage logic status. 0 = nRTS pin output is low level voltage logic state. 1 = nRTS pin output is high level voltage logic state.
[12:10]	**Reserved**	Reserved.
[9]	**RTSACTLV**	**nRTS Pin Active Level** This bit defines the active level state of nRTS pin output. 0 =n RTS pin output is high level active. 1 = nRTS pin output is low level active. (Default) **Note1:** Refer to Figure 6.13-10 and Figure 6.13-11 for UART function mode. **Note2:** Refer to Figure 6.13-21 and Figure 6.13-22 for RS-485 function mode.
[8:2]	**Reserved**	Reserved.
[1]	**RTS**	**nRTS (Request-to-send) Signal Control** This bit is direct control internal nRTS signal active or not, and then drive the nRTS pin output with RTSACTLV bit configuration. 0 = nRTS signal is active. 1 = nRTS signal is inactive. **Note1:** This nRTS signal control bit is not effective when nRTS auto-flow control is enabled in UART function mode. **Note2:** This nRTS signal control bit is not effective when RS-485 auto direction mode (AUD) is enabled in RS-485 function mode.
[0]	**Reserved**	Reserved.

UART Modem Status Register (UART_MODEMSTS)

Register	Offset	R/W	Description	Reset Value
UART_MODEMSTS **x=0,1,2,3**	UARTx_BA+0x14	R/W	UART Modem Status Register	0x0000_0110

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**CTSACTLV**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**CTSSTS**	**Reserved**			**CTSDETF**

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**CTSACTLV**	**nCTS Pin Active Level** This bit defines the active level state of nCTS pin input. 0 = nCTS pin input is high level active. 1 = nCTS pin input is low level active. (Default)
[7:5]	**Reserved**	Reserved.
[4]	**CTSSTS**	**nCTS Pin Status (Read Only)** This bit mirror from nCTS pin input of voltage logic status. 0 = nCTS pin input is low level voltage logic state. 1 = nCTS pin input is high level voltage logic state. **Note:** This bit echoes when UART Controller peripheral clock is enabled, and nCTS multi-function port is selected.
[3:1]	**Reserved**	Reserved.
[0]	**CTSDETF**	**Detect nCTS State Change Flag (Read Only)** This bit is set whenever nCTS input has change state, and it will generate Modem interrupt to CPU when MODEMIEN (UART_INTEN [3]) is set to 1. 0 = nCTS input has not change state. 1 = nCTS input has change state. **Note:** This bit is read only, but can be cleared by writing “1” to it.

UART FIFO Status Register (UART_FIFOSTS)

Register	Offset	R/W	Description	Reset Value
UART_FIFOSTS x=0,1,2,3	UARTx_BA+0x18	R/W	UART FIFO Status Register	0x1040_4000

31	30	29	28	27	26	25	24
Reserved			TXEMPTYF	Reserved			TXOVIF
23	22	21	20	19	18	17	16
TXFULL	TXEMPTY	TXPTR					
15	14	13	12	11	10	9	8
RXFULL	RXEMPTY	RXPTR					
7	6	5	4	3	2	1	0
Reserved	BIF	FEF	PEF	ADDRDETF	ABRDTOIF	ABRDIF	RXOVIF

Bits	Description	
[31:29]	**Reserved**	Reserved.
[28]	**TXEMPTYF**	**Transmitter Empty Flag (Read Only)** This bit is set by hardware when TX FIFO (UART_DAT) is empty and the STOP bit of the last byte has been transmitted. 0 = TX FIFO is not empty or the STOP bit of the last byte has been not transmitted. 1 = TX FIFO is empty and the STOP bit of the last byte has been transmitted. **Note:** This bit is cleared automatically when TX FIFO is not empty or the last byte transmission has not completed.
[27:25]	**Reserved**	Reserved.
[24]	**TXOVIF**	**TX Overflow Error Interrupt Flag (Read Only)** If TX FIFO (UART_DAT) is full, an additional write to UART_DAT will cause this bit to logic 1. 0 = TX FIFO is not overflow. 1 = TX FIFO is overflow. **Note:** This bit is read only, but can be cleared by writing “1” to it.
[23]	**TXFULL**	**Transmitter FIFO Full (Read Only)** This bit indicates TX FIFO full or not. 0 = TX FIFO is not full. 1 = TX FIFO is full. **Note:** This bit is set when the number of usage in TX FIFO Buffer is equal to 16, otherwise is cleared by hardware.
[22]	**TXEMPTY**	**Transmitter FIFO Empty (Read Only)** This bit indicates TX FIFO empty or not. 0 = TX FIFO is not empty. 1 = TX FIFO is empty. **Note:** When the last byte of TX FIFO has been transferred to Transmitter Shift Register, hardware sets this bit high. It will be cleared when writing data into DAT (TX FIFO not empty).

[21:16]	**TXPTR**	**TX FIFO Pointer (Read Only)** This field indicates the TX FIFO Buffer Pointer. When CPU writes one byte into UART_DAT, TXPTR increases one. When one byte of TX FIFO is transferred to Transmitter Shift Register, TXPTR decreases one. The Maximum value shown in TXPTR is 15. When the using level of TX FIFO Buffer equal to 16, the TXFULL bit is set to 1 and TXPTR will show 0. As one byte of TX FIFO is transferred to Transmitter Shift Register, the TXFULL bit is cleared to 0 and TXPTR will show 15.
[15]	**RXFULL**	**Receiver FIFO Full (Read Only)** This bit initiates RX FIFO full or not. 0 = RX FIFO is not full. 1 = RX FIFO is full. **Note:** This bit is set when the number of usage in RX FIFO Buffer is equal to 16, otherwise is cleared by hardware.
[14]	**RXEMPTY**	**Receiver FIFO Empty (Read Only)** This bit initiate RX FIFO empty or not. 0 = RX FIFO is not empty. 1 = RX FIFO is empty. **Note:** When the last byte of RX FIFO has been read by CPU, hardware sets this bit high. It will be cleared when UART receives any new data.
[13:8]	**RXPTR**	**RX FIFO Pointer (Read Only)** This field indicates the RX FIFO Buffer Pointer. When UART receives one byte from external device, RXPTR increases one. When one byte of RX FIFO is read by CPU, RXPTR decreases one. The Maximum value shown in RXPTR is 15. When the using level of RX FIFO Buffer equal to 16, the RXFULL bit is set to 1 and RXPTR will show 0. As one byte of RX FIFO is read by CPU, the RXFULL bit is cleared to 0 and RXPTR will show 15.
[7]	**Reserved**	Reserved.
[6]	**BIF**	**Break Interrupt Flag (Read Only)** This bit is set to logic 1 whenever the received data input (RX) is held in the “spacing state” (logic 0) for longer than a full word transmission time (that is, the total time of “start bit” + data bits + parity + stop bits). 0 = No Break interrupt is generated. 1 = Break interrupt is generated. **Note:** This bit is read only, but can be cleared by writing ‘1’ to it.
[5]	**FEF**	**Framing Error Flag (Read Only)** This bit is set to logic 1 whenever the received character does not have a valid “stop bit” (that is, the stop bit following the last data bit or parity bit is detected as logic 0). 0 = No framing error is generated. 1 = Framing error is generated. **Note:** This bit is read only, but can be cleared by writing ‘1’ to it.
[4]	**PEF**	**Parity Error Flag (Read Only)** This bit is set to logic 1 whenever the received character does not have a valid “parity bit”. 0 = No parity error is generated. 1 = Parity error is generated. **Note:** This bit is read only, but can be cleared by writing ‘1’ to it.
[3]	**ADDRDETF**	**RS-485 Address Byte Detect Flag (Read Only)** 0 = Receiver detects a data that is not an address bit (bit 9 =’0’). 1 = Receiver detects a data that is an address bit (bit 9 =’1’).

		Note1: This field is used for RS-485 function mode and ADDRDEN (UART_ALTCTL[15]) is set to 1 to enable Address detection mode . **Note2:** This bit is read only, but can be cleared by writing '1' to it.
[2]	**ABRDTOIF**	**Auto-baud Rate Time-out Interrupt (Read Only)** 0 = Auto-baud rate counter is underflow. 1 = Auto-baud rate counter is overflow. **Note1:** This bit is set to logic "1" in Auto-baud Rate Detect mode and the baud rate counter is overflow. **Note2:** This bit is read only, but can be cleared by writing "1" to it.
[1]	**ABRDIF**	**Auto-baud Rate Detect Interrupt (Read Only)** 0 = Auto-baud rate detect function is not finished. 1 = Auto-baud rate detect function is finished. This bit is set to logic "1" when auto-baud rate detect function is finished. **Note:** This bit is read only, but can be cleared by writing "1" to it.
[0]	**RXOVIF**	**RX Overflow Error Interrupt Flag (Read Only)** This bit is set when RX FIFO overflow. If the number of bytes of received data is greater than RX_FIFO (UART_DAT) size, 16 bytes this bit will be set. 0 = RX FIFO is not overflow. 1 = RX FIFO is overflow. **Note:** This bit is read only, but can be cleared by writing "1" to it.

UART Interrupt Status Control Register (UART_INTSTS)

Register	Offset	R/W	Description	Reset Value
UART_INTSTS x=0,1,2,3	UARTx_BA+0x1C	R/W	UART Interrupt Status Register	0x0000_0002

31	30	29	28	27	26	25	24
Reserved		HWBUFEINT	HWTOINT	HWMODINT	HWRLSINT	Reserved	
23	22	21	20	19	18	17	16
Reserved		HWBUFEIF	HWTOIF	HWMODIF	HWRLSIF	DATWKIF	CTSWKIF
15	14	13	12	11	10	9	8
LININT	Reserved	BUFERRINT	RXTOINT	MODEMINT	RLSINT	THREINT	RDAINT
7	6	5	4	3	2	1	0
LINIF	WKIF	BUFERRIF	RXTOIF	MODEMIF	RLSIF	THREIF	RDAIF

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**HWBUFEINT**	**in DMA Mode, Buffer Error Interrupt Indicator (Read Only)** This bit is set if BFERRIEN (UART_INTEN[5]) and HWBEIF (UART_INTSTS[5])are both set to 1. 0 = No buffer error interrupt is generated in DMA mode. 1 = Buffer error interrupt is generated in DMA mode.
[28]	**HWTOINT**	**in DMA Mode, Time-out Interrupt Indicator (Read Only)** This bit is set if TOUTIEN (UART_INTEN[4])and HWTOIF(UART_INTSTS[20]) are both set to 1. 0 = No Tout interrupt is generated in DMA mode. 1 = Tout interrupt is generated in DMA mode.
[27]	**HWMODINT**	**in DMA Mode, MODEM Status Interrupt Indicator (Read Only)** This bit is set if MODEMIEN(UART_INTEN[3]) and HWMODIF(UART_INTSTS[3]) are both set to 1. 0 = No Modem interrupt is generated in DMA mode. 1 = Modem interrupt is generated in DMA mode.
[26]	**HWRLSINT**	**in DMA Mode, Receive Line Status Interrupt Indicator (Read Only)** This bit is set if RLSIEN (UART_INTEN[2])and HWRLSIF(UART_INTSTS[18]) are both set to 1. 0 = No RLS interrupt is generated in DMA mode. 1 = RLS interrupt is generated in DMA mode.
[25:22]	**Reserved**	Reserved.
[21]	**HWBUFEIF**	**in DMA Mode, Buffer Error Interrupt Flag (Read Only)** This bit is set when the TX or RX FIFO overflows (TXOVIF (UART_FIFOSTS [24]) or RXOVIF (UART_FIFOSTS[0]) is set). When BERRIF (UART_INTSTS[5]) is set, the transfer maybe is not correct. If BFERRIEN (UART_INTEN [5]) is enabled, the buffer error interrupt will be generated. 0 = No buffer error interrupt flag is generated.

		1 = Buffer error interrupt flag is generated. **Note:** This bit is cleared when both TXOVIF (UART_FIFOSTS[24]]) and RXOVIF (UART_FIFOSTS[0]) are cleared.
[20]	**HWTOIF**	**in DMA Mode, Time-out Interrupt Flag (Read Only)** This bit is set when the RX FIFO is not empty and no activities occurred in the RX FIFO and the time-out counter equal to TOIC (UART_TOUT[7:0]). If TOUTIEN (UART_INTEN [4]) is enabled, the Tout interrupt will be generated. 0 = No Time-out interrupt flag is generated. 1 = Time-out interrupt flag is generated. **Note:** This bit is read only and user can read UART_DAT (RX is in active) to clear it.
[19]	**HWMODIF**	**in DMA Mode, MODEM Interrupt Flag (Read Only)** This bit is set when the nCTS pin has state change (CTSDETF (UART_CTSDETF[0] =1)). If MODEMIEN (UART_INTEN [3]) is enabled, the Modem interrupt will be generated. 0 = No Modem interrupt flag is generated. 1 = Modem interrupt flag is generated. **Note:** This bit is read only and reset to 0 when the bit UART_CTSDETF (US_MSR[0]) is cleared by writing 1 on CTSDETF (UART_CTSDETF [0]).
[18]	**HWRLSIF**	**in DMA Mode, Receive Line Status Flag (Read Only)** This bit is set when the RX receive data have parity error, frame error or break error (at least one of 3 bits, BIF (UART_FIFOSTS[6]), FEF (UART_FIFOSTS[5]) and PEF (UART_FIFOSTS[4]) is set). If RLSIEN (UART_INTEN [2]) is enabled, the RLS interrupt will be generated. 0 = No RLS interrupt flag is generated. 1 = RLS interrupt flag is generated. **Note1:** In RS-485 function mode, this field include “receiver detect any address byte received address byte character (bit9 = ‘1’) bit". **Note2:** In UART function mode, this bit is read only and reset to 0 when all bits of BIF(UART_FIFOSTS[6]) , FEF(UART_FIFOSTS[5]) and PEF(UART_FIFOSTS[4]) are cleared. **Note3:** In RS-485 function mode, this bit is read only and reset to 0 when all bits of BIF(UART_FIFOSTS[6]) , FEF(UART_FIFOSTS[5]) and PEF(UART_FIFOSTS[4]) and ADDRDETF (UART_FIFOSTS[3]) are cleared.
[17]	**DATWKIF**	**Data Wake-up Interrupt Flag (Read Only)** This bit is set if chip wake-up from power-down state by data wake-up. 0 = Chip stays in power-down state. 1 = Chip wake-up from power-down state by data wake-up. **Note1:** If WKDATIEN (UART_INTEN[10]) is enabled, the wake-up interrupt is generated. **Note2:** This bit is read only, but can be cleared by writing ‘1’ to it.
[16]	**CTSWKIF**	**nCTS Wake-up Interrupt Flag (Read Only)** 0 = Chip stays in power-down state. 1 = Chip wake-up from power-down state by nCTS wake-up. **Note1:** If WKCTSIEN (UART_INTEN[9])is enabled, the wake-up interrupt is generated. **Note2:** This bit is read only, but can be cleared by writing ‘1’ to it.
[15]	**LININT**	**LIN Bus Interrupt Indicator (Read Only)(Not Available in UART2/UART3 Channel)** This bit is set if LINIEN (UART_INTEN[8]) and LIN IF(UART_INTSTS[7]) are both set to 1. 0 = No LIN Bus interrupt is generated. 1 = The LIN Bus interrupt is generated.
[14]	**Reserved**	Reserved.

[13]	**BUFERRINT**	**Buffer Error Interrupt Indicator (Read Only)** This bit is set if BFERRIEN(UART_INTEN[5] and BERRIF(UART_INTSTS[5]) are both set to 1. 0 = No buffer error interrupt is generated. 1 = Buffer error interrupt is generated.
[12]	**RXTOINT**	**Time-out Interrupt Indicator (Read Only)** This bit is set if TOUTIEN(UART_INTEN[4]) and RXTOIF(UART_INTSTS[4]) are both set to 1. 0 = No Tout interrupt is generated. 1 = Tout interrupt is generated.
[11]	**MODEMINT**	**MODEM Status Interrupt Indicator (Read Only)** This bit is set if MODEMIEN(UART_INTEN[3] and MODEMIF(UART_INTSTS[4]) are both set to 1 0 = No Modem interrupt is generated. 1 = Modem interrupt is generated..
[10]	**RLSINT**	**Receive Line Status Interrupt Indicator (Read Only)** This bit is set if RLSIEN (UART_INTEN[2]) and RLSIF(UART_INTSTS[2]) are both set to 1. 0 = No RLS interrupt is generated. 1 = RLS interrupt is generated.
[9]	**THREINT**	**Transmit Holding Register Empty Interrupt Indicator (Read Only)** This bit is set if THREIEN (UART_INTEN[1])and THREIF(UART_INTSTS[1]) are both set to 1. 0 = No THRE interrupt is generated. 1 = THRE interrupt is generated.
[8]	**RDAINT**	**Receive Data Available Interrupt Indicator (Read Only)** This bit is set if RDAIEN (UART_INTEN[0]) and RDAIF (UART_INTSTS[0]) are both set to 1. 0 = No RDA interrupt is generated. 1 = RDA interrupt is generated.
[7]	**LINIF**	**LIN Bus Interrupt Flag (Read Only) (Not Available in UART2/UART3 Channel)** This bit is set when LIN slave header detect (SLVHDETF (UART_LINSTS[0] =1)), LIN break detect (BRKDETF(UART_LINSTS[9]=1)), bit error detect (BITEF(UART_LINSTS[9]=1), LIN slave ID parity error (SLVIDPEF(UART_LINSTS[2] = 1) or LIN slave header error detect (SLVHEF (UART_LINSTS[1])). If LIN_ IEN (UART_INTEN [8]) is enabled the LIN interrupt will be generated. 0 = None of SLVHDETF, BRKDETF, BITEF, SLVIDPEF and SLVHEF is generated. 1 = At least one of SLVHDETF, BRKDETF, BITEF, SLVIDPEF and SLVHEF is generated. **Note:** This bit is read only. This bit is cleared when SLVHDETF(UART_LINSTS[0]), BRKDETF(UART_LINSTS[8]), BITEF(UART_LINSTS[9]), SLVIDPEF (UART_LINSTS[2]), SLVHEF(UART_LINSTS[1]) and SLVSYNCF(UART_LINSTS[3]) all are cleared.
[6]	**WKIF**	**UART Wake-up Interrupt Flag (Read Only)** This bit is set when DATWKIF (UART_INTSTS[17]) or CTSWKIF(UART_INTSTS[16]) is set to 1. 0 = No DATWKIF and CTSWKIF are generated. 1 = DATWKIF or CTSWKIF. **Note:** This bit is read only. This bit is cleared if both of DATWKIF (UART_INTSTS[17]) and CTSWKIF(UART_INTSTS[16]) are cleared to 0 by writing 1 to DATWKIF (UART_INTSTS[17]) and CTSWKIF (UART_INTSTS[17]).

[5]	**BUFERRIF**	**Buffer Error Interrupt Flag (Read Only)** This bit is set when the TX FIFO or RX FIFO overflows (TXOVIF (UART_FIFOSTS[24]) or RXOVIF (UART_FIFOSTS[0]) is set). When BERRIF (UART_INTSTS[5])is set, the transfer is not correct. If BFERRIEN (UART_INTEN [8]) is enabled, the buffer error interrupt will be generated. 0 = No buffer error interrupt flag is generated. 1 = Buffer error interrupt flag is generated. **Note:** This bit is read only. This bit is cleared if both of RXOVIF(UART_FIFOSTS[0]) and TXOVIF(UART_FIFOSTS[24]) are cleared to 0 by writing 1 to RXOVIF(UART_FIFOSTS[0]) and TXOVIF(UART_FIFOSTS[24]).
[4]	**RXTOIF**	**Time-out Interrupt Flag (Read Only)** This bit is set when the RX FIFO is not empty and no activities occurred in the RX FIFO and the time-out counter equal to TOIC. If TOUTIEN (UART_INTEN [4]) is enabled, the Tout interrupt will be generated. 0 = No Time-out interrupt flag is generated. 1 = Time-out interrupt flag is generated. **Note:** This bit is read only and user can read UART_DAT (RX is in active) to clear it.
[3]	**MODEMIF**	**MODEM Interrupt Flag (Read Only) Channel** This bit is set when the nCTS pin has state change (CTSDETF (UART_MODEMSTS[0]) = 1). If MODEMIEN (UART_INTEN [3]) is enabled, the Modem interrupt will be generated. 0 = No Modem interrupt flag is generated. 1 = Modem interrupt flag is generated. **Note:** This bit is read only and reset to 0 when bit CTSDETF is cleared by a write 1 on CTSDETF(UART_MODEMSTS[0]).
[2]	**RLSIF**	**Receive Line Interrupt Flag (Read Only)** This bit is set when the RX receive data have parity error, frame error or break error (at least one of 3 bits, BIF(UART_FIFOSTS[6]), FEF(UART_FIFOSTS[5]) and PEF(UART_FIFOSTS[4]), is set). If RLSIEN (UART_INTEN [2]) is enabled, the RLS interrupt will be generated. 0 = No RLS interrupt flag is generated. 1 = RLS interrupt flag is generated. **Note1:** In RS-485 function mode, this field is set include “receiver detect and received address byte character (bit9 = ‘1’) bit". At the same time, the bit of ADDRDETF (UART_FIFOSTS[3]) is also set. **Note2:** This bit is read only and reset to 0 when all bits of BIF (UART_FIFOSTS[6]), FEF(UART_FIFOSTS[5]) and PEF(UART_FIFOSTS[4]) are cleared. **Note3:** In RS-485 function mode, this bit is read only and reset to 0 when all bits of BIF (UART_FIFOSTS[6]) , FEF(UART_FIFOSTS[5]) and PEF(UART_FIFOSTS[4]) and ADDRDETF (UART_FIFOSTS[3]) are cleared.
[1]	**THREIF**	**Transmit Holding Register Empty Interrupt Flag (Read Only)** This bit is set when the last data of TX FIFO is transferred to Transmitter Shift Register. If THREIEN (UART_INTEN[1]) is enabled, the THRE interrupt will be generated. 0 = No THRE interrupt flag is generated. 1 = THRE interrupt flag is generated. **Note:** This bit is read only and it will be cleared when writing data into UART_DAT (TX FIFO not empty).
[0]	**RDAIF**	**Receive Data Available Interrupt Flag (Read Only)** When the number of bytes in the RX FIFO equals the RFITL then the RDAIF(UART_INTSTS[0]) will be set. If RDAIEN (UART_INTEN [0]) is enabled, the RDA interrupt will be generated. 0 = No RDA interrupt flag is generated. 1 = RDA interrupt flag is generated. **Note:** This bit is read only and it will be cleared when the number of unread bytes of RX FIFO drops below the threshold level (RFITL(UART_FIFO[7:4]).

UART Time-out Register (UART_TOUT)

Register	Offset	R/W	Description	Reset Value
UART_TOUT x=0,1,2,3	UARTx_BA+0x20	R/W	UART Time-out Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DLY							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TOIC							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**DLY**	**TX Delay Time Value** This field is used to programming the transfer delay time between the last stop bit and next start bit. The unit is bit time.
[7:0]	**TOIC**	**Time-out Interrupt Comparator** The time-out counter resets and starts counting (the counting clock = baud rate) whenever the RX FIFO receives a new data word. Once the content of time-out counter is equal to that of time-out interrupt comparator (TOIC (UART_TOUT[7:0])), a receiver time-out interrupt (RXTOINT(UART_INTSTS[12])) is generated if RXTOIEN (UART_INTEN [4]) enabled. A new incoming data word or RX FIFO empty will clear RXTOINT(UART_INTSTS[12]). In order to avoid receiver time-out interrupt generation immediately during one character is being received, TOIC value should be set between 40 and 255. So, for example, if TOIC is set with 40, the time-out interrupt is generated after four characters are not received when 1 stop bit and no parity check is set for UART transfer.

UART Baud Rate Divider Register (UART_BAUD)

Register	Offset	R/W	Description	Reset Value
UART_BAUD x=0,1,2,3	UARTx_BA+0x24	R/W	UART Baud Rate Divisor Register	0x0F00_0000

31	30	29	28	27	26	25	24
Reserved		BAUDM1	BAUDM0	EDIVM1			
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
BRD							
7	6	5	4	3	2	1	0
BRD							

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**BAUDM1**	**BAUD Rate Mode Selection Bit 1** This bit is baud rate mode selection bit 1. UART provides three baud rate calculation modes. This bit combines with BAUDM0 (UART_BAUD[28]) to select baud rate calculation mode. The detail description is shown in Table 6-22. **Note:** In IrDA mode must be operated in mode 0.
[28]	**BAUDM0**	**BAUD Rate Mode Selection Bit 0** This bit is baud rate mode selection bit 0. UART provides three baud rate calculation modes. This bit combines with BAUDM1 (UART_BAUD[29]) to select baud rate calculation mode. The detail description is shown in Table 6-22.
[27:24]	**EDIVM1**	**Extra Divider for BAUD Rate Mode 1** This field is used for baud rate calculation in mode 1 and has no effect for baud rate calculation in mode 0 and mode 2. The detail description is shown in Table 6-22.
[23:16]	**Reserved**	Reserved.
[15:0]	**BRD**	**Baud Rate Divider** The field indicates the baud rate divider. This filed is used in baud rate calculation. The detail description is shown in Table 6-22.

UART_IrDA Control Register (UART_IRDA)

Register	Offset	R/W	Description	Reset Value
UART_IRDA x=0,1,2,3	UARTx_BA+0x28	R/W	UART IrDA Control Register	0x0000_0040

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**RXINV**	**TXINV**	**Reserved**			**TXEN**	**Reserved**

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**RXINV**	**IrDA Inverse Receive Input Signal** 0 = None inverse receiving input signal. 1 = Inverse receiving input signal. (Default)
[5]	**TXINV**	**IrDA Inverse Transmitting Output Signal** 0 = None inverse transmitting signal. (Default) 1 = Inverse transmitting output signal.
[4:2]	**Reserved**	Reserved.
[1]	**TXEN**	**IrDA Receiver/Transmitter Selection Enable Bit** 0 = IrDA Transmitter Disabled and Receiver Enabled. (Default) 1 = IrDA Transmitter Enabled and Receiver Disabled.
[0]	**Reserved**	Reserved.

Note: In IrDA mode, the BAUDM1 (UART_BAUD [29]) register must be disabled, the baud equation must be Clock / (16 * (BRD + 2)).

UART Alternate Control/Status Register (UART_ALTCTL)

Register	Offset	R/W	Description	Reset Value
UART_ALTCTL x=0,1,2,3	UARTx_BA+0x2C	R/W	UART Alternate Control/Status Register	0x0000_000C

31	30	29	28	27	26	25	24
ADDRMV							
23	22	21	20	19	18	17	16
Reserved			ABRDBITS		ABRDEN	ABRIF	Reserved
15	14	13	12	11	10	9	8
ADDRDEN	Reserved				RS485AUD	RS485AAD	RS485NMM
7	6	5	4	3	2	1	0
LINTXEN	LINRXEN	Reserved		BRKFL			

Bits	Description	
[31:24]	**ADDRMV**	**Address Match Value** This field contains the RS-485 address match values. **Note:** This field is used for RS-485 auto address detection mode.
[23:21]	**Reserved**	Reserved.
[20:19]	**ABRDBITS**	**Auto-baud Rate Detect Bit Length** 00 = 1-bit time from Start bit to the 1st rising edge. The input pattern shall be 0x01. 01 = 2-bit time from Start bit to the 1st rising edge. The input pattern shall be 0x02. 10 = 4-bit time from Start bit to the 1st rising edge. The input pattern shall be 0x08. 11 = 8-bit time from Start bit to the 1st rising edge. The input pattern shall be 0x80. **Note :** The calculation of bit number includes the START bit.
[18]	**ABRDEN**	**Auto-baud Rate Detect Enable Bit** 0 = Auto-baud rate detect function Disabled. 1 = Auto-baud rate detect function Enabled. This bit is cleared automatically after auto-baud detection is finished.
[17]	**ABRIF**	**Auto-baud Rate Interrupt Flag (Read Only)** This bit is set when auto-baud rate detection function finished or the auto-baud rate counter was overflow and if ABRIEN(UART_INTEN [18]) is set then the auto-baud rate interrupt will be generated. **Note:** This bit is read only, but it can be cleared by writing “1” to ABRDTOIF (UART_FIFOSTS[2]) and ABRDIF(UART_FIFOSTS[1]).
[16]	**Reserved**	Reserved.
[15]	**ADDRDEN**	**RS-485 Address Detection Enable Bit** This bit is used to enable RS-485 Address Detection mode. 0 = Address detection mode Disabled. 1 = Address detection mode Enabled. **Note:** This bit is used for RS-485 any operation mode.

[14:11]	**Reserved**	Reserved.
[10]	**RS485AUD**	**RS-485 Auto Direction Function (AUD)** 0 = RS-485 Auto Direction Operation function (AUD) Disabled. 1 = RS-485 Auto Direction Operation function (AUD) Enabled. **Note:** It can be active with RS-485_AAD or RS-485_NMM operation mode.
[9]	**RS485AAD**	**RS-485 Auto Address Detection Operation Mode (AAD)** 0 = RS-485 Auto Address Detection Operation mode (AAD) Disabled. 1 = RS-485 Auto Address Detection Operation mode (AAD) Enabled. **Note:** It cannot be active with RS-485_NMM operation mode.
[8]	**RS485NMM**	**RS-485 Normal Multi-drop Operation Mode (NMM)** 0 = RS-485 Normal Multi-drop Operation mode (NMM) Disabled. 1 = RS-485 Normal Multi-drop Operation mode (NMM) Enabled. **Note:** It cannot be active with RS-485_AAD operation mode.
[7]	**LINTXEN**	**LIN TX Break Mode Enable Bit (Only Available in UART0/UART1 Channel)** 0 = LIN TX Break mode Disabled. 1 = LIN TX Break mode Enabled. **Note:** When TX break field transfer operation finished, this bit will be cleared automatically.
[6]	**LINRXEN**	**LIN RX Enable Bit (Only Available in UART0/UART1 Channel)** 0 = LIN RX mode Disabled. 1 = LIN RX mode Enabled.
[5:4]	**Reserved**	Reserved.
[3:0]	**BRKFL**	**UART LIN Break Field Length (Only Available in UART0/UART1 Channel)** This field indicates a 4-bit LIN TX break field count. **Note1:** This break field length is BRKFL + 1 **Note2:** According to LIN spec, the reset value is 0xC (break field length = 13).

UART Function Select Register (UART_FUNCSEL)

Register	Offset	R/W	Description	Reset Value
UART_FUNCSEL **x=0,1,2,3**	UARTx_BA+0x30	R/W	UART Function Select Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**FUNCSEL**	

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1:0]	**FUNCSEL**	**Function Select** 00 = UART function. 01 = LIN function (Only Available in UART0/UART1 Channel). 10 = IrDA function. 11 = RS-485 function.

UART LIN Control Register (UART_LINCTL) (Only Available in UART0/UART1 Channel)

Register	Offset	R/W	Description	Reset Value
UART_LINCTL x=0,1	UARTx_BA+0x34	R/W	UART LIN Control Register	0x000C_0000

31	30	29	28	27	26	25	24
PID							
23	22	21	20	19	18	17	16
HSEL		BSL		BRKFL			
15	14	13	12	11	10	9	8
Reserved			BITERREN	RXOFF	BRKDETEN	IDPEN	SENDH
7	6	5	4	3	2	1	0
Reserved			MUTE	SLVDUEN	SLVAREN	SLVHDEN	SLVEN

Bits	Description	
[31:24]	**PID**	**LIN PID Bits** This field contains the LIN frame ID value when in LIN function mode, the frame ID parity can be generated by software or hardware depends on IDPEN (UART_LINCTL[9]) = 1. If the parity generated by hardware, user fill ID0~ID5, (PID [29:24])hardware will calculate P0 (PID[30]) and P1 (PID[31]), otherwise user must filled frame ID and parity in this field. **Note1:** User can fill any 8-bit value to this field and the bit 24 indicates ID0 (LSB first). **Note2:** This field can be used for LIN master mode or slave mode.
[23:22]	**HSEL**	**LIN Header Select** 00 = The LIN header includes “break field”. 01 = The LIN header includes “break field” and “sync field”. 10 = The LIN header includes “break field”, “sync field” and “frame ID field”. 11 = Reserved. **Note:** This bit is used to master mode for LIN to send header field (SENDH (UART_LINCTL [8]) = 1) or used to slave to indicates exit from mute mode condition (MUTE (UART_LINCTL[4] = 1).
[21:20]	**BSL**	**LIN Break/Sync Delimiter Length** 00 = The LIN break/sync delimiter length is 1-bit time. 01 = The LIN break/sync delimiter length is 2-bit time. 10 = The LIN break/sync delimiter length is 3-bit time. 11 = The LIN break/sync delimiter length is 4-bit time. **Note:** This bit used for LIN master to sending header field.
[19:16]	**BRKFL**	**LIN Break Field Length** This field indicates a 4-bit LIN TX break field count. **Note1:** These registers are shadow registers of BRKFL, User can read/write it by setting BRKFL (UART_ALTCTL[3:0]) or BRKFL (UART_LINCTL[19:16]). **Note2:** This break field length is BRKFL + 1. **Note3:** According to LIN spec, the reset value is 12 (break field length = 13).
[15:13]	**Reserved**	Reserved.

[12]	**BITERREN**	**Bit Error Detect Enable Bit** 0 = Bit error detection function Disabled. 1 = Bit error detection Enabled. **Note:** In LIN function mode, when occur bit error, the BITEF (UART_LINSTS[9]) flag will be asserted. If the LINIEN (UART_INTEN[8]) = 1, an interrupt will be generated.
[11]	**RXOFF**	**LIN Receiver Disable Bit** If the receiver is enabled (RXOFF (UART_LINCTL[11]) = 0), all received byte data will be accepted and stored in the RX-FIFO, and if the receiver is disabled (RXOFF (UART_LINCTL[11] = 1), all received byte data will be ignore. 0 = LIN receiver Enabled. 1 = LIN receiver Disabled. **Note:** This bit is only valid when operating in LIN function mode (FUNCSEL (UART_FUNCSEL[1:0]) = 01).
[10]	**BRKDETEN**	**LIN Break Detection Enable Bit** When detect consecutive dominant greater than 11 bits, and are followed by a delimiter character, the BRKDETF (UART_LINSTS[8]) flag is set in UART_LINSTS register at the end of break field. If the LINIEN (UART_INTEN [8])=1, an interrupt will be generated. 0 = LIN break detection Disabled . 1 = LIN break detection Enabled.
[9]	**IDPEN**	**LIN ID Parity Enable Bit** 0 = LIN frame ID parity Disabled. 1 = LIN frame ID parity Enabled. **Note1:** This bit can be used for LIN master to sending header field (SENDH (UART_LINCTL[8])) = 1 and HSEL (UART_LINCTL[23:22]) = 10) or be used for enable LIN slave received frame ID parity checked. **Note2:** This bit is only use when the operation header transmitter is in HSEL (UART_LINCTL[23:22]) = 10.
[8]	**SENDH**	**LIN TX Send Header Enable Bit** The LIN TX header can be “break field” or “break and sync field” or “break, sync and frame ID field”, it is depend on setting HSEL (UART_LINCTL[23:22]). 0 = Send LIN TX header Disabled. 1 = Send LIN TX header Enabled. **Note1:** These registers are shadow registers of SENDH (UART_ALTCTL [7]); user can read/write it by setting SENDH (UART_ALTCTL [7]) or SENDH (UART_LINCTL [8]). **Note2:** When transmitter header field (it may be “break” or “break + sync” or “break + sync + frame ID” selected by HSEL (UART_LINCTL[23:22]) field) transfer operation finished, this bit will be cleared automatically.
[7:5]	**Reserved**	Reserved.
[4]	**MUTE**	**LIN Mute Mode Enable Bit** 0 = LIN mute mode Disabled. 1 = LIN mute mode Enabled. **Note:** The exit from mute mode condition and each control and interactions of this field are explained in 6.13.5.9 (LIN slave mode).
[3]	**SLVDUEN**	**LIN Slave Divider Update Method Enable Bit** 0 = UART_BAUD updated is written by software (if no automatic resynchronization update occurs at the same time). 1 = UART_BAUD is updated at the next received character. User must set the bit before checksum reception. **Note1:** This bit only valid when in LIN slave mode (SLVEN (UART_LINCTL[0]) = 1). **Note2:** This bit used for LIN Slave Automatic Resynchronization mode. (for Non-Automatic Resynchronization mode, this bit should be kept cleared)

		Note3: The control and interactions of this field are explained in 6.13.5.9 (Slave mode with automatic resynchronization).
[2]	**SLVAREN**	**LIN Slave Automatic Resynchronization Mode Enable Bit** 0 = LIN automatic resynchronization Disabled. 1 = LIN automatic resynchronization Enabled. **Note1:** This bit only valid when in LIN slave mode (SLVEN (UART_LINCTL[0]) = 1). **Note2:** When operation in Automatic Resynchronization mode, the baud rate setting must be mode2 (BAUDM1 (UART_BAUD [29]) and BAUDM0 (UART_BAUD [28]) must be 1). **Note3:** The control and interactions of this field are explained in 6.13.5.9(Slave mode with automatic resynchronization).
[1]	**SLVHDEN**	**LIN Slave Header Detection Enable Bit** 0 = LIN slave header detection Disabled. 1 = LIN slave header detection Enabled. **Note1:** This bit only valid when in LIN slave mode (SLVEN (UART_LINCTL[0]) = 1). **Note2:** In LIN function mode, when detect header field (break + sync + frame ID), SLVHDETF (UART_LINSTS [0]) flag will be asserted. If the LINIEN (UART_INTEN[8]) = 1, an interrupt will be generated.
[0]	**SLVEN**	**LIN Slave Mode Enable Bit** 0 = LIN slave mode Disabled. 1 = LIN slave mode Enabled.

UART LIN Status Register (UART_LINSTS) (Not Available in UART2/UART3 Channel)

Register	Offset	R/W	Description	Reset Value
UART_LINSTS x=0,1	UARTx_BA+0x38	R/W	UART LIN Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved						**BITEF**	**BRKDETF**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**SLVSYNCF**	**SLVIDPEF**	**SLVHEF**	**SLVHDETF**

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**BITEF**	**Bit Error Detect Status Flag (Read Only)** At TX transfer state, hardware will monitoring the bus state, if the input pin (SIN) state not equals to the output pin (SOUT) state, BITEF (UART_LINSTS[9]) will be set. When occur bit error, if the LINIEN (UART_INTEN[8]) = 1, an interrupt will be generated. **Note1:** This bit is read only, but it can be cleared by writing 1 to it. **Note2:** This bit is only valid when enable bit error detection function (BITERREN (UART_LINCTL [12]) = 1).
[8]	**BRKDETF**	**LIN Break Detection Flag (Read Only)** This bit is set by hardware when a break is detected and be cleared by writing 1 to it through software. 0 = LIN break not detected. 1 = LIN break detected. **Note1:** This bit is read only, but it can be cleared by writing 1 to it. **Note2:** This bit is only valid when LIN break detection function is enabled (BRKDETEN (UART_LINCTL[10]) =1).
[7:4]	**Reserved**	Reserved.
[3]	**SLVSYNCF**	**LIN Slave Sync Field (Read Only)** This bit indicates that the LIN sync field is being analyzed in Automatic Resynchronization mode. When the receiver header have some error been detect, user must reset the internal circuit to re-search new frame header by writing 1 to this bit. 0 = The current character is not at LIN sync state. 1 = The current character is at LIN sync state. **Note1:** This bit is only valid when in LIN Slave mode (SLVEN(UART_LINCTL[0]) = 1). **Note2:** This bit is read only, but it can be cleared by writing 1 to it. **Note3:** When writing 1 to it, hardware will reload the initial baud rate and re-search a new frame header.
[2]	**SLVIDPEF**	**LIN Slave ID Parity Error Flag**

		This bit is set by hardware when receipted frame ID parity is not correct. 0 = No active. 1 = Receipted frame ID parity is not correct. **Note1:** This bit is read only, but it can be cleared by writing 1 to it. **Note2:** This bit is only valid when in LIN slave mode (SLVEN (UART_LINCTL [0])= 1) and enable LIN frame ID parity check function IDPEN (UART_LINCTL [9]).
[1]	**SLVHEF**	**LIN Slave Header Error Flag (Read Only)** This bit is set by hardware when a LIN header error is detected in LIN slave mode and be cleared by writing 1 to it. The header errors include “break delimiter is too short (less than 0.5 bit time)”, “frame error in sync field or Identifier field”, “sync field data is not 0x55 in Non-Automatic Resynchronization mode”, “sync field deviation error with Automatic Resynchronization mode”, “sync field measure time-out with Automatic Resynchronization mode” and “LIN header reception time-out”. 0 = LIN header error not detected. 1 = LIN header error detected. **Note1:** This bit is read only, but it can be cleared by writing 1 to it. **Note2:** This bit is only valid when UART is operated in LIN slave mode (SLVEN (UART_LINCTL [0]) = 1) and enables LIN slave header detection function (SLVHDEN (UART_LINCTL [1])).
[0]	**SLVHDETF**	**LIN Slave Header Detection Flag (Read Only))** This bit is set by hardware when a LIN header is detected in LIN slave mode and be cleared by writing 1 to it. 0 = LIN header not detected. 1 = LIN header detected (break + sync + frame ID). **Note1:** This bit is read only, but it can be cleared by writing 1 to it. **Note2:** This bit is only valid when in LIN slave mode (SLVEN (UART_LINCTL [0]) = 1) and enable LIN slave header detection function (SLVHDEN (UART_LINCTL [1])). **Note3:** When enable ID parity check IDPEN (UART_LINCTL [9]), if hardware detect complete header (“break + sync + frame ID”), the SLVHDETF will be set whether the frame ID correct or not.

6.14 Smart Card Host Interface (SC)

6.14.1 Overview

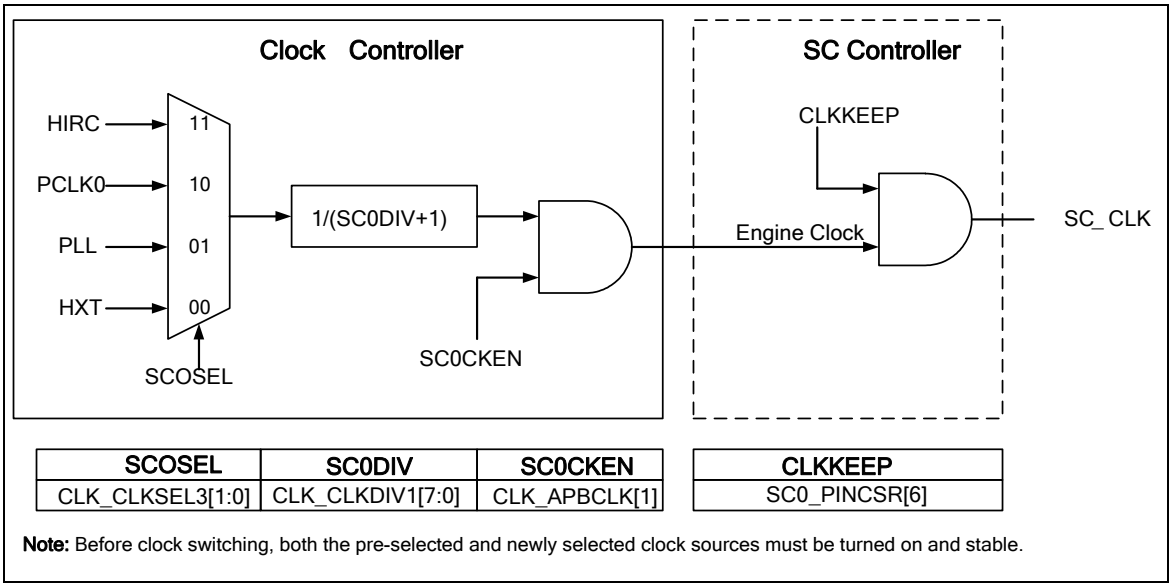
The Smart Card Interface controller (SC controller) is based on ISO/INTENC 7816-3 standard and fully compliant with PC/SC Specifications. It also provides status of card insertion/removal.

6.14.2 Features

- ISO-7816-3 T = 0, T = 1 compliant.
- EMV2000 compliant
- One ISO-7816-3 port
- Separates receive/transmit 4 byte entry FIFO for data payloads.
- Programmable transmission clock frequency.
- Programmable receiver buffer trigger level.
- Programmable guard time selection (11 ETU ~ 267 ETU).
- A 24-bit and two 8-bit timers for Answer to Request (ATR) and waiting times processing.
- Supports auto inverse convention function.
- Supports transmitter and receiver error retry and error number limiting function.
- Supports hardware activation sequence, hardware warm reset sequence and hardware
- deactivation sequence process.
- Supports hardware auto deactivation sequence when detected the card removal.
- Supports UART mode
  - Full duplex, asynchronous communications.
  - Separates receiving / transmitting 4 bytes entry FIFO for data payloads.
  - Supports programmable baud rate generator.
  - Supports programmable receiver buffer trigger level.
  - Programmable transmitting data delay time between the last stop bit leaving the TX-FIFO and the de-assertion by setting EGT (SC_EGT[7:0]).
  - Programmable even, odd or no parity bit generation and detection.
  - Programmable stop bit, 1- or 2- stop bit generation

6.14.3 Block Diagram

The SC clock control and block diagram are shown in Figure 6.14-1 and Figure 6.14-2. The PCLK should be higher or equal than the frequency of peripheral clock (SC_CLK).



SCOSEL	SC0DIV	SC0CKEN	CLKKEEP
CLK_CLKSEL3[1:0]	CLK_CLKDIV1[7:0]	CLK_APBCLK[1]	SC0_PINCSR[6]

Note: Before clock switching, both the pre-selected and newly selected clock sources must be turned on and stable.

Figure 6.14-1 SC Clock Control Diagram (4-bit Pre-scale Counter in Clock Controller)

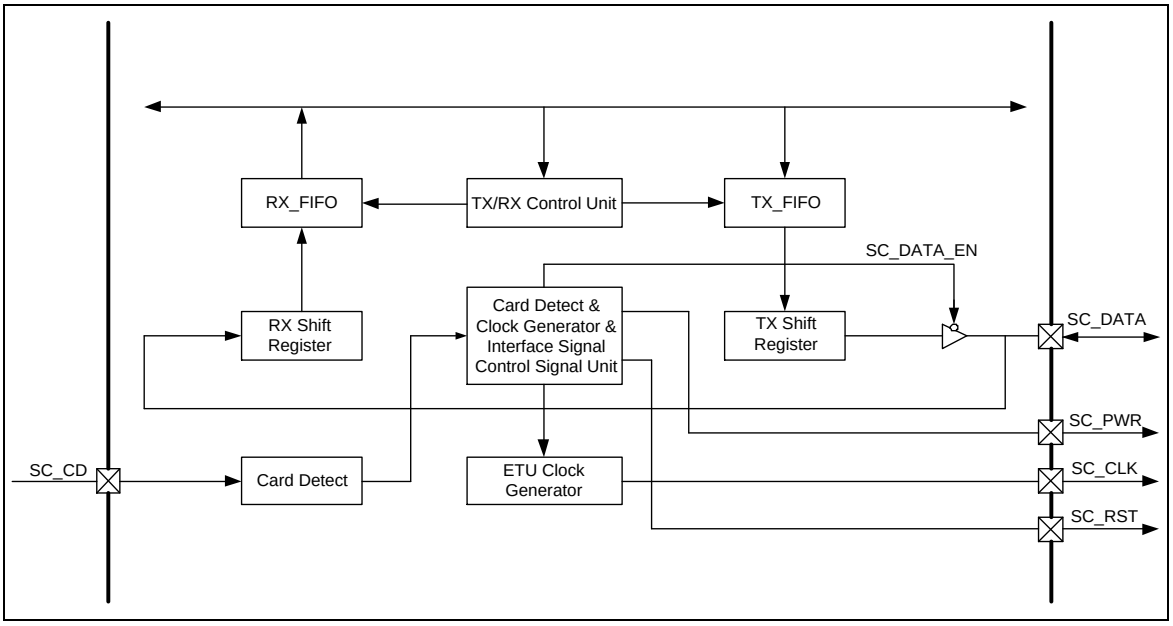


Figure 6.14-2 SC Controller Block Diagram

6.14.4 Basic Configuration

The SC function pins are configured in SYS_GPA_MFPL, SYS_GPB_MFPL, SYS_GPE_MFPL and GPE_MPFH Multiple Function Pin Registers (refer Register Map).

SC Host Controller Pin description is shown as follows:

Pin	Type	Description
SC_DATA	Bi-direction	SC Host Controller DATA
SC_CD	Input	SC Host Controller Card Detect
SC_PWR	Output	SC Host Controller Power ON/OFF Switch for RTC_CALMD
SC_CLK	Output	SC Host Controller Clock
SC_RST	Output	SC Host Controller Reset

Table 6-29 SC Host Controller Pin Description

UART Pin description is shown as follows:

Pin	Type	Description
SC_DATA	Input	UART Receive Data
SC_CLK	Output	UART Transmit Data

Table 6-30 UART Pin Description

6.14.5 Functional description

Basically, the smart card interface acts as a half-duplex asynchronous communication port and its data format is composed of ten consecutive bits which is shown in Figure 6.14-3.

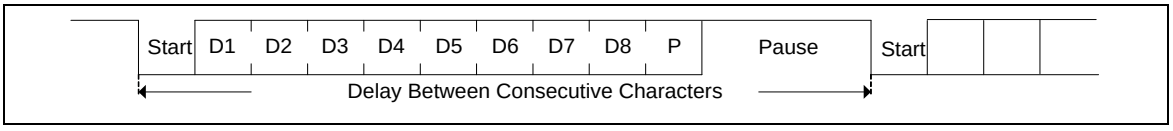


Figure 6.14-3 SC Data Character

The Smart Card Interface controller supports hardware activation, warm reset and deactivation sequence. The activation, Warm Reset and Deactivation and sequence are shown as follows.

6.14.5.1 Activation, Warm Reset and Deactivation Sequence

Activation

The activation sequence is shown in Figure 6.14-4:

1. Set SC_RST to low by programming RSTSTS (SC_PINCTL[18]) to '0'
2. Set SC_PWR at high level by programming PWRSTS (SC_PINCTL[18]) to '1' and SC_DAT at high level (reception mode) by programming DATSTS (SC_PINCTL[16]) to '1' .
3. Enable SC_CLK clock by programming CLKKEEP (SC_PINCTL[6]) to '1'.
4. De-assert SC_RST to high by programming RSTSTS (SC_PINCTL[18]) to '1'.

The activation sequence can be controlled in two ways. The procedure is shown as follows:

Software Timing Control:

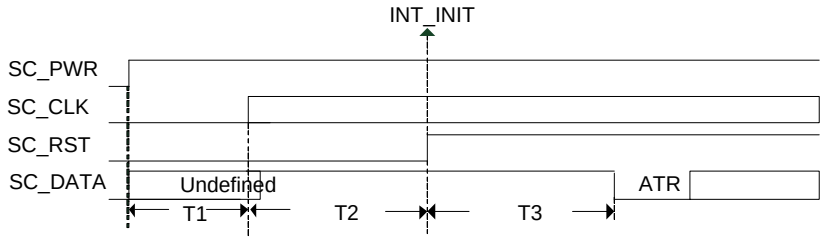
Set SC_PINCTL and SC_TMRx (x = 0, 1 ,2) to process the activation sequence. SC_PWR, SC_CLK, SC_RST and SC_DATA pin state can be programmed by SC_PINCTL. The programming method is shown in Activation description. The activation sequence timing can be controlled by setting SC_TMRx (x = 0, 1 ,2). This programming procedure provides user has a flexible timing setting for activation sequence.

Hardware Timing Control:

Set ACTEN (SC_ALTCTL[3]) to '1' and the interface will perform the activation sequence by hardware. The SC_PWR to SC_CLK start (T1) and SC_CLK_start to SC_RST assert (T2) can be selected by programming INITSEL (SC_ALTCTL[9:8]). This programming procedure provides user has a simple setting for activation sequence.

Following is the activation control sequence generated by hardware:

1. Set activation timing by setting INITSEL (SC_ALTCTL[9:8]).
2. TMR0 can be selected by setting TMRSEL (SC_CTL[14:13]) to '01', '10' or '11'.
3. Set operation mode OPMODE (SC_TMRCTL0[27:24]) to '0011' and give an Answer to Request (ATR) value by setting CNT (SC_TMRCTL0 [23:0]) register.
4. When hardware de-asserts SC_RST to high, hardware will generator an interrupt INITIF (SC_INTSTS[8]) to CPU at the same time INITIEN(SC_INTEN[8]) = "1".
5. If the TMR0 decreases the counter to '0' (start from SC_RST de-assert) and the card does not response ATR before that time, hardware will generate interrupt TMR0IF (SC_INTSTS[3]) to CPU.



Time	Comment
T1	SC_PWR to SC_CLK Start
T2	SC_CLK Start to SC_RST Assert
T3	SC_CLK Start to ATR Appear

INIT_SEL (SC_ ALTCT[9:8])	T1	T2
00	85	489
01	133	537
10	165	569
11	165	42060

Unit: SC Clock

Note : The values are measured by chip I/O pin and the real value will depend on system design

Figure 6.14-4 SC Activation Sequence

Warm Reset

The warm reset sequence is shown in Figure 6.14-5 :

1. Set SC_RST to low by programming RSTSTS (SC_PINCTL[18]) to '0'
2. Set SC_DAT to high by programming DATSTS (SC_PINCTL[16]) to '1' .
3. Set SC_RST to high by programming RSTSTS (SC_PINCTL[18]) to '1'.

The warm reset sequence can be controlled in two ways. The procedure is shown as follows.

Software Timing Control:

Set SC_PINCTL and SC_TMRx (x = 0, 1 ,2) to process the warm reset sequence. SC_RST and SC_DATA pin state can be programmed by SC_PINCTL. The warm reset sequence

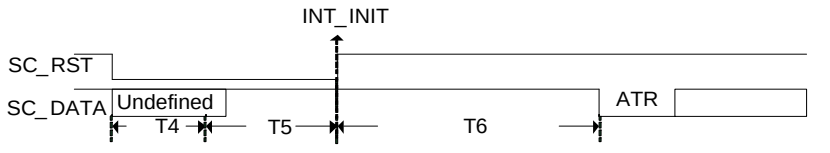
timing can be controlled by setting SC_TMRx (x = 0, 1 ,2). This programming procedure provides user has a flexible timing setting for warm reset sequence.

Hardware Timing Control:

Set WARSTEN (SC_ALTCTL[4]) to '1' and the interface will perform the warm reset sequence by hardware. The SC_RST to SC_DATA reception mode (T4) and SC_DATA reception mode to SC_RST assert (T5) can be selected by programming INITSEL (SC_ALTCTL[9:8]). This programming procedure provides user has a simple setting for warm reset sequence.

Following is the warm reset control sequence by hardware:

1. Set warm reset timing by setting INITSEL (SC_ALTCTL[9:8]).
2. Select TMR0 by setting TMRSEL (SC_CTL[14:13]) register (TMRSEL can be set to '01', '10', or '11').
3. Set operation mode OPMODE (SC_TMRCTL0[27:24]) to '0011' and give an Answer to Request value by setting CNT (SC_TMRCTL0[23:0]) register.
4. Set CNTEN0 (SC_ALTCTL[5]) and WARSTEN (SC_ALTCTL[4]) to start counting.
5. When hardware de-asserts SC_RST to high, hardware will generate an interrupt INITIF (SC_INTSTS[8]) to CPU at the same time (INITIEN(SC_INTEN[8] = '1')).
6. If the TMR0 decreases the counter to '0' (start from SC_RST) and the card does not response ATR before that time, hardware will generate interrupt TMR0IF (SC_INTSTS[3]) to CPU.



Time	Comment
T4	SC_RST to SC_DATA Reception Mode
T5	SC_DATA Reception Mode to SC_RST Assert
T6	SC_CLK Start to ATR Appear

INIT_SEL (SC_ALTCTL[9:8])	T4	T5
00	81	483
01	129	531
10	161	563
11	161	42106

Unit: SC Clock

Note: This value is measured by chip IO pin and the real value will depend on system design

Figure 6.14-5 SC Warm Reset Sequence

Deactivation

The deactivation sequence is shown in Figure 6.14-6.

1. Set SC_RST to low by programming RSTSTS (SC_PINCTL[18]) to '0'.
2. Stop SC_CLK by programming CLKKEEP (SC_PINCTL[6]) to '0'.
3. Set SC_DATA to low by programming DATSTS (SC_PINCTL[16]) to '0'.
4. Deactivate SC_PWR by programming PWRSTS (SC_PINCTL[18]) to '0'.

The deactivation sequence can be controlled in two ways. The procedure is shown as follows.

Software Timing Control:

Set SC_PINCTL and SC_TMRCTL0 to process the deactivation sequence. SC_PWR, SC_CLK, SC_RST and SC_DATA pin state can be programmed by SC_PINCTL. The deactivation sequence timing can be controlled by setting SC_TMRCTL0. This programming procedure

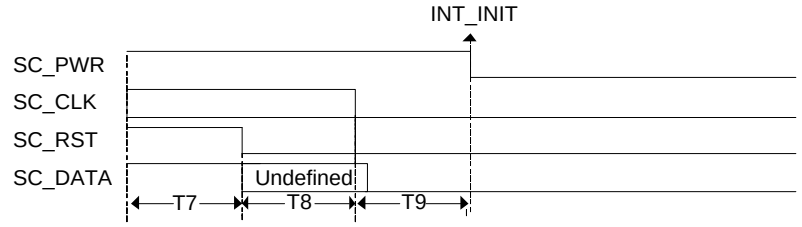
provides user has a flexible timing setting for deactivation sequence.

Hardware Timing Control:

DACTEN (SC_ALTCTL[2]) to '1' and the interface will perform the deactivation sequence by hardware. The Deactivation Trigger to SC_RST low (T7), SMC_RST low to SC_CLK (T8) and stop SC_CLK to stop SC_PWR (T9) time can be selected by programming INITSEL (SC_ALTCTL[9:8]). This programming procedure provides user has a simple setting for deactivation sequence.

The SC controller also supports auto deactivation sequence when the card removal detection is enabled by setting ADAC_CDEN (SC_ALTCTL[11]).

When hardware de-asserts SC_PWR to low, the controller will generate an interrupt to CPU at the same time (if INITIEN= 1(SC_IER[8])



Time	Comment
T7	Deactivation Trigger to SC_RST Low
T8	SMC_RST Low to Stop SC_CLK
T9	Stop SC_CLK to Stop SC_PWR

INITSEL (SC_ALTCTL[9:8])	T7	T8	T9
00	97	83	87
01	145	131	135
10	177	163	167
11	177	163	167

Unit: SC Clock

Figure 6.14-6 SC Deactivation Sequence

The Program Sequence Flow is shown as follows:

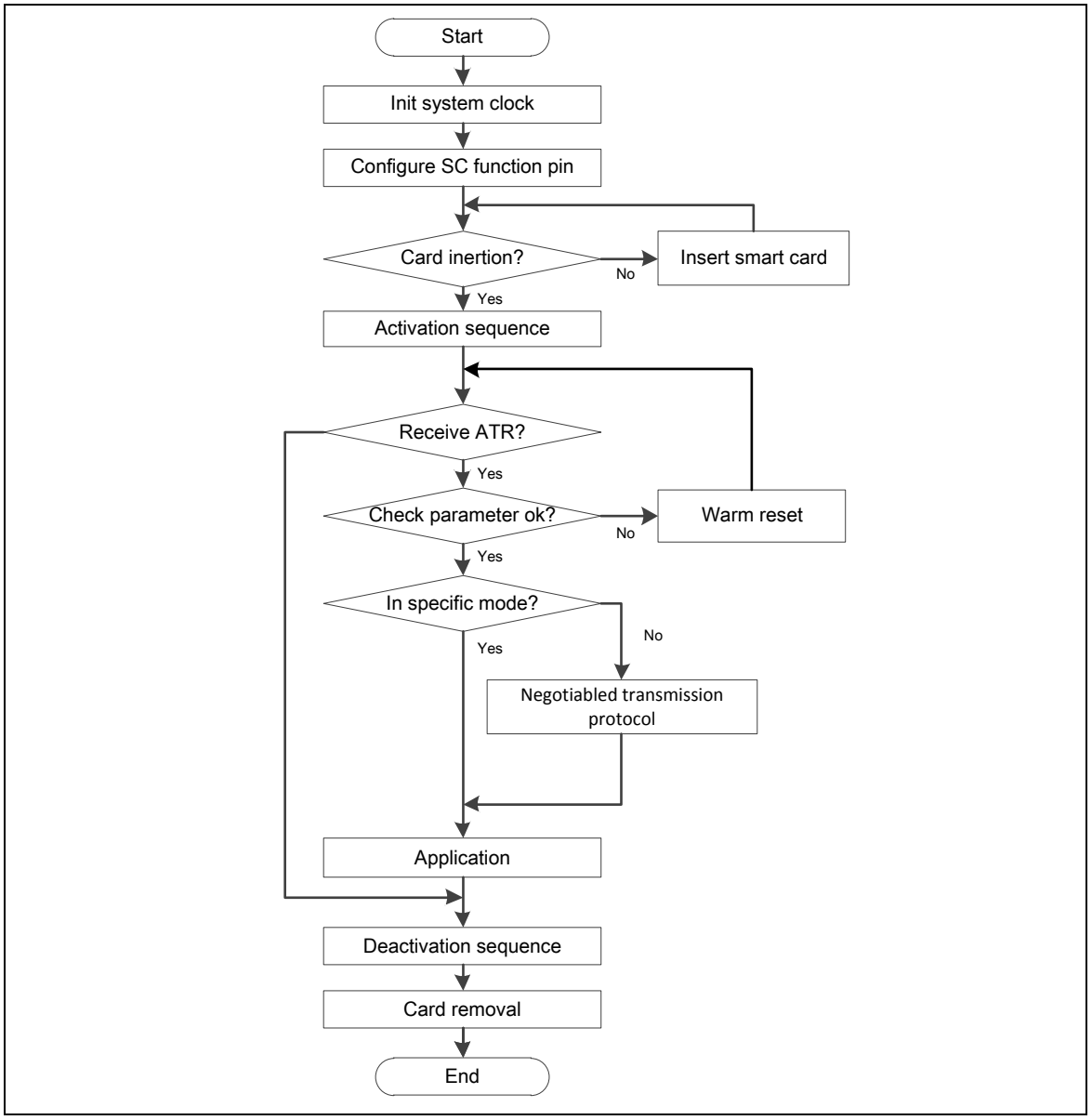


Figure 6.14-7 Basic Operation Flow

6.14.5.2 Initial Character TS

According to 7816-3, the initial character TS has two possible patterns shown in Figure 6.14-8. If the TS pattern is 1100_0000, it is inverse convention. When decoded by inverse convention, the conveyed byte is equal to 0x3F. If the TS pattern is 1101_1100, it is direct convention. When decoded by direct convention, the conveyed byte is equal to 0x3B. Software can set AUTOCEN (SC_CTL[3]) and then the operating convention will be decided by hardware. Software can also set the CONSEL (SC_CTL[5:4]) register (set to '00' or '11') to change the operating convention after SC received TS of answer to request (ATR).

If auto convention function is enabled by setting AUTOCEN (SC_CTL[3]) register, the setting step must be done before Answer to Request state and the first data must be 0x3B or 0x3F. After hardware received first data and stored it at buffer, the hardware will decided the convention and change the CONSEL (SC_CTL[5:4]) register automatically. If the first data is neither 0x3B nor 0x3F, the hardware will generate an interrupt (if ACERRIEN (SC_INTEN [10]) = '1') to CPU.

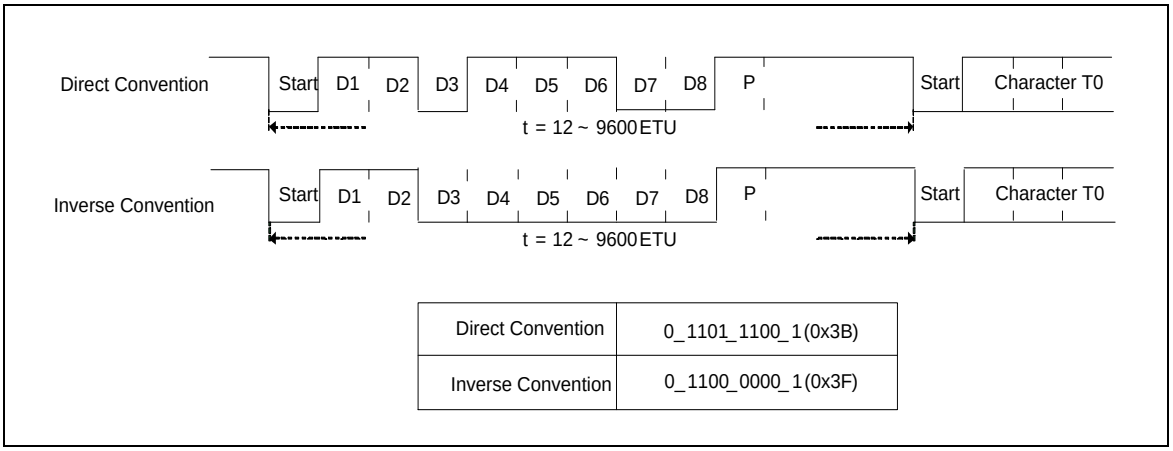


Figure 6.14-8 Initial Character TS

6.14.5.3 Error Signal and Character Repetition

According to ISO7816-3 T=0 mode description, as shown in Figure 6.14-9, if the receiver receives a wrong parity bit, it will pull the SC_DAT to low by 1.5 bit period to inform the transmitter parity error. Then the transmitter will retransmit the character. The SC interface controller supports hardware error detection function in receiver and supports hardware re-transmit function in transmitter. Software can enable re-transmit function by setting TXRTYEN(SC_CTL[23]). Software can also define the retry (re-transmit) number limitation in TXRTY(SC_CTL[22:20]). The re-transmit number is up to TXRTY +1 and if the re-transmit number is equal to TXRTY +1, TXOVERR flag will be set by hardware and if TERRIEN (SC_INTEN [2]), SC controller will generate a transfer error interrupt to CPU. Software can also define the received retry number limitation in RXRTY(SC_CTL[18:16]) register. The receiver retry number is up to RXRTY +1, if the number of received errors by receiver is equal to RXRTY +1, receiver will receive this error data to buffer and RXOVERR flag will be set by hardware and if TERRIEN(SC_INTEN[2]), SC controller will generate a transfer error interrupt to CPU.

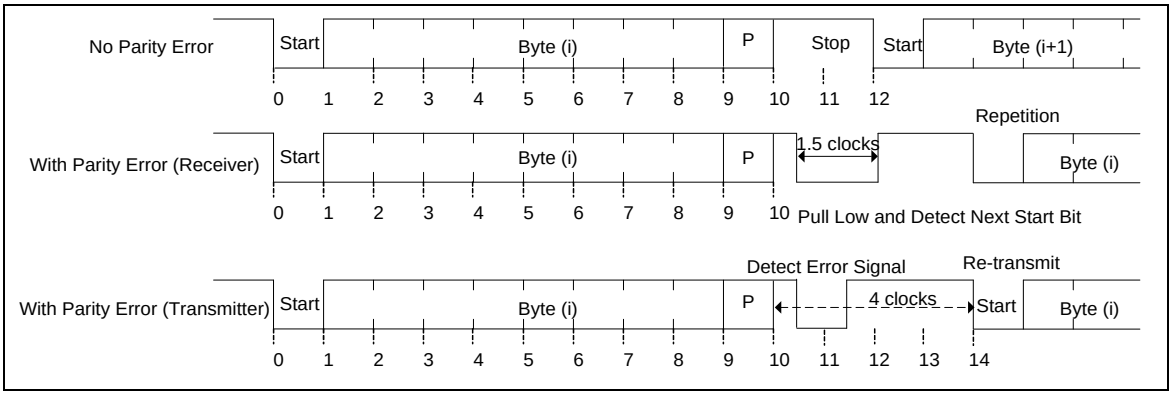


Figure 6.14-9 SC Error Signal

6.14.5.4 Internal Time-out Counter

The smart card interface includes a 24-bit time-out counter and two 8 bit time-out counters. These counters help the controller in processing different real-time interval. Each counter can be set to start counting once the trigger enable bit has been written or a START bit has been detected.

The following is the programming flow:

Enable counter by setting TMRSEL (SC_CTL[14:13]). Select operation mode OPMODE (SC_TMRx[27:24]) and give a count value CNT(SC_TMRx[23:0]) by setting SC_TMRx register. Set CNTEN0 (SC_ALTCTL [5]), CNTEN1 (SC_ALTCTL [6] or CNTEN2 (SC_ALTCTL [7]) is to

start counting.

The SC_TMRCTL0, SC_TMRCTL1 and SC_TMRCTL2 timer operation mode are listed in Table 6-31 Timer2/Timer1/Timer0 Operation Mode.

Note: Only SC_TMRCTL0 supports mode 0011.

OPMODE(SCTMRCTLx[27:24]) (X=0 ~2)	Operation Description	
0000	The down counter started when TMRx_SEN (SC_ALTCTL[7:5]) enabled and ended when counter time-out. The time-out value will be CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0])+1	
	Start	Start counting when TMRx_SEN (SC_ALTCTL[7:5]) enabled
	End	When the down counter equals to 0, hardware will set TMRx_IS(SC_INTSTS[5:3]) and clear TMRx_SEN (SC_ALTCTL[7:5]) automatically.
0001	The down counter started when the first START bit (reception or transmission) detected and ended when counter time-out. The time-out value will be CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0])+1.	
	Start	Start counting when the first START bit (reception or transmission) detected after TMRx_SEN (SC_ALTCTL[7:5]) set to 1.
	End	When the down counter equals to 0, hardware will set TMRx_IS(SC_INTSTS[5:3]) and clear TMRx_SEN (SC_ALTCTL[7:5]) automatically.
0010	The down counter started when the first START bit (reception) detected and ended when counter time-out . The time-out value will be CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0])+1.	
	Start	Start counting when the first START bit (reception) detected bit after TMRx_SEN (SC_ALTCTL[7:5]) set to 1.
	End	When the down counter equals to 0, hardware will set TMRx_IS(SC_INTSTS[5:3]) and clear TMRx_SEN (SC_ALTCTL[7:5]) automatically.
0011	The down counter is only used for hardware activation, warm reset sequence to measure ATR timing. The timing starts when SC_RST de-assertion and ends when ATR response received or time-out. If the counter decreases to 0 before ATR response received, hardware will generate an interrupt to CPU. The time-out value will be CNT (SC_TMRCTL0[23:0])+1.	
	Start	Start counting when SC_RST de-assertion after CNTEN0 (SC_ALTCTL[5]) set to 1. It is used for hardware activation, warm reset mode.
	End	When the down counter equals to 0 before ATR response received, hardware will set TMR0IF and clear CNTEN0 (SC_ALTCTL[5]) automatically. When ATR received and down counter does not equal to 0, hardware will clear CNTEN0 (SC_ALTCTL[5]) automatically.
0100	Same as 0000, but when the down counter equals to 0, hardware will set TMRx_IS(SC_INTSTS[5:3]) and counter will re-load the CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) value and re-count until software clears TMRx_SEN (SC_ALTCTL[7:5]). When TMRx_ATV (SC_ALTCTL[15:13]) =1, software can change CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) value at any time. When the down counter equals to 0, counter will reload the new value of CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) and re-count. The time-out value will be CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0])+1.	
0101	Same as 0001, but when the down counter equals to 0, hardware will set TMRx_IS(SC_INTSTS[5:3]) and counter will re-load the CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) value. When the next START bit is detected, counter will re-count until software clears TMRx_ATV (SC_ALTCTL[15:13]). When TMRx_ATV (SC_ALTCTL[15:13]) =1 software can change CNT (SC_TMRCTL0[23:0], SC_TMRCTL0[7:0], SC_TMRCTL0[7:0]) value at any time. When the down counter equal to 0, it will reload the	

	new value of CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) and re-counting. The time-out value will be CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0])+1.	
0110	Same as 0010, but when the down counter equals to 0, it will set TMRx_IS(SC_INTSTS[5:3]) and counter will re-load the CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) value. When the next START bit is detected, counter will re-count until software clears TMRx_SEN (SC_ALTCTL[7:5]). When TMRx_ATV (SC_ALTCTL[15:13]) =1, software can change CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) value at any time. When the down counter equals to 0, counter will reload the new value of CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) and re-count. The time-out value will be CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0])+1.	
0111	The down counter started when the first START bit (reception or transmission) detected and ended when software clears TMRx_SEN (SC_ALTCTL[7:5]) bit. If next START bit detected, counter will reload the new value of CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) and re-counting. If the counter decreases to 0 before the next START bit detected, hardware will generate an interrupt to CPU. The time-out value will be CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0])+1.	
	Start	Start counting when the first START bit detected after TMRx_SEN (SC_ALTCTL[7:5]) set to 1.
	End	Stop counting after TMRx_SEN (SC_ALTCTL[7:5]) set to 0.
1000	The up counter starts when TMRx_SEN (SC_ALTCTL[7:5]) enabled and ends when TMRx_SEN (SC_ALTCTL[7:5]) disabled. This count value will be stored in CNTx(SC_TMRDAT0 [23:0], SC_TMRDAT1_2[7:0], SC_TMRDAT1_2[15:8). In this mode, hardware cannot generate any interrupt to CPU. The real count value will be CNTx(SC_TMRDAT0 [23:0], SC_TMRDAT1_2[7:0], SC_TMRDAT1_2[15:8]) +1.	
	Start	Start counting after TMRx_SEN (SC_ALTCTL[7:5]) set to 1, and the start count value is 0 (hardware will ignore CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) value).
	End	Stop counting after TMRx_SEN (SC_ALTCTL[7:5]) set to 0 and the value stored to CNTx(SC_TMRDAT0 [23:0], SC_TMRDAT1_2[7:0], SC_TMRDAT1_2[15:8) register.
1111	Down counter starts when software set TMRx_SEN (SC_ALTCTL[7:5]) bit or any START bit been detected and ends when software clears TMRx_SEN (SC_ALTCTL[7:5]) bit. If next START bit detected, counter will reload the new value of CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0]) and re-counting. If the counter decreases to “0” before the next START bit be detected, hardware will generate an interrupt to CPU. The time-out value will be CNT (SC_TMRCTL0[23:0], SC_TMRCTL1[7:0], SC_TMRCTL2[7:0])+1.	
	Start	Start count when the TMRx_SEN (SC_ALTCTL[7:5]) set to “1” or any START bit (TMRx_SEN (SC_ALTCTL[7:5]) must be set) be detected
	End	Stop count after TMRx_SEN (SC_ALTCTL[7:5]) set to “0”.

Table 6-31 Timer2/Timer1/Timer0 Operation Mode

6.14.5.5 Block Guard Time and Extended Guarg Time

Block guard time means the minimum bit length between the leading edges of two consecutive characters between different transfer directions. This field indicates the counter for the bit length of block guard time. According to ISO7816-3, in T = 0 mode, software must fill 15 (real block guard time = 16.5) to this field; in T = 1 mode, software must fill 21 (real block guard time = 22.5) to it.

In transmit direction, the smart card sends data to smart card host controller, first. After the period is greater than BGT (SC_CTL[12:8]), the smart card host controller begin to send the data.

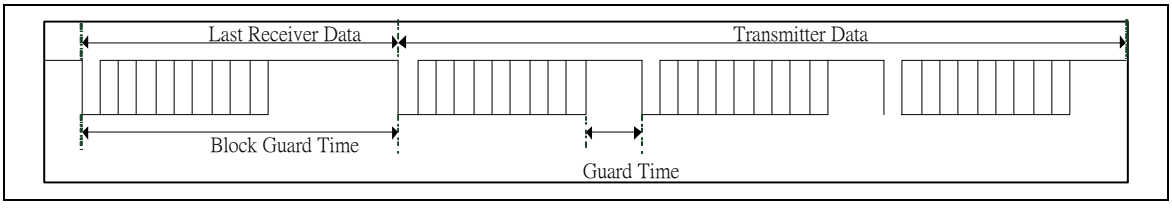


Figure 6.14-10 Transmit Direction Block Guard Time Operation

In receive direction, the smart card host controller sends data to smart card, first. If the smart card sends data to smart card host controller at the time which is less than BGT (SC_CTL[12:8]),the block guard time interrupt BGTIF (SC_INTSTS[6]) is generated when RXBGTEN (SC_ALTCTL[12]) is enabled.

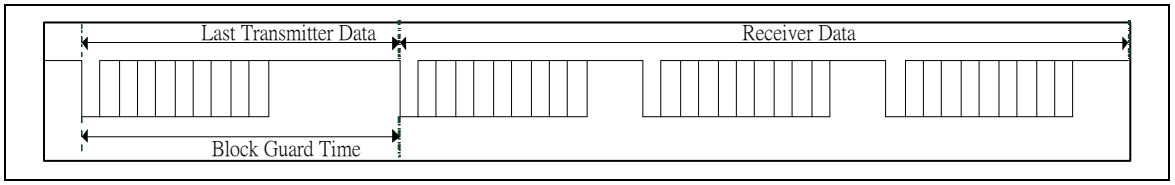


Figure 6.14-11 Receive Direction Block Guard Time Operation

Extended Guard Time is two ETU plus EGT (SC_EGT[7:0]), the format is shown as follows:

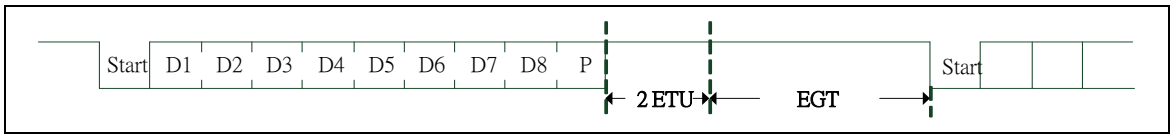


Figure 6.14-12 Extended Guard Time Operation

6.14.5.6 UART Mode

When the UARTEN (SC_UARTCTL[0]) bit set, the Smart Card Interface controller can also be used as base UART function. The following is the program example for UART mode.

Program example:

1. Set UARTEN (SC_UARTCTL[0]) bit to enter UART mode.
2. Do software reset by setting RXRST (SC_ALTCTL[1]) and TXRST(SC_ALTCTL[0]) bit to ensure that all state machine return idle state.
3. Fill “0” to CONSEL (SC_CTL[5:4]) and AUTOCEN (SC_CTL[3]) field. (In UART mode, those fields must be “0”)
4. Select the UART baud rate by setting ETURDIV (SC_ETUCTL[11:0]) fields. For example, if smartcard module clock is 12 MHZ and target baud rate is 115200bps, ETURDIV should fill with (12000000 / 115200 - 1).
5. Select the data format include data length (by setting WLS (SC_UARTCTL [5:4]), parity format (by setting OPE(SC_UARTCTL[7]) and PBOFF(SC_UARTCTL[6])) and stop bit length (by setting NSB(SC_CTL[15] or EGT(SC_EGT[7:0])).
6. Select the receiver buffer trigger level by setting RXTRGLV (SC_CTL[7:6]) field and select the receiver buffer time-out value by setting RFTM (SC_RXTOUT[8:0]) field.
7. Write the SC_DAT (SC_DAT[7:0]) (TX) register or read the SC_DAT (SC_DAT[7:0]) (RX) register can perform UART function.

6.14.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
SC Base Address: **SC_BA = 0x4009_0000**				
SC_DAT	SC_BA+0x00	R/W	SC Receiving/Transmit Holding Buffer Register.	0xXXXX_XXXX
SC_CTL	SC_BA+0x04	R/W	SC Control Register.	0x0000_0000
SC_ALTCTL	SC_BA+0x08	R/W	SC Alternate Control Register.	0x0000_0000
SC_EGT	SC_BA+0x0C	R/W	SC Extend Guard Time Register.	0x0000_0000
SC_RXTOUT	SC_BA+0x10	R/W	SC Receive buffer Time-out Register.	0x0000_0000
SC_ETUCTL	SC_BA+0x14	R/W	SC ETU Control Register.	0x0000_0173
SC_INTEN	SC_BA+0x18	R/W	SC Interrupt Enable Control Register.	0x0000_0000
SC_INTSTS	SC_BA+0x1C	R/W	SC Interrupt Status Register.	0x0000_0002
SC_STATUS	SC_BA+0x20	R/W	SC Status Register.	0x0000_0202
SC_PINCTL	SC_BA+0x24	R/W	SC Pin Control State Register.	0x0000_00x0
SC_TMRCTL0	SC_BA+0x28	R/W	SC Internal Timer Control Register 0.	0x0000_0000
SC_TMRCTL1	SC_BA+0x2C	R/W	SC Internal Timer Control Register 1.	0x0000_0000
SC_TMRCTL2	SC_BA+0x30	R/W	SC Internal Timer Control Register 2.	0x0000_0000
SC_UARTCTL	SC_BA + 0x34	R/W	SC UART Mode Control Register.	0x0000_0000
SC_TMRDAT0	SC_BA+0x38	R	SC Timer Current Data Register A.	0x0000_07FF
SC_TMRDAT1_2	SC_BA+0x3C	R	SC Timer Current Data Register B.	0x0000_7F7F

6.14.7 Register Description

SC Receiving Buffer Register (SC_DAT)

Register	Offset	R/W	Description	Reset Value
SC_DAT	SC_BA+0x00	R/W	SC Receiving/Transmit Holding Buffer Register.	0xXXXX_XXXX

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
DAT							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**DAT**	**Receiving/ Transmit Holding Buffer** Write Operation: By writing data to DAT, the SC will send out an 8-bit data. **Note:** If SCEN(SC_CTL[0]) is not enabled, DAT cannot be programmed. Read Operation: By reading DAT, the SC will return an 8-bit received data.

SC Control Register (SC_CTL)

Register	Offset	R/W	Description	Reset Value
SC_CTL	SC_BA+0x04	R/W	SC Control Register.	0x0000_0000

31	30	29	28	27	26	25	24
DBGOFF	SYNC	Reserved			CDLV	CDDBSEL	
23	22	21	20	19	18	17	16
TXRTYEN	TXRTY			RXRTYEN	RXRTY		
15	14	13	12	11	10	9	8
NSB	TMRSEL		BGT				
7	6	5	4	3	2	1	0
RXTRGLV		CONSEL		AUTOCEN	TXOFF	RXOFF	SCEN

Bits	Description	
[31]	**DBGOFF**	**ICE Debug Mode Acknowledge Enable Bit** 0 = When DBGACK is high, the internal counter will be hold. 1 = No matter DBGACK is high or low, the internal counter will not be hold.
[30]	**SYNC**	**SYNC Flag Indicator** Due to synchronization, software should check this bit before writing a new value to RXRTY and TXRTY. 0 = synchronizing is completion, user can write new data to RXRTY and TXRTY. 1 = Last value is synchronizing. **Note:** This bit is read only.
[29:27]	**Reserved**	Reserved.
[26]	**CDLV**	**Card Detect Level** 0 = When hardware detects the card detect pin (SC_CD) from high to low, it indicates a card is detected. 1 = When hardware detects the card detect pin from low to high, it indicates a card is detected. **Note:** Software must select card detect level before Smart Card engine enabled.
[25:24]	**CDDBSEL**	**Card Detect De-bounce Selection** This field indicates the card detect de-bounce selection. 00 = De-bounce sample card insert once per 384 (128 * 3) peripheral clocks and de-bounce sample card removal once per 128 peripheral clocks. 01 = De-bounce sample card insert once per 192 (64 * 3) peripheral clocks and de-bounce sample card removal once per 64 peripheral clocks. 10 = De-bounce sample card insert once per 96 (32 * 3) peripheral clocks and de-bounce sample card removal once per 32 peripheral clocks. 11 = De-bounce sample card insert once per 48 (16 * 3) peripheral clocks and de-bounce sample card removal once per 16 peripheral clocks.
[23]	**TXRTYEN**	**TX Error Retry Enable Bit** This bit enables transmitter retry function when parity error has occurred. 0 = TX error retry function Disabled.

		1 = TX error retry function Enabled.
[22:20]	**TXRTY**	**TX Error Retry Count Number** This field indicates the maximum number of transmitter retries that are allowed when parity error has occurred. **Note1:** The real retry number is TXRTY + 1, so 8 is the maximum retry number. **Note2:** This field cannot be changed when TXRTYEN enabled. The change flow is to disable TXRTYEN first and then fill in new retry value.
[19]	**RXRTYEN**	**RX Error Retry Enable Bit** This bit enables receiver retry function when parity error has occurred. 0 = RX error retry function Disabled. 1 = RX error retry function Enabled. **Note:** Software must fill in the RXRTY value before enabling this bit.
[18:16]	**RXRTY**	**RX Error Retry Count Number** This field indicates the maximum number of receiver retries that are allowed when parity error has occurred **Note1:** The real retry number is RXRTY + 1, so 8 is the maximum retry number. **Note2:** This field cannot be changed when RXRTYEN enabled. The change flow is to disable RXRTYEN first and then fill in new retry value.
[15]	**NSB**	**Stop Bit Length** This field indicates the length of stop bit. 0 = The stop bit length is 2 ETU. 1= The stop bit length is 1 ETU. **Note:** The default stop bit length is 2. SMC and UART adopts NSB to program the stop bit length
[14:13]	**TMRSEL**	**Timer Selection** 00 = All internal timer function Disabled. 01 = Internal 24 bit timer Enabled. Software can configure it by setting SC_TMRCTL0 [23:0]. SC_TMRCTL1 and SC_TMRCTL2 will be ignored in this mode. 10 = internal 24 bit timer and 8 bit internal timer Enabled. Software can configure the 24 bit timer by setting SC_TMRCTL0 [23:0] and configure the 8 bit timer by setting SC_TMRCTL1[7:0]. SC_TMRCTL2 will be ignored in this mode. 11 = Internal 24 bit timer and two 8 bit timers Enabled. Software can configure them by setting SC_TMRCTL0 [23:0], SC_TMRCTL1 [7:0] and SC_TMRCTL2 [7:0].
[12:8]	**BGT**	**Block Guard Time (BGT)** Block guard time means the minimum bit length between the leading edges of two consecutive characters between different transfer directions. This field indicates the counter for the bit length of block guard time. According to ISO7816-3, in T = 0 mode, software must fill 15 (real block guard time = 16.5) to this field; in T = 1 mode, software must fill 21 (real block guard time = 22.5) to it. **Note:** The real block guard time is BGT + 1.
[7:6]	**RXTRGLV**	**Rx Buffer Trigger Level** When the number of bytes in the receiving buffer equals the RXTRGLV, the RDAIF will be set (if SC_INTEN [RDAIEN] is enabled, an interrupt will be generated). 00 = INTR_RDA Trigger Level with 1 Byte. 01 = INTR_RDA Trigger Level with 2 Bytes. 10 = INTR_RDA Trigger Level with 3 Bytes. 11 = Reserved.
[5:4]	**CONSEL**	**Convention Selection** 00 = Direct convention.

		01 = Reserved. 10 = Reserved. 11 = Inverse convention. **Note:** If AUTOCEN(SC_CTL[3]) enabled, this fields are ignored.
[3]	**AUTOCEN**	**Auto Convention Enable Bit** 0 = Auto-convention Disabled. 1 = Auto-convention Enabled. When hardware receives TS in answer to reset state and the TS is direct convention, CONSEL(SC_CTL[5:4]) will be set to 00 automatically, otherwise if the TS is inverse convention, and CONSEL (SC_CTL[5:4]) will be set to 11. If software enables auto convention function, the setting step must be done before Answer to Reset state and the first data must be 0x3B or 0x3F. After hardware received first data and stored it at buffer, hardware will decided the convention and change the CONSEL (SC_CTL[5:4]) bits automatically. If the first data is not 0x3B or 0x3F, hardware will generate an interrupt INT_ACON_ERR (if ACERRIEN (SC_INTEN[10]) = 1 to CPU.
[2]	**TXOFF**	**TX Transition Disable Control** 0 = The transceiver Enabled. 1 = The transceiver Disabled.
[1]	**RXOFF**	**RX Transition Disable Control** 0 = The receiver Enabled. 1 = The receiver Disabled. **Note:** If AUTOCEN (SC_CTL[3])is enabled, these fields must be ignored.
[0]	**SCEN**	**SC Engine Enable Bit** Set this bit to 1 to enable SC operation. If this bit is cleared, SC will force all transition to IDLE state.

SC Alternate Control Register (SC_ALTCTL)

Register	Offset	R/W	Description	Reset Value
SC_ALTCTL	SC_BA+0x08	R/W	SC Alternate Control Register.	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
ACTSTS2	ACTSTS1	ACTSTS0	RXBGTEN	ADACEN	Reserved	INITSEL	
7	6	5	4	3	2	1	0
CNTEN2	CNTEN1	CNTEN0	WARSTEN	ACTEN	DACTEN	RXRST	TXRST

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**ACTSTS2**	**Internal Timer2 Active State (Read Only)** This bit indicates the timer counter status of timer2. 0 = Timer2 is not active. 1 = Timer2 is active.
[14]	**ACTSTS1**	**Internal Timer1 Active State (Read Only)** This bit indicates the timer counter status of timer1. 0 = Timer1 is not active. 1 = Timer1 is active.
[13]	**ACTSTS0**	**Internal Timer0 Active State (Read Only)** This bit indicates the timer counter status of timer0. 0 = Timer0 is not active. 1 = Timer0 is active.
[12]	**RXBGTEN**	**Receiver Block Guard Time Function Enable Bit** 0 = Receiver block guard time function Disabled. 1 = Receiver block guard time function Enabled.
[11]	**ADACEN**	**Auto Deactivation When Card Removal** 0 = Auto deactivation Disabled when hardware detected the card removal. 1 = Auto deactivation Enabled when hardware detected the card removal. **Note:** When the card is removed, hardware will stop any process and then do deactivation sequence (if this bit is set). If this process completes, hardware will generate an interrupt INITIF to CPU.
[10]	**Reserved**	Reserved.
[9:8]	**INITSEL**	**Initial Timing Selection** This fields indicates the timing of hardware initial state (activation or warm-reset or deactivation). Unit: SC clock

		Activation: refer to SC Activation Sequence in Figure 6.14-4 Warm-reset: refer to Warm-Reset Sequence in Figure 6.14-5 Deactivation: refer to Deactivation Sequence in Figure 6.14-6
[7]	**CNTEN2**	**Internal Timer2 Start Enable Bit** This bit enables Timer 2 to start counting. Software can fill 0 to stop it and set 1 to reload and count. 0 = Stops counting. 1 = Start counting. **Note1:** This field is used for internal 8 bit timer when TMRSEL(SC_CTL[14:13]) = 11. Don't filled CNTEN2 when TMRSEL(SC_CTL[14:13]) = 00 or TMRSEL(SC_CTL[14:13]) = 01 or TMRSEL(SC_CTL[14:13]) = 10. **Note2:** If the operation mode is not in auto-reload mode (SC_TMRCTL2[26] = 0), this bit will be auto-cleared by hardware. **Note3:** This field will be cleared by TXRST(SC_ALTCTL[0]) and RXRST(SC_ALTCTL[1]). So don't fill this bit, TXRST(SC_ALTCTL[0]), and RXRST(SC_ALTCTL[1]) at the same time. **Note4:** If SCEN(SC_CTL[0]) is not enabled, this filed cannot be programmed.
[6]	**CNTEN1**	**Internal Timer1 Start Enable Bit** This bit enables Timer 1 to start counting. Software can fill 0 to stop it and set 1 to reload and count. 0 = Stops counting. 1 = Start counting. **Note1:** This field is used for internal 8 bit timer when TMRSEL(SC_CTL[14:13]) = 10 or TMRSEL(SC_CTL[14:13]) = 11. Don't filled CNTEN1 when TMRSEL(SC_CTL[14:13]) = 00 or TMRSEL(SC_CTL[14:13]) = 01. **Note2:** If the operation mode is not in auto-reload mode (SC_TMRCTL1[26] = 0), this bit will be auto-cleared by hardware. **Note3:** This field will be cleared by TXRST(SC_ALTCTL[0]) and RXRST(SC_ALTCTL[1]), so don't fill this bit, TXRST(SC_ALTCTL[0]), and RXRST(SC_ALTCTL[1]) at the same time. **Note4:** If SCEN(SC_CTL[0]) is not enabled, this filed cannot be programmed.
[5]	**CNTEN0**	**Internal Timer0 Start Enable Bit** This bit enables Timer 0 to start counting. Software can fill 0 to stop it and set 1 to reload and count. 0 = Stops counting. 1 = Start counting. **Note1:** This field is used for internal 24 bit timer when TMRSEL (SC_CTL[14:13]) = 01. **Note2:** If the operation mode is not in auto-reload mode (SC_TMRCTL0[26] = 0), this bit will be auto-cleared by hardware. **Note3:** This field will be cleared by TXRST(SC_ALTCTL[0]) and RXRST(SC_ALTCTL[1]). So don't fill this bit, TXRST and RXRST at the same time. **Note4:** If SCEN(SC_CTL[0]) is not enabled, this filed cannot be programmed.
[4]	**WARSTEN**	**Warm Reset Sequence Generator Enable Bit** This bit enables SC controller to initiate the card by warm reset sequence 0 = No effect. 1 = Warm reset sequence generator Enabled. **Note1:** When the warm reset sequence completed, this bit will be cleared automatically and the INITIF(SC_INTSTS[8]) will be set to 1. **Note2:** This field will be cleared by TXRST(SC_ALTCTL[0]) and RXRST(SC_ALTCTL[1]), so don't fill this bit, TXRST, and RXRST at the same time. **Note3:** If SCEN(SC_CTL[0]) is not enabled, this filed cannot be programmed.
[3]	**ACTEN**	**Activation Sequence Generator Enable Bit**

		This bit enables SC controller to initiate the card by activation sequence 0 = No effect. 1 = Activation sequence generator Enabled. **Note1:** When the activation sequence completed, this bit will be cleared automatically and the INITIF(SC_INTSTS[8]) will be set to 1. **Note2:** This field will be cleared by TXRST(SC_ALTCTL[0]) and RXRST(SC_ALTCTL[1]), so don't fill this bit, TXRST(SC_ALTCTL[0]), and RXRST(SC_ALTCTL[1]) at the same time. **Note3:** If SCEN(SC_CTL[0]) is not enabled, this filed cannot be programmed.
[2]	**DACTEN**	**Deactivation Sequence Generator Enable Bit** This bit enables SC controller to initiate the card by deactivation sequence 0 = No effect. 1 = Deactivation sequence generator Enabled. **Note1:** When the deactivation sequence completed, this bit will be cleared automatically and the INITIF(SC_INTSTS[8]) will be set to 1. **Note2:** This field will be cleared by TXRST (SC_ALTCTL[0]) and RXRST(SC_ALTCTL[1]). So don't fill this bit, TXRST, and RXRST at the same time. **Note3:** If SCEN (SC_CTL[0]) is not enabled, this filed cannot be programmed.
[1]	**RXRST**	**Rx Software Reset** When RXRST is set, all the bytes in the receiver buffer and Rx internal state machine will be cleared. 0 = No effect. 1 = Reset the Rx internal state machine and pointers. **Note:** This bit will be auto cleared after reset is complete.
[0]	**TXRST**	**TX Software Reset** When TXRST is set, all the bytes in the transmit buffer and TX internal state machine will be cleared. 0 = No effect. 1 = Reset the TX internal state machine and pointers. **Note:** This bit will be auto cleared after reset is complete.

SC Extend Guard Time Register (SC_EGT)

Register	Offset	R/W	Description	Reset Value
SC_EGT	SC_BA+0x0C	R/W	SC Extend Guard Time Register.	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
EGT							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**EGT**	**Extended Guard Time** This field indicates the extended guard timer value. **Note:** The counter is ETU base and the real extended guard time is EGT.

SC Receiver buffer Time-out Register (SC_RXTOUT)

Register	Offset	R/W	Description	Reset Value
SC_RXTOUT	SC_BA+0x10	R/W	SC Receive buffer Time-out Register.	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							RFTM
7	6	5	4	3	2	1	0
RFTM							

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:0]	**RFTM**	**SC Receiver FIFO Time-out (ETU Base)** The time-out counter resets and starts counting whenever the RX buffer received a new data word. Once the counter decrease to 1 and no new data is received or CPU does not read data by reading SC_DAT buffer, a receiver time-out interrupt INT_RTMR will be generated(if RXTOIF(SC_INTEN[9]) = 1). **Note1:** The counter unit is ETU based and the interval of time-out is RFTM + 0.5. **Note2:** Filling all 0 to this field indicates to disable this function.

SC Clock Divider Control Register (SC_ETUCTL)

Register	Offset	R/W	Description	Reset Value
SC_ETUCTL	SC_BA+0x14	R/W	SC ETU Control Register.	0x0000_0173

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
CMPEN	Reserved			ETURDIV			
7	6	5	4	3	2	1	0
ETURDIV							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**CMPEN**	**Compensation Mode Enable Bit** This bit enables clock compensation function. When this bit enabled, hardware will alternate between n clock cycles and n-1 clock cycles, where n is the value to be written into the ETURDIV . 0 = Compensation function Disabled. 1 = Compensation function Enabled.
[14:12]	**Reserved**	Reserved.
[11:0]	**ETURDIV**	**ETU Rate Divider** The field indicates the clock rate divider. The real ETU is ETURDIV + 1. **Note:** Software can configure this field, but this field must be greater than 0x004.

SC Interrupt Control Register (SC_INTEN)

Register	Offset	R/W	Description	Reset Value
SC_INTEN	SC_BA+0x18	R/W	SC Interrupt Enable Control Register.	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					ACERRIEN	RXTOIF	INITIEN
7	6	5	4	3	2	1	0
CDIEN	BGTIEN	TMR2IEN	TMR1IEN	TMR0IEN	TERRIEN	TBEIEN	RDAIEN

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10]	**ACERRIEN**	**Auto Convention Error Interrupt Enable Bit** This field is used for auto-convention error interrupt enable. 0 = Auto-convention error interrupt Disabled. 1 = Auto-convention error interrupt Enabled.
[9]	**RXTOIF**	**Receiver Buffer Time-out Interrupt Enable Bit** This field is used for receiver buffer time-out interrupt enable. 0 = Receiver buffer time-out interrupt Disabled. 1 = Receiver buffer time-out interrupt Enabled.
[8]	**INITIEN**	**Initial End Interrupt Enable Bit** This field is used for activation (ACTEN(SC_ALTCTL[3] = 1)), deactivation ((DACTEN SC_ALTCTL[2]) = 1) and warm reset (WARSTEN (SC_ALTCTL [4])) sequence interrupt enable. 0 = Initial end interrupt Disabled. 1 = Initial end interrupt Enabled.
[7]	**CDIEN**	**Card Detect Interrupt Enable Bit** This field is used for card detect interrupt enable. The card detect status is CINSERT(SC_STATUS[12]) 0 = Card detect interrupt Disabled. 1 = Card detect interrupt Enabled.
[6]	**BGTIEN**	**Block Guard Time Interrupt Enable Bit** This field is used for block guard time interrupt enable. 0 = Block guard time Disabled. 1 = Block guard time Enabled.
[5]	**TMR2IEN**	**Timer2 Interrupt Enable Bit** This field is used for TMR2 interrupt enable. 0 = Timer2 interrupt Disabled.

		1 = Timer2 interrupt Enabled.
[4]	**TMR1IEN**	**Timer1 Interrupt Enable Bit** This field is used to enable the TMR1 interrupt. 0 = Timer1 interrupt Disabled. 1 = Timer1 interrupt Enabled.
[3]	**TMR0IEN**	**Timer0 Interrupt Enable Bit** This field is used to enable TMR0 interrupt enable. 0 = Timer0 interrupt Disabled. 1 = Timer0 interrupt Enabled.
[2]	**TERRIEN**	**Transfer Error Interrupt Enable Bit** This field is used for transfer error interrupt enable. The transfer error states is at SC_STATUS register which includes receiver break error BEF(SC_STATUS[6]), frame error FEF(SC_STATUS[5]), parity error PEF(SC_STATUS[4]), receiver buffer overflow error RXOV(SC_STATUS[0]), transmit buffer overflow error TXOV(SC_STATUS[8]), receiver retry over limit error RXOVERR(SC_STATUS[22]) and transmitter retry over limit error TXOVERR (SC_STATUS[30]). 0 = Transfer error interrupt Disabled. 1 = Transfer error interrupt Enabled.
[1]	**TBEIEN**	**Transmit Buffer Empty Interrupt Enable Bit** This field is used for transmit buffer empty interrupt enable. 0 = Transmit buffer empty interrupt Disabled. 1 = Transmit buffer empty interrupt Enabled.
[0]	**RDAIEN**	**Receive Data Reach Interrupt Enable Bit** This field is used for received data reaching trigger level RXTRGLV (SC_CTL[7:6]) interrupt enable. 0 = Receive data reach trigger level interrupt Disabled. 1 = Receive data reach trigger level interrupt Enabled.

SC Interrupt Status Register (SC_INTSTS)

Register	Offset	R/W	Description	Reset Value
SC_INTSTS	SC_BA+0x1C	R/W	SC Interrupt Status Register.	0x0000_0002

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					ACERRIF	RBTOIF	INITIF
7	6	5	4	3	2	1	0
CDIF	BGTIF	TMR2IF	TMR1IF	TMR0IF	TERRIF	TBEIF	RDAIF

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10]	**ACERRIF**	**Auto Convention Error Interrupt Status Flag (Read Only)** This field indicates auto convention sequence error. If the received TS at ATR state is neither 0x3B nor 0x3F, this bit will be set. **Note:** This bit is read only, but it can be cleared by writing 1 to it.
[9]	**RBTOIF**	**Receiver Buffer Time-out Interrupt Status Flag (Read Only)** This field is used for receiver buffer time-out interrupt status flag. **Note:** This field is the status flag of receiver buffer time-out state. If software wants to clear this bit, software must read all receiver buffer remaining data by reading SC_DAT buffer,
[8]	**INITIF**	**Initial End Interrupt Status Flag (Read Only)** This field is used for activation (ACTEN(SC_ALTCTL[3])), deactivation (DACTEN (SC_ALTCTL[2])) and warm reset (WARSTEN (SC_ALTCTL[4])) sequence interrupt status flag. **Note:** This bit is read only, but it can be cleared by writing 1 to it.
[7]	**CDIF**	**Card Detect Interrupt Status Flag (Read Only)** This field is used for card detect interrupt status flag. The card detect status is CINSERT (SC_STATUS[12]) and CREMOVE(SC_STATUS[11]). **Note:** This field is the status flag of CINSERT(SC_STATUS[12]) or CREMOVE(SC_STATUS[11])]. So if software wants to clear this bit, software must write 1 to this field.
[6]	**BGTIF**	**Block Guard Time Interrupt Status Flag (Read Only)** This field is used for block guard time interrupt status flag. **Note1:** This bit is valid when RXBGTEN (SC_ALTCTL[12]) is enabled. **Note2:** This bit is read only, but it can be cleared by writing “1” to it.
[5]	**TMR2IF**	**Timer2 Interrupt Status Flag (Read Only)** This field is used for TMR2 interrupt status flag. **Note:** This bit is read only, but it can be cleared by writing 1 to it.

[4]	**TMR1IF**	**Timer1 Interrupt Status Flag (Read Only)** This field is used for TMR1 interrupt status flag. **Note:** This bit is read only, but it can be cleared by writing 1 to it.
[3]	**TMR0IF**	**Timer0 Interrupt Status Flag (Read Only)** This field is used for TMR0 interrupt status flag. **Note:** This bit is read only, but it can be cleared by writing 1 to it.
[2]	**TERRIF**	**Transfer Error Interrupt Status Flag (Read Only)** This field is used for transfer error interrupt status flag. The transfer error states is at SC_STATUS register which includes receiver break error BEF(SC_STATUS[6]), frame error FEF(SC_STATUS[5], parity error PEF(SC_STATUS[4] and receiver buffer overflow error RXOV(SC_STATUS[0]), transmit buffer overflow error TXOV(SC_STATUS[8]), receiver retry over limit error RXOVERR(SC_STATUS[22] and transmitter retry over limit error TXOVERR(SC_STATUS[30]). **Note:** This field is the status flag of BEF(SC_STATUS[6]), FEF(SC_STATUS[5]), PEF(SC_STATUS[4]), RXOV(SC_STATUS[0]), TXOV(SC_STATUS[8]), RXOVERR(SC_STATUS[22]) or TXOVERR(SC_STATUS[30]). So, if software wants to clear this bit, software must write 1 to each field.
[1]	**TBEIF**	**Transmit Buffer Empty Interrupt Status Flag (Read Only)** This field is used for transmit buffer empty interrupt status flag. **Note:** This field is the status flag of transmit buffer empty state. If software wants to clear this bit, software must write data to DAT(SC_DAT[7:0]) buffer and then this bit will be cleared automatically.
[0]	**RDAIF**	**Receive Data Reach Interrupt Status Flag (Read Only)** This field is used for received data reaching trigger level RXTRGLV (SC_CTL[7:6]) interrupt status flag. **Note:** This field is the status flag of received data reaching RXTRGLV (SC_CTL[7:6]). If software reads data from SC_DAT and receiver buffer data byte number is less than RXTRGLV (SC_CTL[7:6]), this bit will be cleared automatically.

SC Transfer Status Register (SC_STATUS)

Register	Offset	R/W	Description	Reset Value
SC_STATUS	SC_BA+0x20	R/W	SC Status Register.	0x0000_0202

31	30	29	28	27	26	25	24
TXACT	TXOVERR	TXRERR	Reserved			TXPOINT	
23	22	21	20	19	18	17	16
RXACT	RXOVERR	RXRERR	Reserved			RXPOINT	
15	14	13	12	11	10	9	8
Reserved		CDPINSTS	CINSERT	CREMOVE	TXFULL	TXEMPTY	TXOV
7	6	5	4	3	2	1	0
Reserved	BEF	FEF	PEF	Reserved	RXFULL	RXEMPTY	RXOV

Bits	Description	
[31]	**TXACT**	**Transmit in Active Status Flag (Read Only)** 0 = This bit is cleared automatically when TX transfer is finished or the last byte transmission has completed. 1 = This bit is set by hardware when TX transfer is in active and the STOP bit of the last byte has been transmitted.
[30]	**TXOVERR**	**Transmitter over Retry Error (Read Only)** This bit is set by hardware when transmitter re-transmits over retry number limitation. **Note:** This bit is read only, but it can be cleared by writing 1 to it.
[29]	**TXRERR**	**Transmitter Retry Error (Read Only)** This bit is set by hardware when transmitter re-transmits. **Note1:** This bit is read only, but it can be cleared by writing 1 to it. **Note2** This bit is a flag and cannot generate any interrupt to CPU.
[28:26]	**Reserved**	Reserved.
[25:24]	**TXPOINT**	**Transmit Buffer Pointer Status Flag (Read Only)** This field indicates the TX buffer pointer status flag. When CPU writes data into SC_DAT, TXPOINT increases one. When one byte of TX Buffer is transferred to transmitter shift register, TXPOINT decreases one.
[23]	**RXACT**	**Receiver in Active Status Flag (Read Only)** This bit is set by hardware when RX transfer is in active. This bit is cleared automatically when RX transfer is finished.
[22]	**RXOVERR**	**Receiver over Retry Error (Read Only)** This bit is set by hardware when RX transfer error retry over retry number limit. **Note1:** This bit is read only, but it can be cleared by writing 1 to it. **Note2:** If CPU enables receiver retries function by setting RXRTYEN (SC_CTL[19]), the PEF(SC_STATUS[4]) flag will be ignored (hardware will not set PEF(SC_STATUS[4])).
[21]	**RXRERR**	**Receiver Retry Error (Read Only)** This bit is set by hardware when RX has any error and retries transfer. **Note1:** This bit is read only, but it can be cleared by writing 1 to it.

		Note2 This bit is a flag and cannot generate any interrupt to CPU. **Note3:** If CPU enables receiver retry function by setting RXRTYEN (SC_CTL[19]) , the PEF(SC_STATUS[4]) flag will be ignored (hardware will not set PEF(SC_STATUS[4])).
[20:18]	**Reserved**	Reserved.
[17:16]	**RXPOINT**	**Receiver Buffer Pointer Status Flag (Read Only)** This field indicates the RX buffer pointer status flag. When SC receives one byte from external device, RXPOINT(SC_STATUS[17:16]) increases one. When one byte of RX buffer is read by CPU, RXPOINT(SC_STATUS[17:16]) decreases one.
[15:14]	**Reserved**	Reserved.
[13]	**CDPINSTS**	**Card Detect Status of SC_CD Pin Status (Read Only)** This bit is the pin status flag of SC_CD 0 = The SC_CD pin state at low. 1 = The SC_CD pin state at high.
[12]	**CINSERT**	**Card Detect Insert Status of SC_CD Pin (Read Only)** This bit is set whenever card has been inserted. 0 = No effect. 1 = Card insert. **Note1:** This bit is read only, but it can be cleared by writing “1” to it. **Note2:** The card detect engine will start after SCEN (SC_CTL[0]) set.
[11]	**CREMOVE**	**Card Detect Removal Status of SC_CD Pin (Read Only)** This bit is set whenever card has been removal. 0 = No effect. 1 = Card removed. **Note1:** This bit is read only, but it can be cleared by writing “1” to it. **Note2:** Card detect engine will start after SCEN (SC_CTL[0])set.
[10]	**TXFULL**	**Transmit Buffer Full Status Flag (Read Only)** This bit indicates TX buffer full or not.This bit is set when TX pointer is equal to 4, otherwise is cleared by hardware.
[9]	**TXEMPTY**	**Transmit Buffer Empty Status Flag (Read Only)** This bit indicates TX buffer empty or not. When the last byte of TX buffer has been transferred to Transmitter Shift Register, hardware sets this bit high. It will be cleared when writing data into DAT(SC_DAT[7:0]) (TX buffer not empty).
[8]	**TXOV**	**TX Overflow Error Interrupt Status Flag (Read Only)** If TX buffer is full, an additional write to DAT(SC_DAT[7:0]) will cause this bit be set to “1” by hardware. **Note:** This bit is read only, but it can be cleared by writing 1 to it.
[7]	**Reserved**	Reserved.
[6]	**BEF**	**Receiver Break Error Status Flag (Read Only)** This bit is set to logic 1 whenever the received data input (RX) held in the “spacing state” (logic 0) is longer than a full word transmission time (that is, the total time of “start bit” + data bits + parity + stop bits). . **Note1:** This bit is read only, but it can be cleared by writing 1 to it. **Note2:** If CPU sets receiver retries function by setting RXRTYEN(SC_CTL[19]) , hardware will not set this flag.
[5]	**FEF**	**Receiver Frame Error Status Flag (Read Only)** This bit is set to logic 1 whenever the received character does not have a valid “stop bit” (that is, the stop bit following the last data bit or parity bit is detected as logic 0).

		Note1: This bit is read only, but it can be cleared by writing 1 to it. **Note2:** If CPU sets receiver retries function by setting RXRTYEN(SC_CTL[19]) , hardware will not set this flag.
[4]	**PEF**	**Receiver Parity Error Status Flag (Read Only)** This bit is set to logic 1 whenever the received character does not have a valid "parity bit". **Note1:** This bit is read only, but it can be cleared by writing 1 to it. **Note2:** If CPU sets receiver retries function by setting RXRTYEN(SC_CTL[19]) , hardware will not set this flag.
[3]	**Reserved**	Reserved.
[2]	**RXFULL**	**Receiver Buffer Full Status Flag (Read Only)** This bit indicates RX buffer full or not. This bit is set when RX pointer is equal to 4, otherwise it is cleared by hardware.
[1]	**RXEMPTY**	**Receiver Buffer Empty Status Flag(Read Only)** This bit indicates RX buffer empty or not. When the last byte of Rx buffer has been read by CPU, hardware sets this bit high. It will be cleared when SC receives any new data.
[0]	**RXOV**	**RX Overflow Error Status Flag (Read Only)** This bit is set when RX buffer overflow. If the number of received bytes is greater than Rx Buffer size (4 bytes), this bit will be set. **Note:** This bit is read only, but it can be cleared by writing 1 to it.

SC PIN Control State Register (SC_PINCTL)

Register	Offset	R/W	Description	Reset Value
SC_PINCTL	SC_BA+0x24	R/W	SC Pin Control State Register.	0x0000_00x0

31	30	29	28	27	26	25	24
Reserved	SYNC	Reserved					
23	22	21	20	19	18	17	16
Reserved					RSTSTS	PWRSTS	DATSTS
15	14	13	12	11	10	9	8
Reserved			SCDOSTS	PWRINV	Reserved	SCDOUT	Reserved
7	6	5	4	3	2	1	0
Reserved	CLKKEEP	Reserved				SCRST	PWREN

Bits	Description	
[31]	**Reserved**	Reserved.
[30]	**SYNC**	**SYNC Flag Indicator** Due to synchronization, software should check this bit when writing a new value to SC_PINCTL register. 0 = Synchronizing is completion, user can write new data to SC_PINCTL register. 1 = Last value is synchronizing. **Note:** This bit is read only.
[29:19]	**Reserved**	Reserved.
[18]	**RSTSTS**	**SCRST Pin Signals** This bit is the pin status of SC_RST 0 = SC_RST pin is low. 1 = SC_RST pin is high. **Note:** When SC is operated at activation, warm reset or deactivation mode, this bit will be changed automatically. This bit is not allowed to program when SC is operated at these modes.
[17]	**PWRSTS**	**SC_PWR Pin Signal** This bit is the pin status of SC_PWR 0 = SC_PWR pin to low. 1 = SC_PWR pin to high. **Note:** When SC is operated at activation, warm reset or deactivation mode, this bit will be changed automatically. This bit is not allowed to program when SC is operated at these modes.
[16]	**DATSTS**	This bit is the pin status of SC_DAT 0 = The SC_DAT pin is low. 1 = The SC_DAT pin is high.
[15:13]	**Reserved**	Reserved.

[12]	**SCDOSTS**	**SC Data Pin Output Status** This bit is the pin status of SCDATOUT 0 = SCDATOUT pin to low. 1 = SCDATOUT pin to high. **Note:** When SC is operated at activation, warm reset or deactivation mode, this bit will be changed automatically. This bit is not allowed to program when SC is operated at these modes.
[11]	**PWRINV**	**SC_POW Pin Inverse** This bit is used for inverse the SC_POW pin. There are four kinds of combination for SC_POW pin setting by PWRINV(SC_PINCTL[11]) and PWREN(SC_PINCTL[0]). PWRINV (SC_PINCTL[11]) is bit 1 and PWREN(SC_PINCTL[0]) is bit 0 for SC_POW_Pin as high or low voltage selection. 00 = SC_POW_ Pin is 0. 01 = SC_POW _Pin is 1. 10 = SC_POW _Pin is 1. 11 = SC_POW_ Pin is 0. **Note:** Software must select PWRINV (SC_PINCTL[11]) before Smart Card is enabled by SCEN (SC_CTL[0]).
[10]	**Reserved**	Reserved.
[9]	**SCDOUT**	**SC Data Output Pin** This bit is the pin status of SCDATOUT but user can drive SCDATOUT pin to high or low by setting this bit. 0 = Drive SCDATOUT pin to low. 1 = Drive SCDATOUT pin to high. **Note:** When SC is at activation, warm reset or deactivation mode, this bit will be changed automatically. So don't fill this field when SC is in these modes.
[8:7]	**Reserved**	Reserved.
[6]	**CLKKEEP**	**SC Clock Enable Bit** 0 = SC clock generation Disabled. 1 = SC clock always keeps free running. **Note:** When operating in activation, warm reset or deactivation mode, this bit will be changed automatically. So don't fill this field when operating in these modes.
[5:2]	**Reserved**	Reserved.
[1]	**SCRST**	**SC_RST Pin Signal** This bit is the pin status of SC_RST but user can drive SC_RST pin to high or low by setting this bit. Write this field to drive SC_RST pin. 0 = Drive SC_RST pin to low. 1 = Drive SC_RST pin to high. Read this field to get SC_RST pin status. 0 = SC_RST pin status is low. 1 = SC_RST pin status is high. **Note:** When operating at activation, warm reset or deactivation mode, this bit will be changed automatically. So don't fill this field when operating in these modes.
[0]	**PWREN**	**SC_PWREN Pin Signal** Software can set PWREN (SC_PINCTL[0]) and PWRINV (SC_PINCTL[11])to decide SC_PWR pin is in high or low level. Write this field to drive SC_PWR pin Refer PWRINV (SC_PINCTL[11]) description for programming SC_PWR pin voltage level.

		Read this field to get SC_PWR pin status. 0 = SC_PWR pin status is low. 1 = SC_PWR pin status is high. **Note:** When operating at activation, warm reset or deactivation mode, this bit will be changed automatically. So don't fill this field when operating in these modes.

SC Timer Control Register 0 (SC_TMRCTL0)

Register	Offset	R/W	Description	Reset Value
SC_TMRCTL0	SC_BA+0x28	R/W	SC Internal Timer Control Register 0.	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				OPMODE			
23	22	21	20	19	18	17	16
CNT							
15	14	13	12	11	10	9	8
CNT							
7	6	5	4	3	2	1	0
CNT							

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:24]	**OPMODE**	**Timer 0 Operation Mode Selection** This field indicates the internal 24-bit timer operation selection. Refer to 6.14.5.4 for programming Timer0
[23:0]	**CNT**	**Timer 0 Counter Value (ETU Base)** This field indicates the internal timer operation values.

SC Timer Control Register 1 (SC_TMRCTL1)

Register	Offset	R/W	Description	Reset Value
SC_TMRCTL1	SC_BA+0x2C	R/W	SC Internal Timer Control Register 1.	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				OPMODE			
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
CNT							

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:24]	**OPMODE**	**Timer 1 Operation Mode Selection** This field indicates the internal 8-bit timer operation selection. Refer to 6.14.5.4 for programming Timer1
[23:8]	**Reserved**	Reserved.
[7:0]	**CNT**	**Timer 1 Counter Value (ETU Base)** This field indicates the internal timer operation values.

SC Timer Control Register 2 (SC_TMRCTL2)

Register	Offset	R/W	Description	Reset Value
SC_TMRCTL2	SC_BA+0x30	R/W	SC Internal Timer Control Register 2.	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				OPMODE			
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
CNT							

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:24]	**OPMODE**	**Timer 2 Operation Mode Selection** This field indicates the internal 8-bit timer operation selection Refer to 6.14.5.4 for programming Timer2
[23:8]	**Reserved**	Reserved.
[7:0]	**CNT**	**Timer 2 Counter Value (ETU Base)** This field indicates the internal timer operation values.

SC UART Mode Control Register (SC_UARTCTL)

Register	Offset	R/W	Description	Reset Value
SC_UARTCTL	SC_BA + 0x34	R/W	SC UART Mode Control Register.	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
OPE	PBOFF	WLS		Reserved			UARTEN

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**OPE**	**Odd Parity Enable Bit** 0 = Even number of logic 1's are transmitted or check the data word and parity bits in receiving mode. 1 = Odd number of logic 1's are transmitted or check the data word and parity bits in receiving mode. **Note:** This bit has effect only when PBOFF bit is '0'.
[6]	**PBOFF**	**Parity Bit Disable Control** 0 = Parity bit is generated or checked between the "last data word bit" and "stop bit" of the serial data. 1 = Parity bit is not generated (transmitting data) or checked (receiving data) during transfer. **Note:** In smart card mode, this field must be '0' (default setting is with parity bit)
[5:4]	**WLS**	**Word Length Selection** 00 = Word length is 8 bits. 01 = Word length is 7 bits. 10 = Word length is 6 bits. 11 = Word length is 5 bits. **Note:** In smart card mode, this WLS must be '00'
[3:1]	**Reserved**	Reserved.
[0]	**UARTEN**	**UART Mode Enable Bit** 0 = Smart Card mode. 1 = UART mode. **Note1:** When operating in UART mode, user must set CONSEL (SC_CTL[5:4]) = 00 and AUTOCEN(SC_CTL[3]) = 0. **Note2:** When operating in Smart Card mode, user must set UARTEN(SC_UARTCTL [0]) = 00. **Note3:** When UART is enabled, hardware will generate a reset to reset FIFO and internal state machine.

SC Timer Current Data Register A (SC_TMRDAT0)

Register	Offset	R/W	Description	Reset Value
SC_TMRDAT0	SC_BA+0x38	R	SC Timer Current Data Register A.	0x0000_07FF

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
CNT0							
15	14	13	12	11	10	9	8
CNT0							
7	6	5	4	3	2	1	0
CNT0							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**CNT0**	**Timer0 Current Data Value (Read Only)** This field indicates the current count values of timer0.

SC Timer Current Data Register B (SC_TMRDAT1_2)

Register	Offset	R/W	Description	Reset Value
SC_TMRDAT1_2	SC_BA+0x3C	R	SC Timer Current Data Register B.	0x0000_7F7F

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
CNT2							
7	6	5	4	3	2	1	0
CNT1							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**CNT2**	**Timer2 Current Data Value (Read Only)** This field indicates the current count values of timer2.
[7:0]	**CNT1**	**Timer1 Current Data Value (Read Only)** This field indicates the current count values of timer1.

6.15 I²C Serial Interface Controller (I²C)

6.15.1 Overview

I²C is a two-wire, bi-directional serial bus that provides a simple and efficient method of data exchange between devices. The I²C standard is a true multi-master bus including collision detection and arbitration that prevents data corruption if two or more masters attempt to control the bus simultaneously.

There are two sets of I²C controller which supports Bus Management (System Management (SM)/Power Management (PM) bus compatible) and Power-down wake-up function.

6.15.2 Features

The I²C bus uses two wires (SDA and SCL) to transfer information between devices connected to the bus. The main features of the I²C bus include:

- Supports up to two I²C ports
- Master/Slave mode
- Bidirectional data transfer between masters and slaves
- Multi-master bus (no central master)
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- Serial clock synchronization allow devices with different bit rates to communicate via one serial bus
- Built-in 14-bit time-out counter requesting the I²C interrupt if the I²C bus hangs up and timer-out counter overflows.
- Programmable clocks allow for versatile rate control
- Supports 7-bit addressing mode
- Supports multiple address recognition (four slave address with mask option)
- Supports Bus Management (SM/PM compatible) function
- Supports Power-down wake-up function

6.15.3 Basic Configuration

The basic configurations of I2C0 are as follows:

- I2C0 pins are configured on SYS_GPA_MFPL or SYS_GPD_MFPL or SYS_GPE_MFPH or SYS_GPA_MFPH registers.
- Enable I2C0 clock (I2C0CKEN) on CLK_APBCLK0 [8] register.
- Reset I2C0 controller (I2C0RST) on SYS_IPRST1 [8] register.

The basic configurations of I2C1 are as follows:

- I2C1 pins are configured on SYS_GPC_MFPL or SYS_GPE_MFPL or SYS_GPE_MFPH or SYS_GPF_MFPL registers.
- Enable I2C1 clock (I2C1CKEN) on CLK_APBCLK0[9] register.
- Reset I2C1 controller (I2C1RST) on SYS_IPRST1[9] register.

6.15.4 Block Diagram

The block diagram of I^2C controller is shown as Figure 6.15-1.

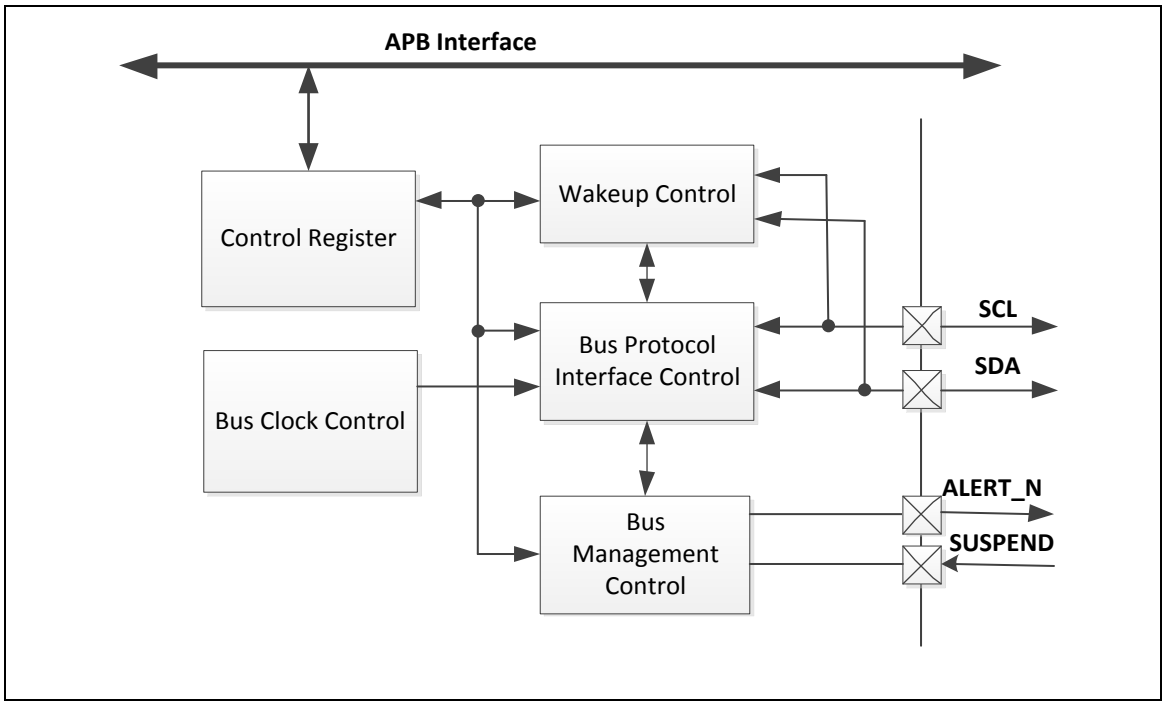


Figure 6.15-1 I^2C Controller Block Diagram

6.15.5 Functional Description

On I^2C bus, data is transferred between a Master and a Slave. Data bits transfer on the SCL and SDA lines are synchronously on a byte-by-byte basis. Each data byte is 8-bit long. 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). Please refer to the Figure 6.15-2 for more detailed I^2C BUS Timing.

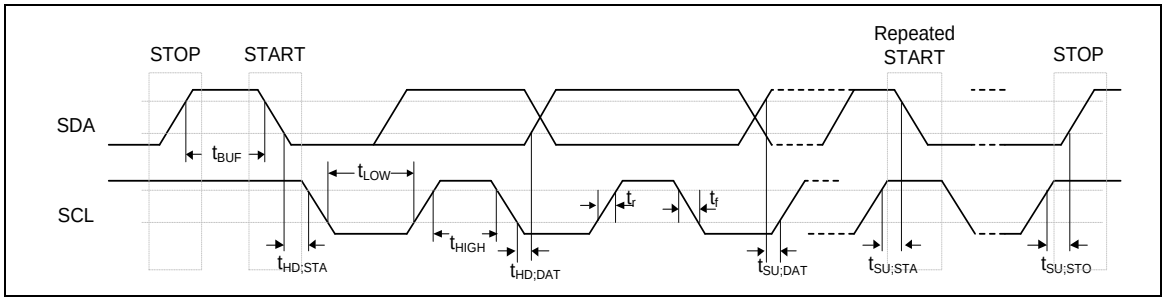


Figure 6.15-2 I²C Bus Timing

The device's on-chip I²C provides the serial interface that meets the I²C bus standard mode specification. The I²C port handles byte transfers autonomously. To enable this port, the bit I2CEN in I2C_CTL should be set to '1'. The I²C hardware interfaces to the I²C bus via two pins: SDA and SCL. When I/O pins are used as I²C ports, user must set the pins function to I²C in advance.

Note: Pull-up resistor is needed for I²C operation as the SDA and SCL are open-drain pins.

6.15.5.1 I²C Protocol

The Figure 6.15-3 shows the typical I²C protocol. Normally, a standard communication consists of four parts:

- START or Repeated START signal generation
- Slave address and R/W bit transfer
- Data transfer
- STOP signal generation

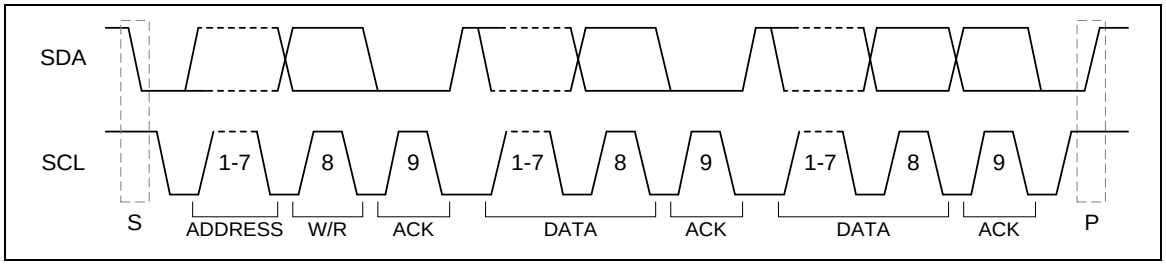


Figure 6.15-3 I²C Protocol

START or Repeated START signal

When the bus is free/idle, which means no master device is engaging the bus (both SCL and SDA lines are high), a master can initiate a transfer by sending a START signal. A START signal, usually referred to as the “S” bit, is defined as a HIGH to LOW transition on the SDA line while SCL is HIGH. The START signal denotes the beginning of a new data transmission.

After having sent the address byte (address and read/write bit), the master may send any number of bytes followed by a stop condition. Instead of sending the stop condition it is also allowed to send another start condition again followed by an address (and of course including a read/write bit) and more data. The start condition is called as Repeat START (Sr). This is defined recursively allowing any number of start conditions to be sent. The purpose of this is to allow combined write/read operations to one or more devices without releasing the bus and thus with the guarantee that the operation is not interrupted. The controller uses this method to communicate with another slave or the same slave in a different transfer direction (e.g. from writing to a device to reading from a device) without releasing the bus.

STOP signal

The master can terminate the communication by generating a STOP signal. A STOP signal, usually referred to as the “P” bit, is defined as a LOW to HIGH transition on the SDA line while SCL is HIGH.

The Figure 6.15-4 shows the waveform of START, Repeat START and STOP.

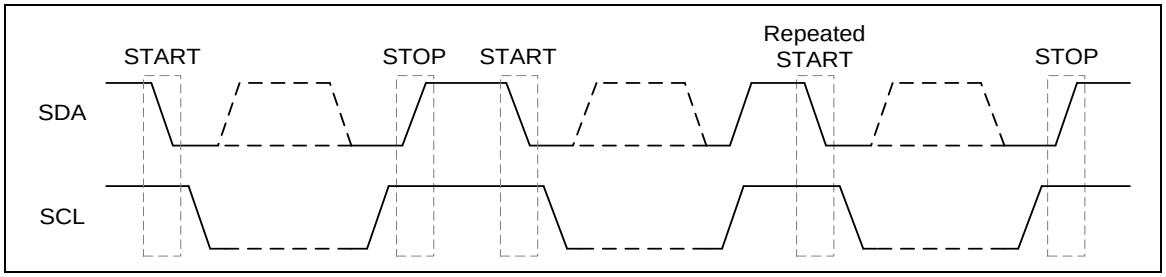


Figure 6.15-4 START and STOP Conditions

Slave Address Transfer

The first byte of data transferred by the master immediately after the START signal is the Slave address (SLA). This is a 7-bit calling address followed by a Read/Write (R/W) bit. The R/W bit signals of the slave indicate the data transfer direction. No two slaves in the system can have the same address. Only the slave with an address that matches the one transmitted by the master will respond by returning an acknowledge bit by pulling the SDA low at the 9th SCL clock cycle.

Data Transfer

When a slave receives a correct address with an R/W bit, the data will follow R/W bit specified to transfer. Each transferred byte is followed by an acknowledge bit on the 9th SCL clock cycle. If the slave signals a Not Acknowledge (NACK), the master can generate a STOP signal to abort the data transfer or generate a Repeated START signal and start a new transfer cycle.

If the master, as a receiving device, does Not Acknowledge (NACK) the slave, the slave releases the SDA line for the master to generate a STOP or Repeated START signal.

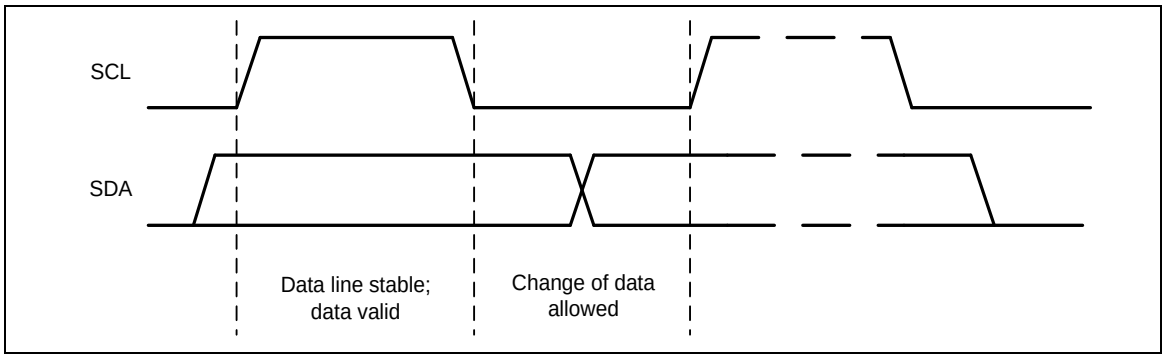


Figure 6.15-5 Bit Transfer on the I^{2}C Bus

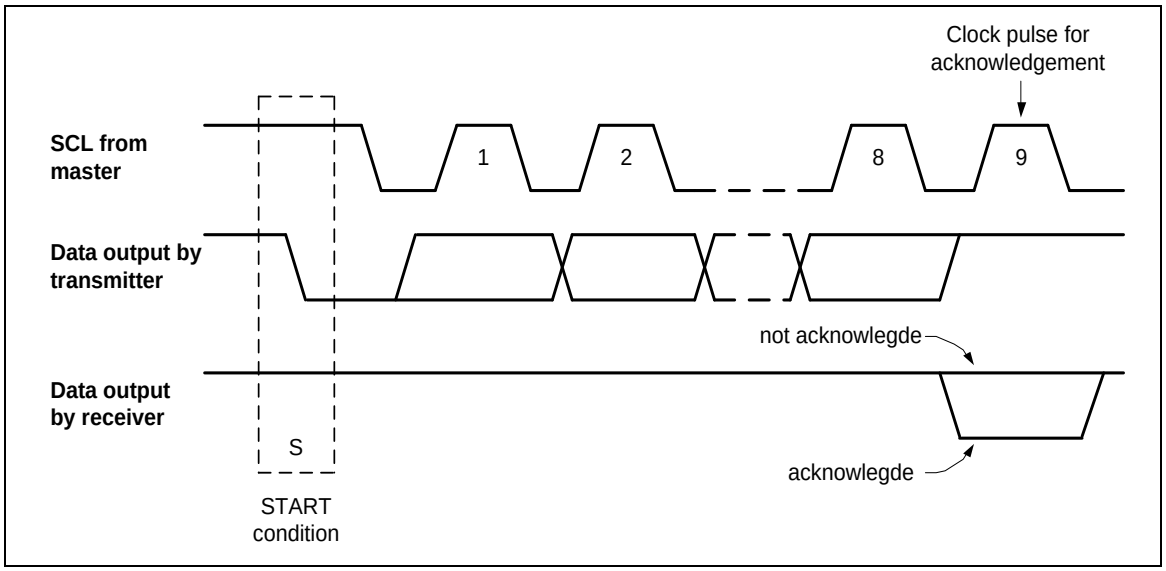


Figure 6.15-6 Acknowledge on the I^{2}C Bus

Data transfer on I²C bus

The Figure 6.15-7 shows a master transmits data to slave. A master addresses a slave with a 7-bit address and 1-bit write index to denote that the master wants to transmit data to the slave. The master keeps transmitting data after the slave returns acknowledge to the master.

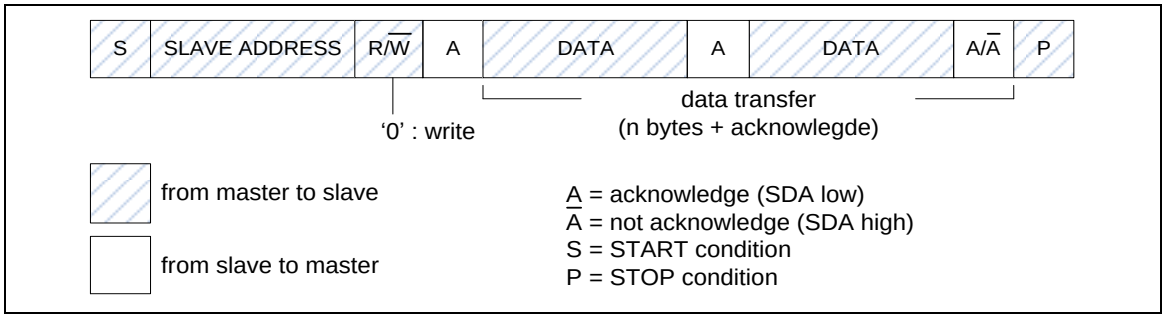


Figure 6.15-7 Master Transmits Data to Slave

The Figure 6.15-8 shows a master read data from slave. A master addresses a slave with a 7-bit address and 1-bit read index to denote that the master wants to read data from the slave. The slave will start transmitting data after the slave returns acknowledge to the master.

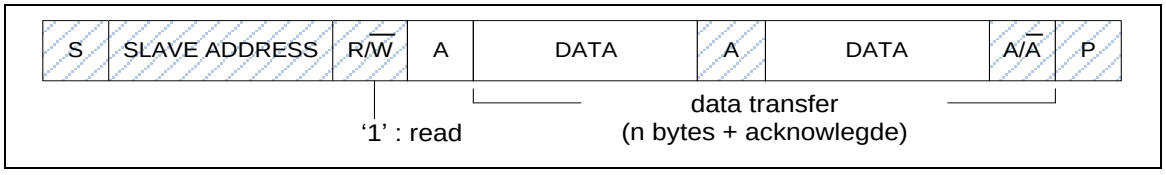


Figure 6.15-8 Master Reads Data from Slave

6.15.5.2 Operation Modes

The on-chip I²C ports support three operation modes, Master, Slave, and General Call Mode.

In a given application, I²C port may operate as a master or as a slave. In Slave mode, the I²C port hardware looks for its own slave address and the general call address. If one of these addresses is detected, and if the slave is willing to receive or transmit data from/to master(by setting the AA bit), acknowledge pulse will be transmitted out on the 9th clock, hence an interrupt is requested on both master and slave devices if interrupt is enabled. When the microcontroller wishes to become the bus master, hardware waits until the bus is free before entering Master mode so that a possible slave action is not be interrupted. If bus arbitration is lost in Master mode, I²C port switches to Slave mode immediately and can detect its own slave address in the same serial transfer.

To control the I²C bus transfer in each mode, user needs to set I2C_CTL, I2C_DAT registers according to current status code of I2C_STATUS register. In other words, for each I²C bus action, user needs to check current status by I2C_STATUS register, and then set I2C_CTL, I2C_DAT registers to take bus action. Finally, check the response status by I2C_STATUS.

The bits, STA, STO and AA in I2C_CTL register are used to control the next state of the I²C hardware after SI flag of I2C_CTL [3] register is cleared. Upon completion of the new action, a new status code will be updated in I2C_STATUS register and the SI flag of I2C_CTL register will be set. If the I²C interrupt control bit INTEN (I2C_CTL [7]) is set, appropriate action or software branch of the new status code can be performed in the Interrupt service routine.

The Figure 6.15-9 shows the current I²C status code is 0x08, and then set I2C_DATA=SLA+W and (STA,STO,SI,AA) = (0,0,1,x) to send the address to I²C bus. If a slave on the bus matches the address and response ACK, the I2C_STATUS will be updated by status code 0x18.

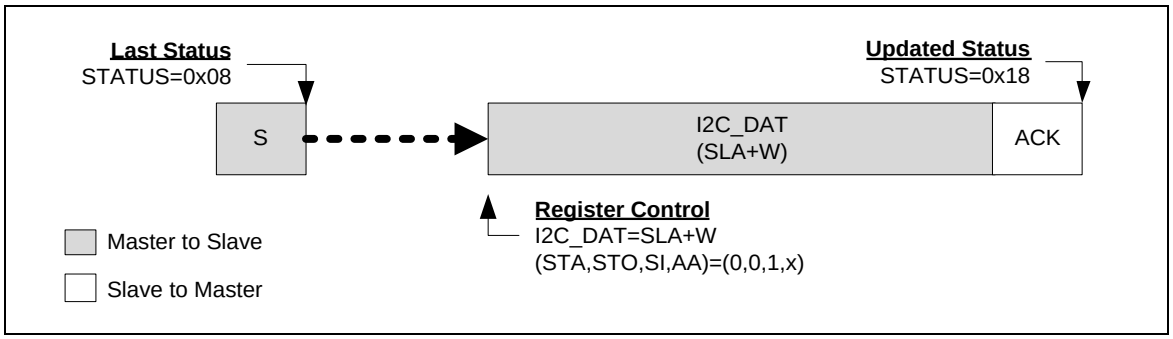


Figure 6.15-9 Control I²C Bus according to the current I²C Status

Master Mode

InFigure 6.15-10 and Figure 6.15-11, all possible protocols for I^{2}C master are shown. User needs to follow proper path of the flow to implement required I^{2}C protocol.

In other words, user can send a START signal to bus and I^{2}C will be in Master Transmitter (MT) mode (Figure 6.15-10) or Master receiver (MR) mode (Figure 6.15-11) after START signal has been sent successfully and new status code would be 0x08. Followed by START signal, user can send slave address, read/write bit, data and Repeat START, STOP to perform I^{2}C protocol.

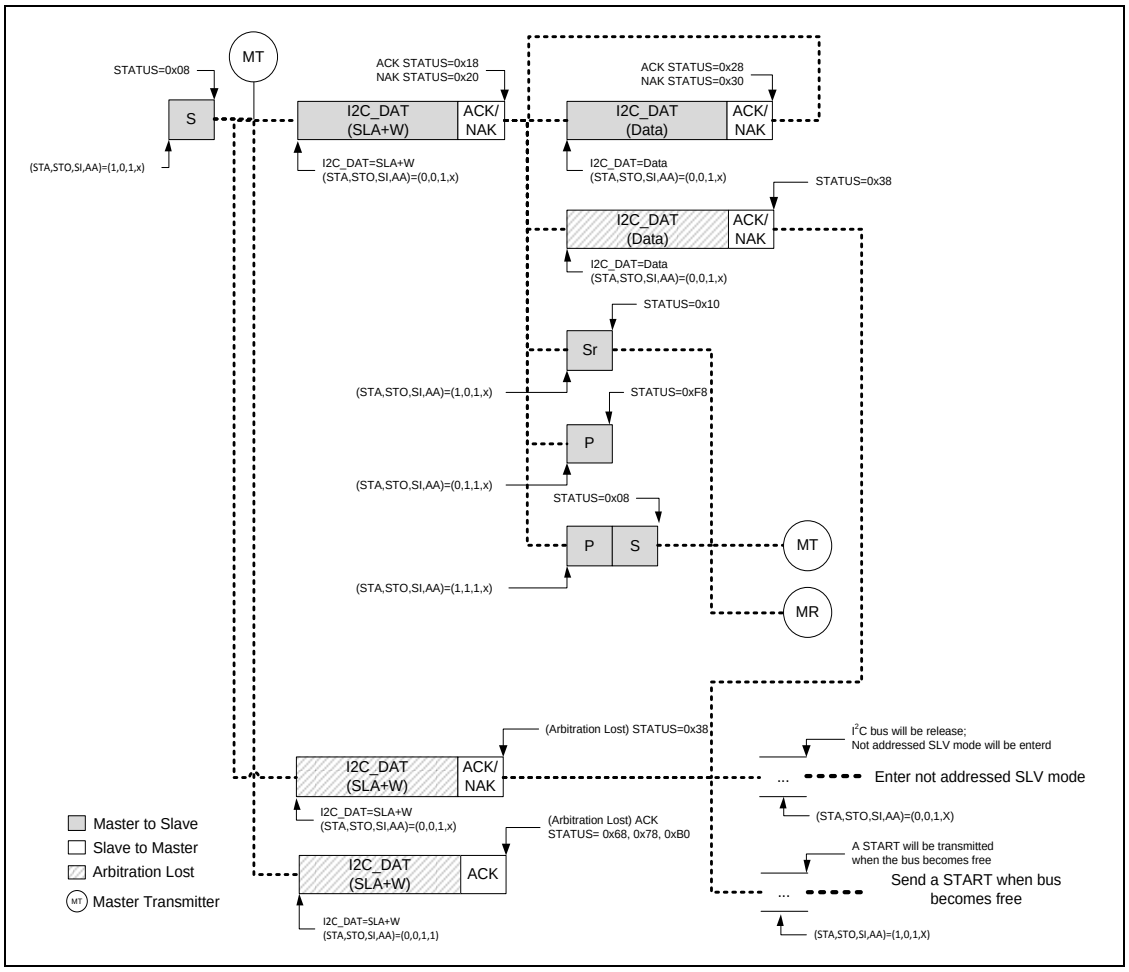


Figure 6.15-10 Master Transmitter Mode Control Flow

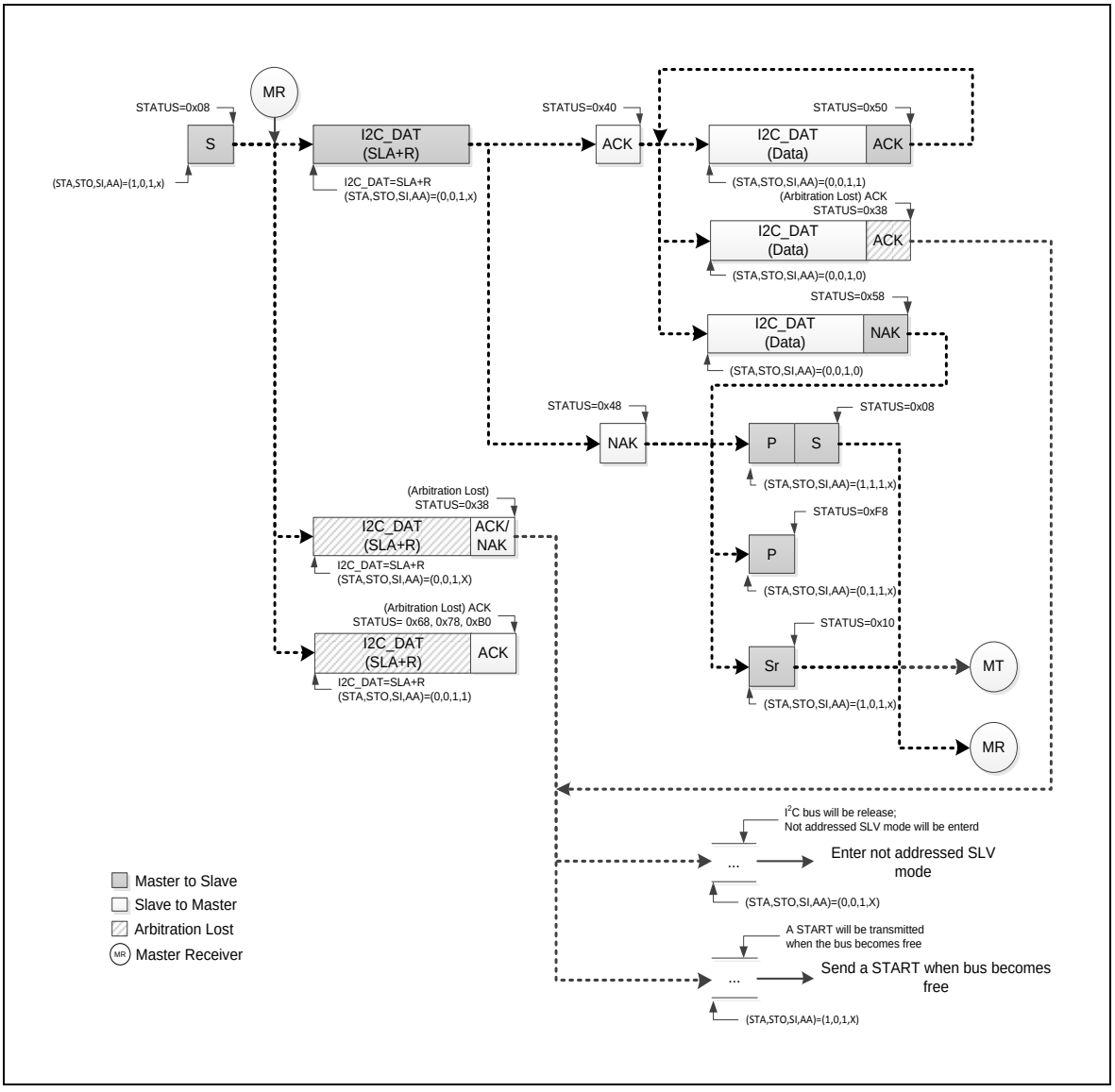


Figure 6.15-11 Master Receiver Mode Control Flow

If the I²C is in Master mode and gets arbitration lost, the status code will be 0x38. In status 0x38, user may set (STA, STO, SI, AA) = (1, 0, 1, X) to send START to re-start Master operation when bus become free. Otherwise, user may set (STA, STO, SI, AA) = (0, 0, 1, X) to release I²C bus and enter not addressed Slave mode.

Slave Mode

When reset default, I^2C is not addressed and will not recognize the address on I^2C bus. User can set slave address by I2C_ADDRn (n=0~3) and set (STA, STO, SI, AA) = (0, 0, 1, 1) to let I^2C recognize the address sent by master. Figure 6.15-12 shows all the possible flow for I^2C in Slave mode. Users need to follow a proper flow (as shown in Figure 6.15-12 to implement their own I^2C protocol.

If bus arbitration is lost in Master mode, I^2C port switches to Slave mode immediately and can detect its own slave address in the same serial transfer. If the detected address is SLA+W (Master want to write data to Slave) after arbitration lost, the status code is 0x68. If the detected address is SLA+R (Master want to read data from Slave) after arbitration lost, the status code is 0xB0.

Note: During I^2C communication, the SCL clock will be released when writing '1' to clear SI flag in Slave mode.

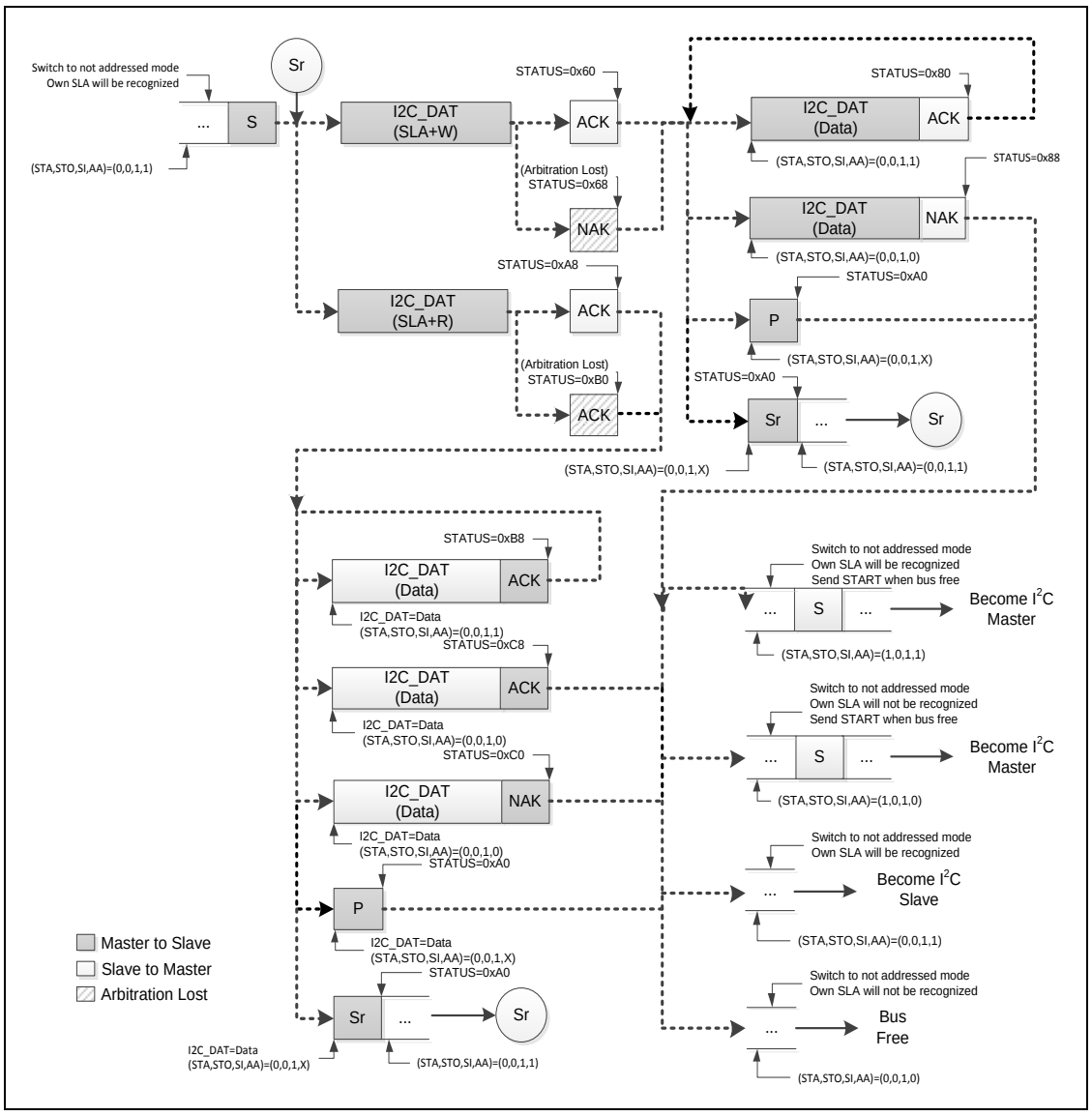


Figure 6.15-12 Save Mode Control Flow

If I^2C is still receiving data in addressed Slave mode but got a STOP or Repeat START, the status code will be 0xA0. User could follow the action for status code 0x88 as shown in the above figure when getting 0xA0 status.

If I^2C is still transmitting data in addressed Slave mode but got a STOP or Repeat START, the status code will be 0xA0. User could follow the action for status code 0xC8 as shown in the above figure when getting 0xA0 status.

Note: After slave gets status of 0x88, 0xC8, 0xC0 and 0xA0, slave can switch to not address mode and own SLA will not be recognized. If entering this status, slave will not receive any I^2C signal or address from master. At this status, I^2C should be reset to leave this status.

General Call (GC) Mode

If the GC bit (I2C_ADDRn [0]) is set, the I²C port hardware will respond to General Call address (00H). User can clear GC bit to disable general call function. When the GC bit is set and the I²C in Slave mode, it can receive the general call address by 0x00 after master send general call address to I²C bus, then it will follow status of GC mode.

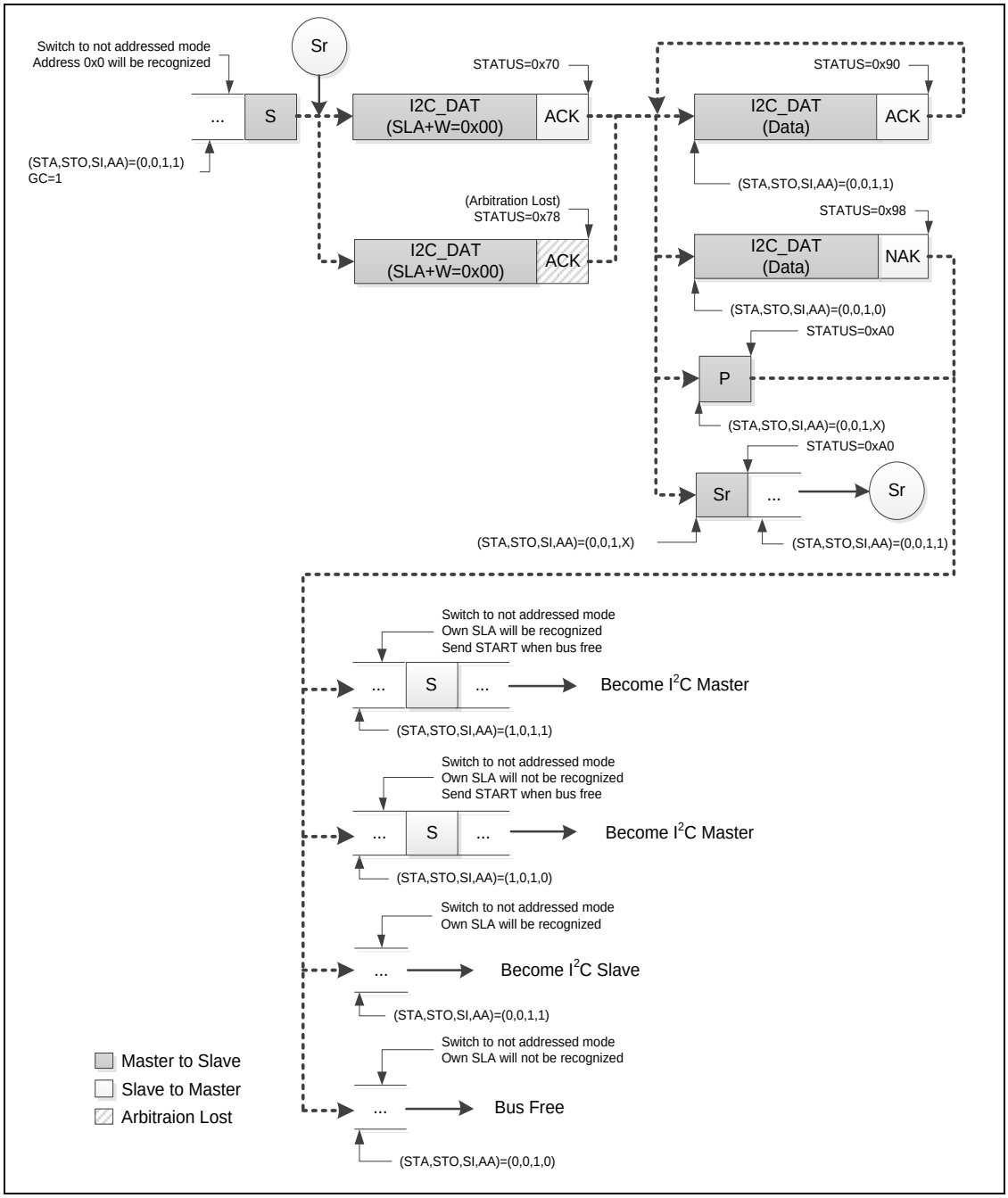


Figure 6.15-13 GC Mode

If I²C is still receiving data in GC mode but got a STOP or Repeat START, the status code will be 0xA0. User could follow the action for status code 0x98 in above figure when getting 0xA0 status.

Note: After slave gets status of 0x98 and 0xA0, slave can switch to not address mode and own

SLA will not be recognized. If entering this status, slave will not receive any I^2C signal or address from master. At this time, I^2C controller should be reset to leave this status.

Multi-Master

In some applications, there are two or more masters on the same I^2C bus to access slaves, and the masters may transmit data simultaneously. The I^2C supports multi-master by including collision detection and arbitration to prevent data corruption.

If for some reason two masters initiate command at the same time, the arbitration procedure determines which master wins and can continue with the command. Arbitration is performed on the SDA signal while the SCL signal is high. Each master checks if the SDA signal on the bus corresponds to the generated SDA signal. If the SDA signal on the bus is low but it should be high, then this master has lost arbitration. The device that has lost arbitration can generate SCL pulses until the byte ends and must then release the bus and go into slave mode. The arbitration procedure can continue until all the data is transferred. This means that in multi-master system each master must monitor the bus for collisions and act accordingly.

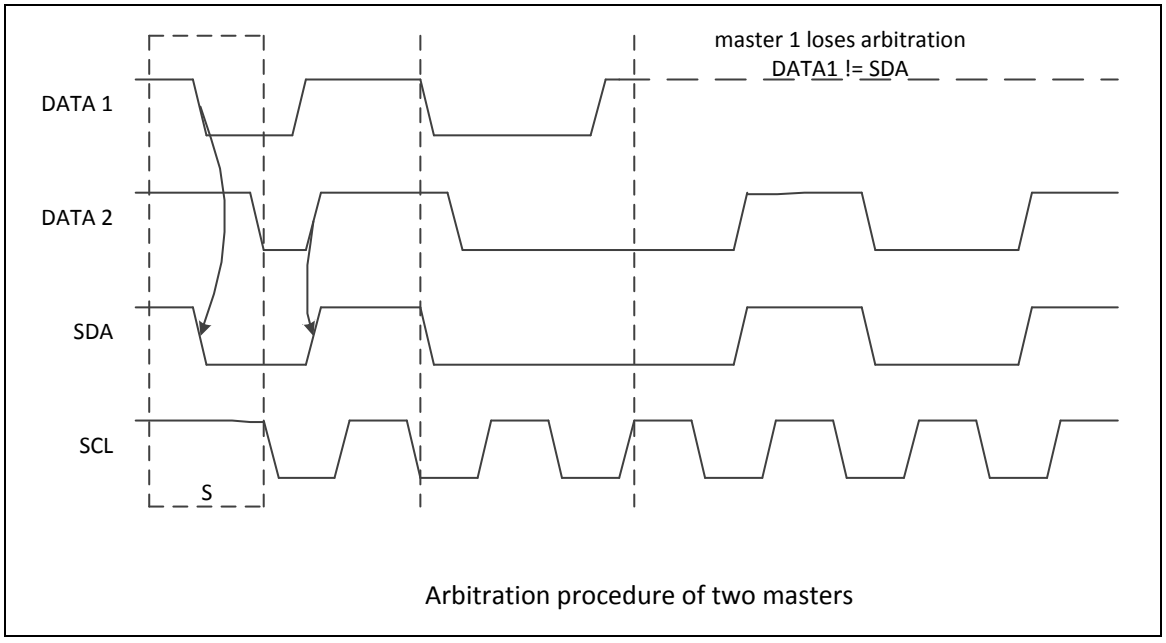


Figure 6.15-14 Arbitration Lost

- When I2C_STATUS = 0x38, an “Arbitration Lost” is received. Arbitration lost event maybe occur during the send START bit, data bits or STOP bit. User could set (STA, STO, SI, AA) = (1, 0, 1, X) to send START again when bus free, or set (STA, STO, SI, AA) = (0, 0, 1, X) to send STOP to back to not addressed Slave mode.

- When I2C_STATUS = 0x00, a “Bus Error” is received. To recover I^2C bus from a bus error, STO should be set and SI should be cleared, and then STO is cleared to release bus.
  - Set (STA, STO, SI, AA) = (0, 1, 1, X) to stop current transfer
  - Set (STA, STO, SI, AA) = (0, 0, 1, X) to release bus

Bus Management (SMBus/PMBus Compatiable)

This section is relevant only when Bus Management feature is supported.

Introduction

The Bus Management is an I^2C interface through which various devices can communicate with each other and with the rest of the system. It is based on I^2C principles of operation. The Bus Management provides a control bus for system and power management related tasks.

This peripheral is compatible with the SMBUS specification rev 2.0 (*http://smbus.org/specs/*) and PMBUS specification rev 1.2 (http://pmbus.org/).

The System Management Bus Specification refers to three types of devices.

● A slave is a device that receives or responds to a command.

● A master is a device that issues commands, generates the clocks and terminates the transfer.

● A host is a specialized master that provides the main interface to the system's CPU. A host must be a master-slave and must support the SMBus host notify protocol. Only one host is allowed in a system.

This Bus Management peripheral is based on I^2C specification rev 2.1.

Device Identification – slave address

Any device that exists on the Bus Management as a slave has a unique address called the Slave Address. For reference, the following addresses are reserved and must not be used by or assign to any Bus Management device. (Refer to SMBus specification for detail information)

Slave Address Bits 7-1	R/W Bit Bit 0	Comment
0000 000	0	General Call Address
0000 000	1	START byte
0000 001	X	CBUS address
0000 010	X	Address reserved for different bus format
0000 011	X	Reserved for future use
0000 1XX	X	Reserved for future use
0101 000	X	Reserved for ACCESS.bus host
0110 111	X	Reserved for ACCESS.bus default address
1111 0XX	X	10-bit slave addressing
1111 1XX	X	Reserved for future use
0001 000	X	SMBus Host
0001 100	X	SMBus Alert Response Address
1100 001	X	SMBus Device Default Address

Table 6-32 Reserved SMBus Address

Bus protocols

There are eleven possible command protocols for any given device. A device may use any or all of the eleven protocols to communicate. The protocols are **Quick CMD**, **Send Byte**, **Receive Byte**, **Write Byte**, **Write Word**, **Read Byte**, **Read Word**, **Process Call**, **Block Read**, **Block Write** and **Block Write-Block Read Process Call**. These protocols should be implemented by the user software. (**For more details of these protocols, refer to SMBus specification ver. 2.0**)

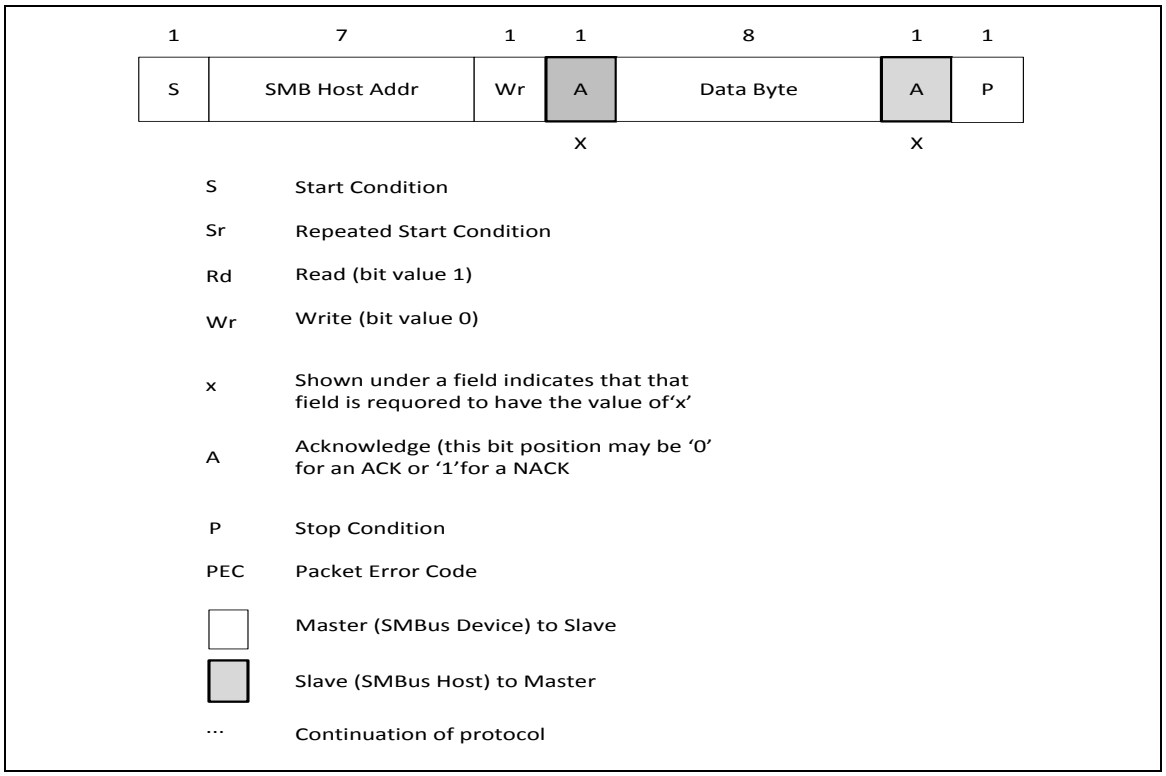


Figure 6.15-15 Bus Management Packet Protocol Diagram Element Key

Address resolution protocol (ARP)

Bus Management slave address conflicts can be resolved by dynamically assigning a new unique address to each slave device. In order to provide a mechanism to isolate each device for the purpose of address assignment each device must implement a unique device identifier (UDID). This 128-bit number is implemented by software.

This peripheral supports the Address Resolution Protocol (ARP). The Bus Management Device Default Address (0b1100 001) is enabled by setting BUSEN (I2C_BUSCTL[7]), BMDEN (I2C_BUSCTL[2]) and ALERTEN (I2C_BUSCTL[4]) bits. The ARP commands should be implemented by the user software. Arbitration is also performed in slave mode for ARP support.

Received CMD and Data acknowledge control

A Bus Management receiver must be able to NACK each received command or data. In order to allow the ACK control in slave mode, the Slave Byte Control mode must be enabled by setting ACKMEN bit (I2C_BUSCTL[0]).

Host Notify protocol

To prevent message coming to the Bus Management host controller from unknown devices in

unknown formats only one method of communication is allowed, a modified form of the Write Word protocol. The standard Write Word protocol is modified by replacing the command code with the alerting device's address.

This peripheral supports the Host Notify protocol by setting the BUSEN (I2C_BUSCTL[7]), BMHEN (I2C_BUSCTL[3]) and ALERTEN (I2C_BUSCTL[4]). In this case the host will acknowledge the Bus Management Host address (0b0001000). This protocol is used when the device acts as a master and the host as a slave.

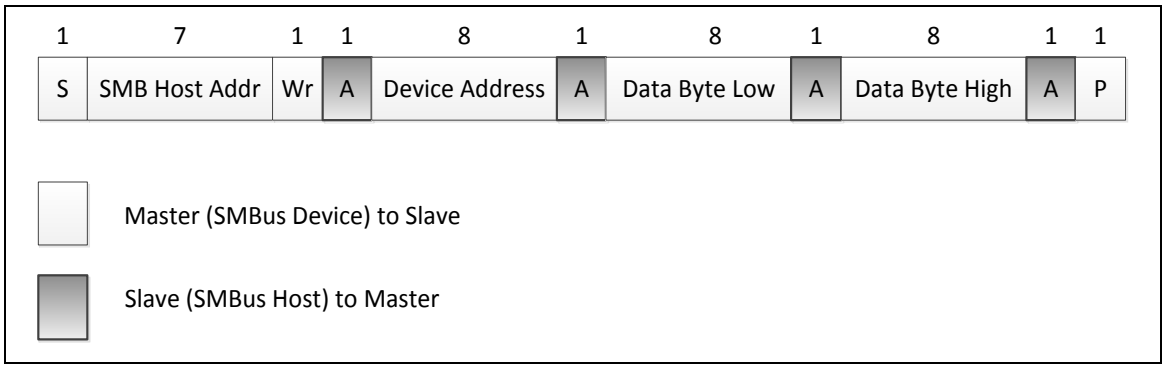


Figure 6.15-167-Bit Addressable Device to Host Communication

Bus Management Alert

The Bus Management ALERT optional signal is supported. A slave-only device can signal the host through the Bus Management ALERT pin (GPA[14]/GPE[10]) that it wants to talk. The host processes the interrupt and simultaneously accesses all Bus Management ALERT pin's devices through the Alert Response Address (0b0001 100). Only the device(s) which pulled Bus Management ALERT pin low will acknowledge the Alert Response Address.

When configured as a slave device(BMHEN=0), the Bus Management ALERT pin is pulled low by setting the ALERTEN bit (I2C_BUSCTL[4]). The Alert Response Address (ARA) is enabled at the same time.

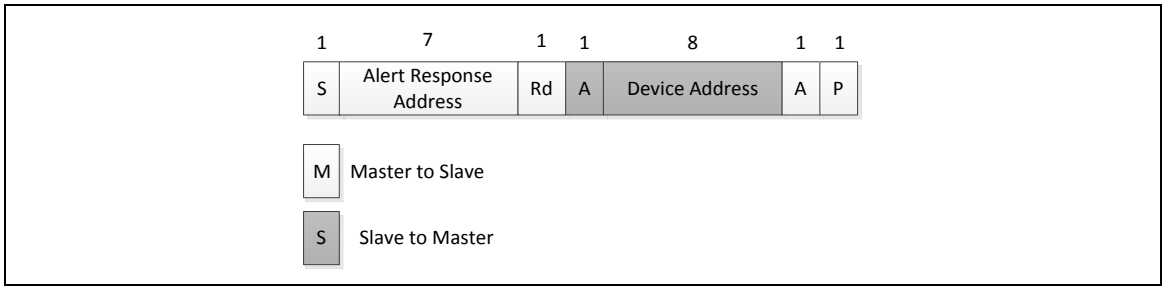


Figure 6.15-177-Bit Addressable Device Responds to an ARA

When configured as a host (BMHEN=1), the ALERT flag (I2C_BUSSTS[3]) is set when a falling edge is detected on the Bus Management ALERT pin and ALERTEN=1. When ALERTEN=0, the ALERT line is considered high even if the external Bus Management ALERT pin is low. If the Bus Management ALERT pin is not needed, the Bus Management ALERT pin can be used as a standard GPIO if ALERTEN = 0;

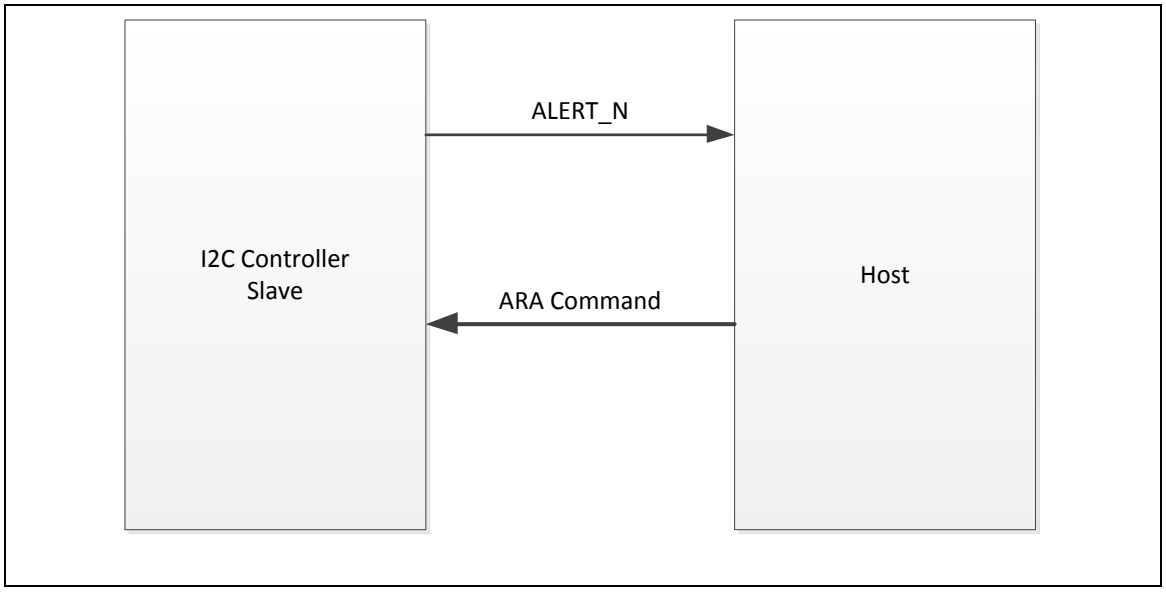


Figure 6.15-18 Bus Management ALERT function

Packet error checking

A packet error checking mechanism has been introduced in the SMBus specification to improve reliability and communication robustness. Packet Error Checking is implemented by appending a Packet Error Code (PEC) at the end of each message transfer. The PEC is calculated by using the $C(x) = x^8 + x^2 + x + 1$ CRC-8 polynomial on all the message bytes (including addresses and read/write bits).

The peripheral embeds a hardware PEC calculator when the PECEN bit (I2C_BUSCTL[1]) is set and allows to send a Not Acknowledge automatically when the received byte does not match with the hardware calculated PEC. The calculated value of PEC also can be read back on I2C_PKTCRC.

Time-out

This peripheral embeds hardware timers in order to be compliant with the 3 time-outs defined in SMBus specification ver. 2.0.

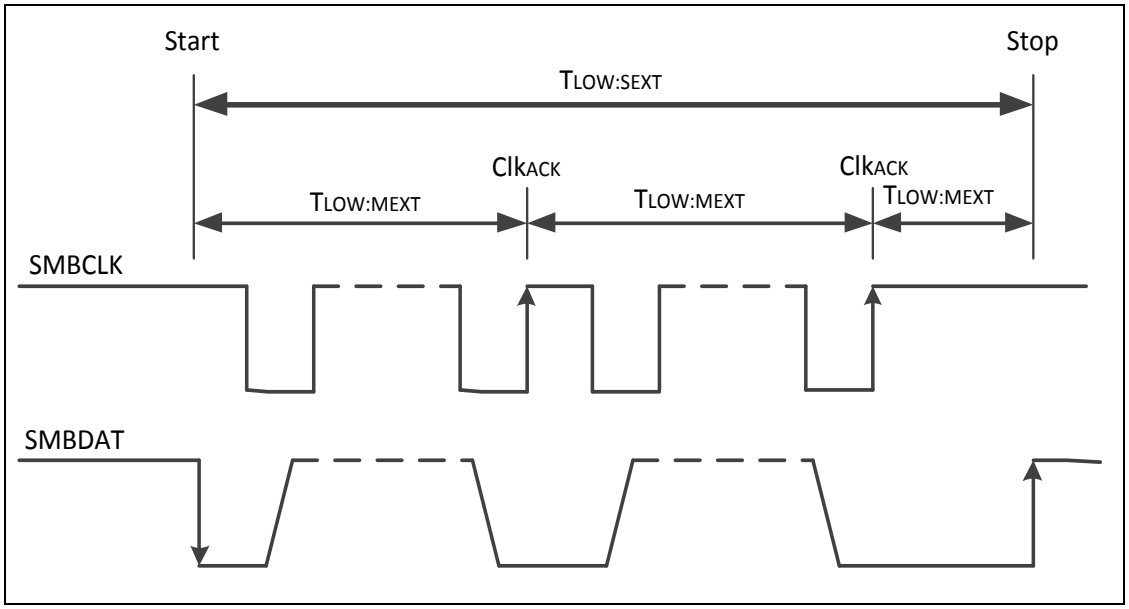


Figure 6.15-19 SM Bus Time Out Timing

Bus management time-out:

1. The SCLK low time-out condition when bus no IDLE

$T_{Time-out}$ = (BUSTO(I2C_BUSTOUT[7:0]) +1) x 16x1024 (14-bit) x T_{PCLK} (if TOCDIV4 = 0).
= (BUSTO(I2C_BUSTOUT[7:0])+1) x 16x1024 (14-bit) x 4 x T_{PCLK} (if TOCDIV4 = 1)

2. The bus idle condition (both SCLK and SDA high) when bus IDLE

$T_{Time-out}$ = (BUSTO(I2C_BUSTOUT[7:0])+ 1) x 4 x T_{PCLK}.

Bus clock low time-out:

In Master mode, the Master cumulative clock low extend time ($T_{LOW:MEXT}$) is detected

In Slave mode, the slave cumulative clock low extend time ($T_{LOW:SEXT}$) is detected

$T_{TLOW:EXT}$ = (CLKTO (I2C_CLKTOUT[7:0])+1) x 16x1024 (14-bit) x T_{PCLK} (if TOCDIV4= 0).
= (CLKTO (I2C_CLKTOUT[7:0])+1) x 16x1024 (14-bit) x 4 x T_{PCLK} (if TOCDIV4= 1)

Bus idle detection

A master can assume that the bus is free if it detects that the clock and data signals have been high for T_{IDLE} greater than $T_{HIGH,MAX}$.

This timing parameter covers the condition where a master has been dynamically added to the bus and may not have detected a state transition on the SMBCLK or SMBDAT lines. In this case, the master must wait long enough to ensure that a transfer is not currently in progress. The peripheral supports a hardware bus idle detection.

6.15.5.3 I²C Protocol Registers

To control I²C port through the following fifteen special function registers: I2C_CTL (control register), I2C_STATUS (status register), I2C_DAT (data register), I2C_ADDRn (address registers, n=0~3),I2C_ ADDRMSKn (address mask registers, n=0~3), I2C_CLKDIV (clock rate register), I2C_TOCTL (Time-out control register), I2C_WKCTL(wake up control register), I2C_WKSTS(wake up status register), I2C_BUSCTL (bus management register), I2C_BUSTCTL

(bus management timer control register), I2C_BUSSTS (bus management status register), I2C_PKTSIZE (TX/RX byte number), I2C_PKTCRC (PEC value register), I2C_BUSTOUT (bus management timer register), and I2C_CLKTOUT (bus management clock low timer register).

Address Registers (ADDR)

The I^2C port is equipped with four slave address registers, I2C_ADDRn (n=0~3). The contents of the register are irrelevant when I^2C is in Master mode. In Slave mode, the bit field ADDR(I2C_ADDRn[7:1]) must be loaded with the chip's own slave address. The I^2C hardware will react if the contents of I2C_ADDRn are matched with the received slave address.

The I^2C ports support the "General Call" function. If the GC bit (I2C_ADDRn [0]) is set the I^2C port hardware will respond to General Call address (00H). Clear GC bit to disable general call function.

When the GC bit is set and the I^2C is in Slave mode, it can receive the general call address by 00H after Master send general call address to I^2C bus, then it will follow status of GC mode.

Slave Address Mask Registers (ADDRMSK)

The I^2C bus controller supports multiple address recognition with four address mask registers I2C_ADDRMSKn (n=0~3). When the bit in the address mask register is set to 1, it means the received corresponding address bit is "Don't care". If the bit is set to 0, it means the received corresponding register bit should be exactly the same as address register.

Data Register (I2C_DAT)

This register contains a byte of serial data to be transmitted or a byte which just has been received. The CPU can be read from or written to the 8-bit (I2C_DAT [7:0]) directly while it is not in the process of shifting a byte. When I^2C is in a defined state and the serial interrupt flag (SI) is set, data in I2C_DAT [7:0] remains stable. While data is being shifted out, data on the bus is simultaneously being shifted in; I2C_DAT [7:0] always contains the last data byte presented on the bus.

The acknowledge bit is controlled by the I^2C hardware and cannot be accessed by the CPU. Serial data is shifted into I2C_DAT [7:0] on the rising edges of serial clock pulses on the SCL line. When a byte has been shifted into I2C_DAT [7:0], the serial data is available in I2C_DAT [7:0], and the acknowledge bit (ACK or NACK) is returned by the control logic during the ninth clock pulse. In order to monitor bus status while sending data, the bus date will be shifted to I2C_DAT[7:0] when sending I2C_DAT[7:0] to bus. In the case of sending data, serial data bits are shifted out from I2C_DAT [7:0] on the falling edge of SCL clocks, and is shifted to I2C_DAT [7:0] on the rising edge of SCL clocks.

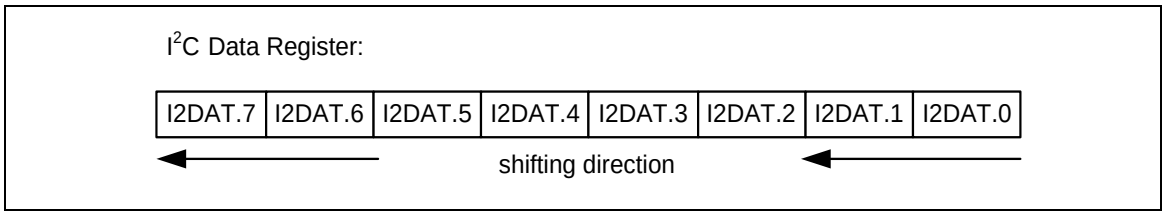


Figure 6.15-20 I^2C Data Shifting Direction

Control Register (I2C_CTL)

The CPU can be read from and written to I2C_CTL [7:0] directly. When the I^2C port is enabled by setting I2CEN (I2C_CTL [6]) to high, the internal states will be controlled by I2C_CTL and I^2C logic hardware.

There are two bits are affected by hardware: the SI bit is set when the I^2C hardware requests a serial interrupt, and the STO bit is cleared when a STOP condition is present on the bus. The STO bit is also cleared when I2CEN = 0.

Once a new status code is generated and stored in I2C_STATUS, the I^2C Interrupt Flag bit SI (I2C_CTL [3]) will be set automatically. If the Enable Interrupt bit INTEN (I2C_CTL [7]) is set at this time, the I^2C interrupt will be generated. The bit field I2C_STATUS[7:0] stores the internal state code, the content keeps stable until SI is cleared by software.

Status Register (I2C_STATUS)

I2C_STATUS [7:0] is an 8-bit read-only register. The bit field I2C_STATUS [7:0] contains the status code and there are 26 possible status codes. All states are listed in Table 6-33. When I2C_STATUS [7:0] is F8H, no serial interrupt is requested. All other I2C_STATUS [7:0] values correspond to the defined I^2C states. When each of these states is entered, a status interrupt is requested (SI = 1). A valid status code is present in I2C_STATUS[7:0] one cycle after SI set by hardware and is still present one cycle after SI reset by software.

In addition, the state 00H stands for a Bus Error, which occurs when a START or STOP condition is present at an incorrect position in the I^2C format frame. A Bus Error may occur during the serial transfer of an address byte, a data byte or an acknowledge bit. To recover I^2C from bus error, STO should be set and SI should be cleared to enter Not Addressed Slave mode. Then STO is cleared to release bus and to wait for a new communication. The I^2C bus cannot recognize stop condition during this action when a bus error occurs.

Master Mode		**Slave Mode**	
STATUS	**Description**	**STATUS**	**Description**
0x08	Start	0xA0	Slave Transmit Repeat Start or Stop
0x10	Master Repeat Start	0xA8	Slave Transmit Address ACK
0x18	Master Transmit Address ACK	0xB0	Slave Transmit Arbitration Lost
0x20	Master Transmit Address NACK	0xB8	Slave Transmit Data ACK
0x28	Master Transmit Data ACK	0xC0	Slave Transmit Data NACK
0x30	Master Transmit Data NACK	0xC8	Slave Transmit Last Data ACK
0x38	Master Arbitration Lost	0x60	Slave Receive Address ACK
0x40	Master Receive Address ACK	0x68	Slave Receive Arbitration Lost
0x48	Master Receive Address NACK	0x80	Slave Receive Data ACK
0x50	Master Receive Data ACK	0x88	Slave Receive Data NACK
0x58	Master Receive Data NACK	0x70	GC mode Address ACK
0x00	Bus error	0x78	GC mode Arbitration Lost
		0x90	GC mode Data ACK
		0x98	GC mode Data NACK
0xF0	If the BMDEN =1 and the ACKMEN bit is enabled, the information of I2C_STATUS will be fixed as 0xF0 in slave receive condition.		
0xF8	Bus Released **Note:** Status “0xF8” exists in both master/slave modes, and it won’t raise interrupt.		

Table 6-33 I^2C Status Code Description

Clock Baud Rate Bits (I2C_CLKDIV)

The data baud rate of I^2C is determines by CLK(I2C_CLKDIV [7:0])register when I^2C is in Master Mode, and it is not necessary in a Slave mode. In the Slave mode, I^2C will automatically synchronize it with any clock frequency from master I^2C device.

The data baud rate of I^2C setting is Data Baud Rate of I^2C = (system clock) / (4x (I2C_CLKDIV [7:0] +1)). If system clock = 16 MHz, the I2C_CLKDIV [7:0] = 40 (28H), the data baud rate of I^2C = 16 MHz/ (4x (40 +1)) = 97.5 Kbits/sec.

Time-out Control Register (I2C_TOCTL)

There is a 14-bit time-out counter which can be used to deal with the I^2C bus hang-up. If the time-out counter is enabled, the counter starts up counting until it overflows (TOIF=1) and generates I^2C interrupt to CPU or stops counting by clearing TOCEN to 0. When time-out counter is enabled, writing 1 to the SI flag will reset counter and re-start up counting after SI is cleared. If I^2C bus hangs up, it causes the I2C_STATUS and flag SI are not updated for a period, the 14-bit time-out counter may overflow and acknowledge CPU the I^2C interrupt. Refer to the Figure 6.15-21 for the 14-bit time-out counter. User may write 1 to clear TOIF to 0.

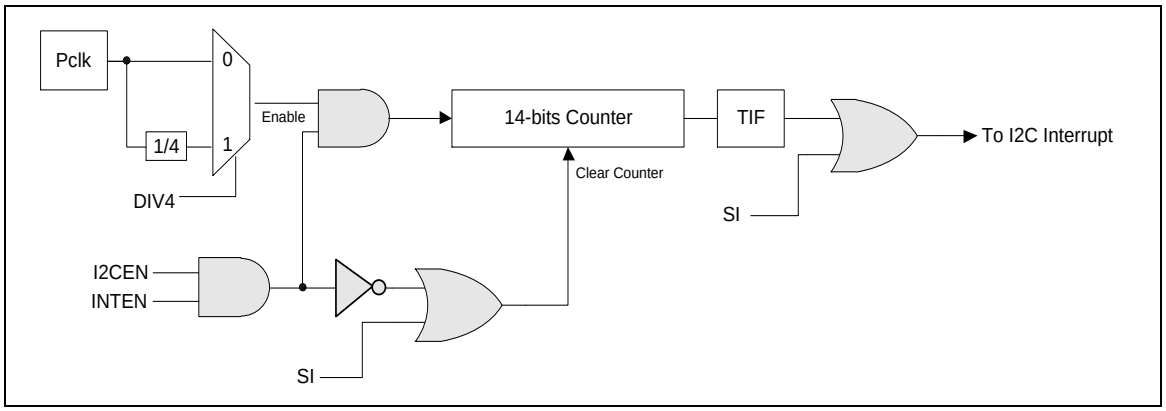


Figure 6.15-21 I^2C Time-out Count Block Diagram

Wake-up Control Register (I2C_WKCTL)

When chip enters Power-down mode, other I^2C master can wake up our chip by addressing our I^2C device, user must configure the related setting before entering Sleep mode. When the chip is woken-up by address match with one of the four address register, the following data will be abandoned at this time.

Wake-up Status Register (I2C_WKSTS)

When system is woken up by other I^2C master device, WKIF is set to indicate this event. User needs write “1” to clear this bit.

Bus Management Control Register (I2C_BUSCTL)

The SM bus management control events are defined in this register. It includes the Acknowledge Control by Manual (ACKMEN (I2C_BUSCTL[0])), Packet Error Checking Enable (PECEN (I2C_BUSCTL[1])), device (BMDEN(I2C_BUSCTL[2])) or host (BMHEN (I2C_BUSCTL[3])) enable in this peripheral device. Both the alert and the suspend function can be set in ALERTEN (I2C_BUSCTL[4]), SCTLOSTS (I2C_BUSCTL[5])) and SCTLOEN (I2C_BUSCTL[6]).

The calculated PEC (when the PECEN is set) value is transmitted or received can be controlled by PECTXEN bit (I2C_BUSCTL[8]).

There is a special bit of ACKM9SI (I2C_BUSCTL[11]). When the ACKMEN is set, there is SI interrupt in the 8th clock input and the user can read the data and status register. If the 8th clock bus is released when the SI interrupt is cleared, there is another SI interrupt event in the 9th clock cycle when this bit is set to 1 to know the bus status in this transaction frame done.

I^2C Byte Number Register (I2C_PKTSIZE)

When the PECEN bit (I2C_BUSCTL[1]) is set. The I^2C controller will calculate the PEC value of the data on the bus. The I2C_PKTSIZE is used to define the data number in the bus. When the counter reach the value of I2C_PKTSIZE, the final PEC value will be transmitted or received automatically when the PECTXEN bit (I2C_BUSCTL[8]) is set.

I^2C PEC VALUR Register (I2C_PKTCRC)

The register indicates the calculated PEC value of data on the I^2C bus. The detail of information is defined the PEC section of SM Bus.

I²C Bus Management Timer and I²C CLock Low Timer Register (I2C_BUSTOUT/ I2C_CLKTOUT)

Both of the definition of these registers are described in the time-out section of SM Bus.

6.15.5.4 Example for Random Read on EEPROM

The following steps are used to configure the I²C0 related registers when using I²C to read data from EEPROM.

1. Set I²C0 the multi-function pin in the SYS_GPA_MFPL or SYS_GPD_MFPL or SYS_GPE_MFPH or SYS_GPA_MFPH registers as SCL and SDA pins.
2. Enable I²C0 APB clock, I2C0CKEN=1 in the “CLK_APBCLK0” register.
3. Set I2C0RST=1 to reset I²C0 controller then set I²C0 controller to normal operation, I2C0RST=0 in the “SYS_IPRST1” register.
4. Set I2CEN=1 to enable I²C0 controller in the “I2C_CTL” register.
5. Give I²C0 clock a divided register value for I²C clock rate in the “I2C_CLKDIV”.
6. Set SETENA=0x00000040 in the “NVIC_ISER2” register to set I²C0 IRQ.
7. Set INTEN=1 to enable I²C0 Interrupt in the “I2C_CTL” register.
8. Set I²C0 address registers “I2C_ADDR0 ~ I2C_ADDR3”.

Random read operation is one of the methods of access EEPROM. The method allows the master to access any address of EEPROM space. The Figure 6.15-22 shows the EEPROM random read operation.

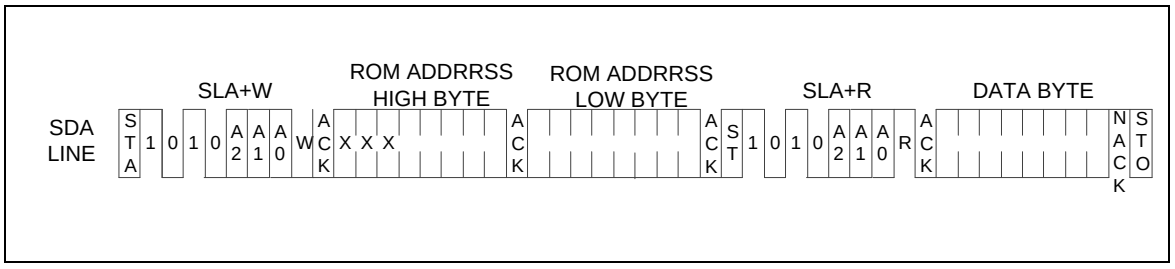


Figure 6.15-22 EEPROM Random Read

The Figure 6.15-23 shows how to use I²C controller to implement the protocol of EEPROM random read.

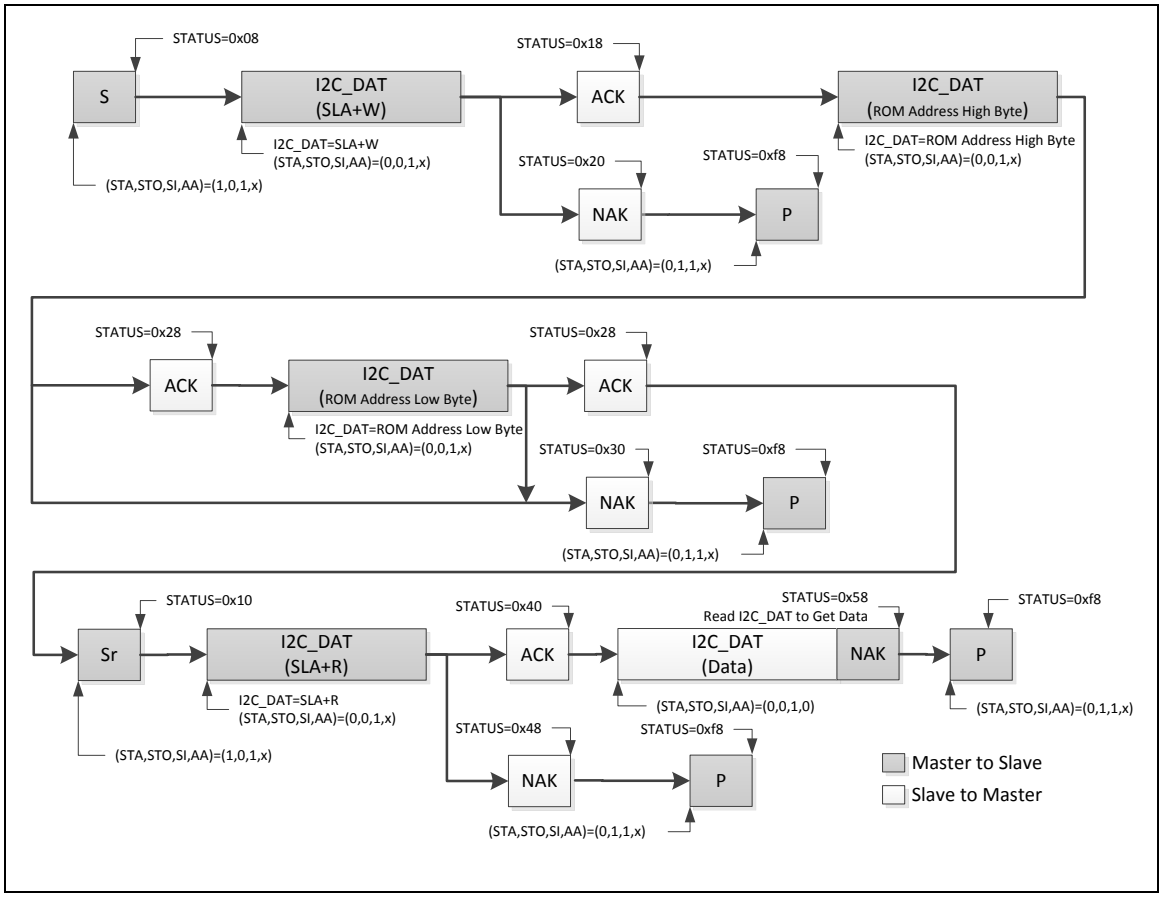


Figure 6.15-23 Protocol of EEPROM Random Read

The I^2C controller, which is a master, sends START to bus. Then, it sends a SLA+W (Slave address + Write bit) to EERPOM followed by two bytes data address to set the EEPROM address to read. Finally, a Repeat START followed by SLA+R is sent to read the data from EEPROM.

6.15.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
I²C Base Address: **I2Cn_BA = 0x4008_0000 + (0x1000 *n)** **n= 0,1**				
I2C_CTL	I2Cn_BA+0x00	R/W	I²C Control Register	0x0000_0000
I2C_ADDR0	I2Cn_BA+0x04	R/W	I²C Slave Address Register0	0x0000_0000
I2C_DAT	I2Cn_BA+0x08	R/W	I²C Data Register	0x0000_0000
I2C_STATUS	I2Cn_BA+0x0C	R	I²C Status Register	0x0000_00F8
I2C_CLKDIV	I2Cn_BA+0x10	R/W	I²C Clock Divided Register	0x0000_0000
I2C_TOCTL	I2Cn_BA+0x14	R/W	I²C Time-out Control Register	0x0000_0000
I2C_ADDR1	I2Cn_BA+0x18	R/W	I²C Slave Address Register1	0x0000_0000
I2C_ADDR2	I2Cn_BA+0x1C	R/W	I²C Slave Address Register2	0x0000_0000
I2C_ADDR3	I2Cn_BA+0x20	R/W	I²C Slave Address Register3	0x0000_0000
I2C_ADDRMSK0	I2Cn_BA+0x24	R/W	I²C Slave Address Mask Register0	0x0000_0000
I2C_ADDRMSK1	I2Cn_BA+0x28	R/W	I²C Slave Address Mask Register1	0x0000_0000
I2C_ADDRMSK2	I2Cn_BA+0x2C	R/W	I²C Slave Address Mask Register2	0x0000_0000
I2C_ADDRMSK3	I2Cn_BA+0x30	R/W	I²C Slave Address Mask Register3	0x0000_0000
I2C_WKCTL	I2Cn_BA+0x3C	R/W	I²C Wake-up Control Register	0x0000_0000
I2C_WKSTS	I2Cn_BA+0x40	R/W	I²C Wake-up Status Register	0x0000_0000
I2C_BUSCTL	I2Cn_BA+0x44	R/W	I²C Bus Management Control Register	0x0000_0000
I2C_BUSTCTL	I2Cn_BA+0x48	R/W	I²C Bus Management Timer Control Register	0x0000_0000
I2C_BUSSTS	I2Cn_BA+0x4C	R/W	I²C Bus Management Status Register	0x0000_0000
I2C_PKTSIZE	I2Cn_BA+0x50	R/W	I²C Packet Error Checking Byte Number Register	0x0000_0000
I2C_PKTCRC	I2Cn_BA+0x54	R	I²C Packet Error Checking Byte Value Register	0x0000_0000
I2C_BUSTOUT	I2Cn_BA+0x58	R/W	I²C Bus Management Timer Register	0x0000_0005
I2C_CLKTOUT	I2Cn_BA+0x5C	R/W	I²C Bus Management Clock Low Timer Register	0x0000_0005

6.15.7 Register Description

I²C Control Register (I2C_CTL)

Register	Offset	R/W	Description	Reset Value
I2C_CTL	I2Cn_BA+0x00	R/W	I²C Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
INTEN	I2CEN	STA	STO	SI	AA	Reserved	

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**INTEN**	**Enable Interrupt** 0 = I²C interrupt Disabled. 1 = I²C interrupt Enabled.
[6]	**I2CEN**	**I²C Controller Enable Bit** Set to enable I²C serial function controller. When I2CEN=1 the I²C serial function enable. The multi-function pin function must set to SDA, and SCL of I²C function first. 0 = I2C Controller Disabled. 1 = I2C Controller Enabled.
[5]	**STA**	**I²C START Control** Setting STA to logic 1 to enter Master mode, the I²C hardware sends a START or repeat START condition to bus when the bus is free.
[4]	**STO**	**I²C STOP Control** In Master mode, setting STO to transmit a STOP condition to bus then I²C controller will check the bus condition if a STOP condition is detected. This bit will be cleared by hardware automatically.
[3]	**SI**	**I²C Interrupt Flag** When a new I²C state is present in the I2C_STATUS register, the SI flag is set by hardware. If bit INTEN (I2C_CTL [7]) is set, the I²C interrupt is requested. SI must be cleared by software. Clear SI by writing 1 to this bit. For ACKMEN is set in slave read mode, the SI flag is set in 8th clock period for user to confirm the acknowledge bit and 9th clock period for user to read the data in the data buffer.
[2]	**AA**	**Assert Acknowledge Control** When AA =1 prior to address or data is received, an acknowledged (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line when 1.). A slave is acknowledging the address sent from master, 2.) The receiver devices are acknowledging the data sent by transmitter. When AA=0 prior to address or data received, a Not acknowledged (high level to SDA) will be returned during the acknowledge clock pulse on

		the SCL line.
[1:0]	**Reserved**	Reserved.

I²C Data Register (I2C_DAT)

Register	Offset	R/W	Description	Reset Value
I2C_DAT	I2Cn_BA+0x08	R/W	I²C Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DAT							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**DAT**	**I²C Data** Bit [7:0] is located with the 8-bit transferred/received data of I²C serial port.

I²C Status Register (I2C_STATUS)

Register	Offset	R/W	Description	Reset Value
I2C_STATUS	I2Cn_BA+0x0C	R	I²C Status Register	0x0000_00F8

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
STATUS							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**STATUS**	**I²C Status** The three least significant bits are always 0. The five most significant bits contain the status code. There are 28 possible status codes. When the content of I2C_STATUS is F8H, no serial interrupt is requested. Others I2C_STATUS values correspond to defined I²C states. When each of these states is entered, a status interrupt is requested (SI = 1). A valid status code is present in I2C_STATUS one cycle after SI is set by hardware and is still present one cycle after SI has been reset by software. In addition, states 00H stands for a Bus Error. A Bus Error occurs when a START or STOP condition is present at an illegal position in the formation frame. Example of illegal position are during the serial transfer of an address byte, a data byte or an acknowledge bit. **Note:** 1. If the BUSEN and ACKMEN are enabled in slave received mode, there is SI interrupt in the 8th clock. The user can read the I2C_STATUS = 0xf0 for the function condition has done. 2. If the BUSEN and PECEN are enabled, the status of PECERR, I2C_BUSSTS[3], is used to substitute for I2C_STATUS to check the ACK status in the last frame when the byte count done interrupt has active and the PEC frame has been transformed.

I²C Clock Divided Register (I2C_CLKDIV)

Register	Offset	R/W	Description	Reset Value
I2C_CLKDIV	I2Cn_BA+0x10	R/W	I²C Clock Divided Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DIVIDER							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**DIVIDER**	**I²C Clock Divided** Indicates the I²C clock rate: Data Baud Rate of I²C = (system clock) / (4x (I2C_CLKDIV+1)). **Note:** The minimum value of I2C_CLKDIV is 4.

I²C Time-out Control Register (I2C_TOCTL)

Register	Offset	R/W	Description	Reset Value
I2C_TOCTL	I2Cn_BA+0x14	R/W	I²C Time-out Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					TOCEN	TOCDIV4	TOIF

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**TOCEN**	**Time-out Counter Enable Bit** When Enabled, the 14-bit time-out counter will start counting when SI is clear. Setting flag SI to '1' will reset counter and re-start up counting after SI is cleared. 0 = Time-Out Counter Disabled. 1 = Time-Out Counter Enabled.
[1]	**TOCDIV4**	**Time-out Counter Input Clock Divided by 4** When Enabled, The time-out period is extend 4 times. 0 = Time-Out Counter Input Clock Divided Disabled. 1 = Time-Out Counter Input Clock Divided Enabled.
[0]	**TOIF**	**Time-out Flag** This bit is set by hardware when I²C time-out happened and it can interrupt CPU if I²C interrupt enable bit (INTEN) is set to 1. **Note:** Software can write 1 to clear this bit.

I²C Slave Address Register (ADDRx)

Register	Offset	R/W	Description	Reset Value
I2C_ADDR0	I2Cn_BA+0x04	R/W	I²C Slave Address Register0	0x0000_0000
I2C_ADDR1	I2Cn_BA+0x18	R/W	I²C Slave Address Register1	0x0000_0000
I2C_ADDR2	I2Cn_BA+0x1C	R/W	I²C Slave Address Register2	0x0000_0000
I2C_ADDR3	I2Cn_BA+0x20	R/W	I²C Slave Address Register3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDR							**GC**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:1]	**ADDR**	**I²C Address** The content of this register is irrelevant when I²C is in Master mode. In the slave mode, the seven most significant bits must be loaded with the chip's own address. The I²C hardware will react if either of the address is matched.
[0]	**GC**	**General Call Function** 0 = General Call Function Disabled. 1 = General Call Function Enabled.

I²C Slave Address Mask Register (ADDRMSKx)

Register	Offset	R/W	Description	Reset Value
I2C_ADDRMSK0	I2Cn_BA+0x24	R/W	I²C Slave Address Mask Register0	0x0000_0000
I2C_ADDRMSK1	I2Cn_BA+0x28	R/W	I²C Slave Address Mask Register1	0x0000_0000
I2C_ADDRMSK2	I2Cn_BA+0x2C	R/W	I²C Slave Address Mask Register2	0x0000_0000
I2C_ADDRMSK3	I2Cn_BA+0x30	R/W	I²C Slave Address Mask Register3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ADDRMSK							**Reserved**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:1]	**ADDRMSK**	**I²C Address Mask** 0 = Mask Disabled (the received corresponding register bit should be exact the same as address register.). 1 = Mask Enabled (the received corresponding address bit is don't care.). I²C bus controllers support multiple address recognition with four address mask register. When the bit in the address mask register is set to one, it means the received corresponding address bit is don't-care. If the bit is set to zero, that means the received corresponding register bit should be exact the same as address register.
[0]	**Reserved**	Reserved.

I²C Wake-up Control Register (I2C_WKCTL)

Register	Offset	R/W	Description	Reset Value
I2C_WKCTL	I2Cn_BA+0x3C	R/W	I²C Wake-up Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							WKEN

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**WKEN**	**I²C Wake-up Enable Bit** 0= I²C wake-up function Disabled. 1= I²C wake-up function Enabled.

I²C Wake-up Status Register (I2C_WKSTS)

Register	Offset	R/W	Description	Reset Value
I2C_WKSTS	I2Cn_BA+0x40	R/W	I²C Wake-up Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							WKIF

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	WKIF	**I²C Wake-up Flag** When chip is woken up from Power-down mode by I²C, this bit is set to 1. Software can write 1 to clear this bit.

I²C Bus Manage Control Register (I2C_BUSCTL)

Register	Offset	R/W	Description	Reset Value
I2C_BUSCTL	I2Cn_BA+0x44	R/W	I²C Bus Management Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				ACKM9SI	PECCLR	TIDLE	PECTXEN
7	6	5	4	3	2	1	0
BUSEN	SCTLOEN	SCTLOSTS	ALERTEN	BMHEN	BMDEN	PECEN	ACKMEN

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11]	**ACKM9SI**	**Acknowledge Manual Enable Extra SI Interrupt** 0 = There is no SI interrupt in the 9th clock cycle when the BUSEN =1 and ACKMEN =1. 1 = There is SI interrupt in the 9th clock cycle when the BUSEN =1 and ACKMEN =1.
[10]	**PECCLR**	**PEC Clear at Repeat Start** The calculation of PEC starts when PECEN is set to 1 and it is clear when the STA or STO bit is detected. This PECCLR bit is used to enable the condition of REPEAT START can clear the PEC calculation. 0 = The PEC calculation is cleared by “Repeat Start” function is Disabled. 1 = The PEC calculation is cleared by “Repeat Start” function is Enabled.
[9]	**TIDLE**	**Timer Check in Idle State** The BUSTOUT is used to calculate the time-out of clock low in bus active and the idle period in bus Idle. This bit is used to define which condition is enabled. 0 = The BUSTOUT is used to calculate the clock low period in bus active. 1 = The BUSTOUT is used to calculate the IDLE period in bus Idle. **Note:** The BUSY (I2C_BUSSTS[0]) indicate the current bus state.
[8]	**PECTXEN**	**Packet Error Checking Byte Transmission/Reception** This bit is set by software, and cleared by hardware when the PEC is transferred, or when a STOP condition or an Address Matched is received 0 = No PEC transfer. 1 = PEC transmission/reception is requested. **Note:** 1.This bit has no effect in slave mode when ACKMEN =0.
[7]	**BUSEN**	**BUS Enable Bit** 0 = The system management function is Disabled. 1 = The system management function is Enable. **Note:** When the bit is enabled, the internal 14-bit counter is used to calculate the time out event of clock low condition.
[6]	**SCTLOEN**	**Suspend or Control Pin Output Enable Bit**

		0 = The SUSCON pin in input. 1 = The output enable is active on the SUSCON pin.
[5]	**SCTLOSTS**	**Suspend/Control Data Output Status** 0 = The output of SUSCON pin is low. 1 = The output of SUSCON pin is high.
[4]	**ALERTEN**	**Bus Management Alert Enable Bit** Device Mode (BMHEN =0). 0 = Release the BM_ALERT pin high and Alert Response Header disabled: 0001100x followed by NACK if both of BMDEN and ACKMEN are enabled. 1 = Drive BM_ALERT pin low and Alert Response Address Header enables: 0001100x followed by ACK if both of BMDEN and ACKMEN are enabled. Host Mode (BMHEN =1). 0 = BM_ALERT pin not supported. 1 = BM_ALERT pin supported.
[3]	**BMHEN**	**Bus Management Host Enable Bit** 0 = Host function Disabled. 1 = Host function Enabled and the SUSCON will be used as CONTROL function.
[2]	**BMDEN**	**Bus Management Device Default Address Enable Bit** 0 = Device default address Disable. When the address 0'b1100001x coming and the both of BMDEN and ACKMEN are enabled, the device responses NACKed 1 = Device default address Enabled. When the address 0'b1100001x coming and the both of BMDEN and ACKMEN are enabled, the device responses ACKed.
[1]	**PECEN**	**Packet Error Checking Calculation Enable Bit** 0 = Packet Error Checking Calculation Disabled. 1 = Packet Error Checking Calculation Enabled.
[0]	**ACKMEN**	**Acknowledge Control by Manual** In order to allow ACK control in slave reception including the command and data, slave byte control mode must be enabled by setting the ACKMEN bit. 0 = Slave byte control Disabled. 1 = Slave byte control Enabled. The 9th bit can response the ACK or NACK according the received data by user. When the byte is received, stretching the SCLK signal low between the 8th and 9th SCLK pulse. **Note:** If the BMDEN =1 and this bit is enabled, the information of I2C_STATUS will be fixed as 0xF0 in slave receive condition.

I²C Bus Management Timer Control Register (I2C_BUSTCTL)

Register	Offset	R/W	Description	Reset Value
I2C_BUSTCTL	I2Cn_BA+0x48	R/W	I²C Bus Management Timer Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		PECIEN	TORSTEN	CLKTOIEN	BUSTOIEN	CLKTOEN	BUSTOEN

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**PECIEN**	**Packet Error Checking Byte Count Done Interrupt Enable Bit** 0 = Indicates the byte count done interrupt is Disabled. 1 = Indicates the byte count done interrupt is Enabled. **Note:** This bit is used in PECEN =1.
[4]	**TORSTEN**	**Time Out Reset Enable Bit** 0 = Indicates the I²C state machine reset is Disable. 1 = Indicates the I²C state machine reset is Enable. (The clock and data bus will be released to high)
[3]	**CLKTOIEN**	**Extended Clock Time Out Interrupt Enable Bit** 0 = Indicates the time extended interrupt is Disabled. 1 = Indicates the time extended interrupt is Enabled.
[2]	**BUSTOIEN**	**Time-out Interrupt Enable Bit** BUSY =1. 0 = Indicates the SCLK low time-out interrupt is Disabled. 1 = Indicates the SCLK low time-out interrupt is Enabled. BUSY =0. 0 = Indicates the bus IDLE time-out interrupt is Disabled. 1 = Indicates the bus IDLE time-out interrupt is Enabled.
[1]	**CLKTOEN**	**Cumulative Clock Low Time Out Enable Bit** 0 = Indicates the cumulative clock low time-out detection is Disabled. 1 = Indicates the cumulative clock low time-out detection is Enabled. For Master, it calculates the period from START to ACK For Slave, it calculates the period from START to STOP
[0]	**BUSTOEN**	**Bus Time Out Enable Bit**

		0 = Indicates the bus clock low time-out detection is Disabled. 1 = Indicates the bus clock low time-out detection is Enabled (bus clock is low for more than TTime-out (in BIDLE=0) or high more than TTime-out(in BIDLE =1).

I²C Bus Management Status Register (I2C_BUSSTS)

Register	Offset	R/W	Description	Reset Value
I2C_BUSSTS	I2Cn_BA+0x4C	R/W	I²C Bus Management Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	CLKTO	BUSTO	SCTLDIN	ALERT	PECERR	BCDONE	BUSY

Bits	Description	
[31:6]	**Reserved**	Reserved.
[6]	**CLKTO**	**Clock Low Cumulate Time-out Status** 0 = Indicates that the cumulative clock low is no any time-out. 1 = Indicates that the cumulative clock low time-out occurred. **Note:** Software can write 1 to clear this bit.
[5]	**BUSTO**	**Bus Time-out Status** 0 = Indicates that there is no any time-out or external clock time-out. 1 = Indicates that a time-out or external clock time-out occurred. In bus busy, the bit indicates the total clock low time-out event occurred otherwise, it indicates the bus idle time-out event occurred. **Note:** Software can write 1 to clear this bit.
[4]	**SCTLDIN**	**Bus Suspend or Control Signal Input Status** 0 = The input status of SUSCON pin is 0. 1 = The input status of SUSCON pin is 1.
[3]	**ALERT**	**SMBus Alert Status** Device Mode (BMHEN =0). 0 = Indicates SMALERT pin state is low. 1 = Indicates SMALERT pin state is high. Host Mode (BMHEN =1). 0 = No SMBALERT event. 1 = Indicates there is SMBALERT event (falling edge) is detected in SMALERT pin when the BMHEN = 1 (SMBus host configuration) and the ALERTEN = 1. **Note:** 1. The SMALERT pin is an open-drain pin, the pull-high resistor is must in the system. 2. Software can write 1 to clear this bit.
[2]	**PECERR**	**PEC Error in Reception** 0 = Indicates the PEC value equal the received PEC data packet. 1 = Indicates the PEC value doesn't match the receive PEC data packet. **Note:** Software can write 1 to clear this bit.

[1]	**BCDONE**	**Byte Count Transmission/Receive Done** 0 = Indicates the transmission/ receive is not finished when the PECEN is set. 1 = Indicates the transmission/ receive is finished when the PECEN is set. **Note:** Software can write 1 to clear this bit.
[0]	**BUSY**	**Bus Busy** Indicates that a communication is in progress on the bus. It is set by hardware when a START condition is detected. It is cleared by hardware when a STOP condition is detected 0 = The bus is IDLE (both SCLK and SDA High). 1 = The bus is busy.

I²C Byte Number Register (I2C_PKTSIZE)

Register	Offset	R/W	Description	Reset Value
I2C_PKTSIZE	I2Cn_BA+0x50	R/W	I²C Packet Error Checking Byte Number Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
PLDSIZE							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**PLDSIZE**	**Transfer Byte Number** The transmission or receive byte number in one transaction when the PECEN is set. The maximum transaction or receive byte is 255 Bytes. Notice: The byte number counting includes address, command code, and data frame.

I²C PEC Value Register (I2C_PKTCRC)

Register	Offset	R/W	Description	Reset Value
I2C_PKTCRC	I2Cn_BA+0x54	R	I²C Packet Error Checking Byte Value Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
PECCRC							

Bits	Description	
[31:8]	Reserved	Reserved.
[7:0]	PECCRC	**Packet Error Checking Byte Value** This byte indicates the packet error checking content after transmission or receive byte count by using the $C(x) = x^8 + x^2 + x + 1$. It is read only.

I²C Bus Management Timer Register (I2C_BUSTOUT)

Register	Offset	R/W	Description	Reset Value
I2C_BUSTOUT	I2Cn_BA+0x58	R/W	I²C Bus Management Timer Register	0x0000_0005

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
BUSTO							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**BUSTO**	**Bus Management Time-out Value** Indicate the bus time-out value in bus is IDLE or SCLK low. **Note:** If the user wants to revise the value of BUSTOUT, the TORSTEN (I2C_BUSTCTL[4]) bit shall be set to 1 and clear to 0 first in the BUSEN(I2C_BUSCTL[7]) is set.

I²C Clock Low Timer Register (I2C_CLKTOUT)

Register	Offset	R/W	Description	Reset Value
I2C_CLKTOUT	I2Cn_BA+0x5C	R/W	I²C Bus Management Clock Low Timer Register	0x0000_0005

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CLKTO							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**CLKTO**	**Bus Clock Low Timer** The field is used to configure the cumulative clock extension time-out. **Note:** If the user wants to revise the value of CLKLTOUT, the TORSTEN bit shall be set to 1 and clear to 0 first in the BUSEN is set.

6.16 Serial Peripheral Interface (SPI)

6.16.1 Overview

The Serial Peripheral Interface (SPI) applies to synchronous serial data communication and allows full duplex transfer. Devices communicate in Master/Slave mode with the 4-wire bi-direction interface. The NuMicro® M4TK series contains up to three sets of SPI controllers performing a serial-to-parallel conversion on data received from a peripheral device, and a parallel-to-serial conversion on data transmitted to a peripheral device. Each SPI controller can be configured as a master or a slave device.

SPI0 controller supports 2-bit Transfer mode to perform full-duplex 2-bit data transfer and also supports Dual and Quad I/O Transfer mode. SPI1 and SPI2 controller also support I^2S mode to connect external audio CODEC.

6.16.2 Features

- SPI Mode
  - Up to three sets of SPI controllers
  - Supports Master or Slave mode operation
  - Supports 2-bit Transfer mode
  - Supports Dual and Quad I/O Transfer mode for SPI0
  - Configurable bit length of a transaction word from 8 to 32-bit
  - Provides separate 4-/8-level depth transmit and receive FIFO buffers
  - Supports MSB first or LSB first transfer sequence
  - Supports Byte Reorder function
  - Supports PDMA transfer
  - Supports 3-Wire, no slave selection signal, bi-direction interface
- I^2S Mode for SPI1 and SPI2
  - Supports Master or Slave
  - Capable of handling 8-, 16-, 24- and 32-bit word sizes
  - Provides separate 4-level depth transmit and receive FIFO buffers
  - Supports monaural and stereo audio data
  - Supports PCM mode A, PCM mode B, I^2S and MSB justified data format
  - Supports PDMA transfer

6.16.3 Block Diagram

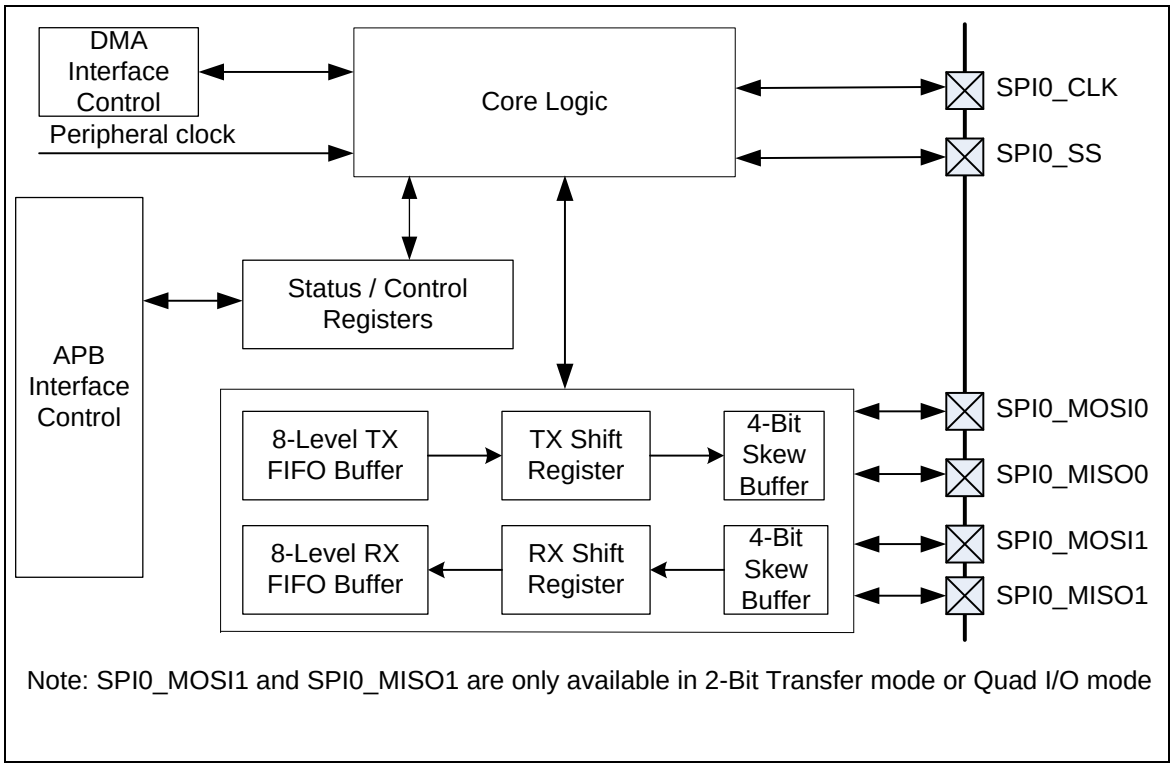


Figure 6.16-1 SPI Block Diagram (SPI0)

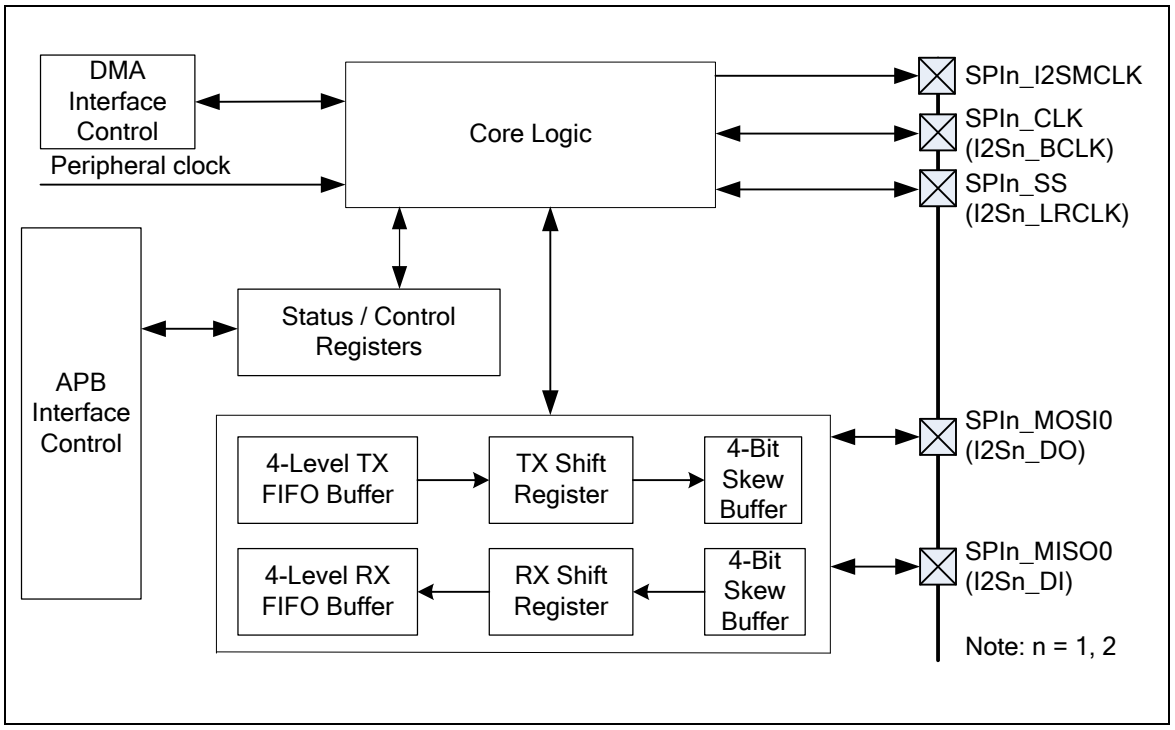


Figure 6.16-2 SPI Block Diagram (SPI1/2)

TX FIFO Buffer:

The transmit FIFO buffer is a 4-/8-level depth, 32-bit wide, first-in, first-out register buffer. The data can be written to the transmit FIFO buffer in advance through software by writing the SPI_TX register.

RX FIFO Buffer:

The received FIFO buffer is also a 4-/8-level depth, 32-bit wide, first-in, first-out register buffer. The receive control logic will store the received data to this buffer. The FIFO buffer data can be read from SPI_RX register by software.

TX Shift Register:

The transmit shift register is a 32-bit wide register buffer. The transmit data is loaded from the TX FIFO buffer and shifted out bit-by-bit to the skew buffer.

RX Shift Register:

The receive shift register is also a 32-bit wide register buffer. The receive data is shifted in bit-by-bit from the skew buffer and is loaded into RX FIFO buffer when a transaction done.

Skew Buffer:

The skew buffer is a 4-level 1-bit buffer.

For transmitting, it is written from shift register by peripheral clock and read out by SPI bus clock. Three bit data is loaded into this buffer first and the 4th bit data is written into the buffer after 1-bit data is read out by the SPI bus clock.

For receiving, the serial data in the bus of SPIn_MOSI (in slave mode) is written into the skew buffer basing on the SPI bus clock and it is read out and written into the RX shift register by SPI peripheral clock after there is no empty in the RX skew buffer.

6.16.4 Basic Configuration

The basic configurations of SPI0 are as follows:

- SPI0 pins are configured in SYS_GPB_MFPL or SYS_GPE_MFPH registers.
- Select the source of SPI0 peripheral clock on SPI0SEL (CLK_CLKSEL2[3:2]).
- Enable SPI0 peripheral clock in SPI0CKEN (CLK_APBCLK0[12]).
- Reset SPI0 controller in SPI0RST (SYS_IPRST1[12]).

The basic configurations of SPI1 are as follows:

- SPI1 pins are configured in SYS_GPA_MFPL**,** SYS_GPB_MFPL, SYS_GPD_MFPL or SYS_GPE_MFPH registers.
- Select the source of SPI1 peripheral clock on SPI1SEL (CLK_CLKSEL2[5:4]).
- Enable SPI1 peripheral clock in SPI1CKEN (CLK_APBCLK0[13]).
- Reset SPI1 controller in SPI1RST (SYS_IPRST1[13]).

The basic configurations of SPI2 are as follows:

- SPI2 pins are configured in SYS_GPC_MFPL, SYS_GPC_MFPH, SYS_GPD_MFPH or SYS_GPE_MFPL registers.
- Select the source of SPI2 peripheral clock on SPI2SEL (CLK_CLKSEL2[7:6]).
- Enable SPI2 peripheral clock in SPI2CKEN (CLK_APBCLK0[14]).
- Reset SPI2 controller in SPI2RST (SYS_IPRST1[14]).

SPI/I^{2}S Interface Controller Pin description is shown as Table 6-34:

Pin	SPI Mode	I^{2}S Mode
SPIn_SS	SPI slave selection pin	I^{2}S left/right channel synchronization clock pin (I2Sn_LRCLK)
SPIn_CLK	SPI clock pin	I^{2}S bit clock pin (I2Sn_BCLK)
SPIn_MISO	SPI master input or slave output pin	I^{2}S data input pin (I2Sn_DI)
SPIn_MOSI	SPI master output or slave input pin	I^{2}S data output pin (I2Sn_DO)

Table 6-34 SPI/I^{2}S Interface Controller Pin

6.16.5 Functional Description

6.16.5.1 Terminology

SPI Peripheral Clock and SPI Bus Clock

The SPI controller needs the peripheral clock to drive the SPI logic unit to perform the data transfer. The peripheral clock rate is determined by the settings of clock divisor (SPI_CLKDIV) and the clock source which can be HXT, HIRC, PLL out or the PCLK. SPInSEL (n=0, 1, 2) of CLK_CLKSEL2 register determines the clock source of the SPIn peripheral clock. The DIVIDER (SPI_CLKDIV[7:0]) setting determines the divisor of the clock rate calculation.

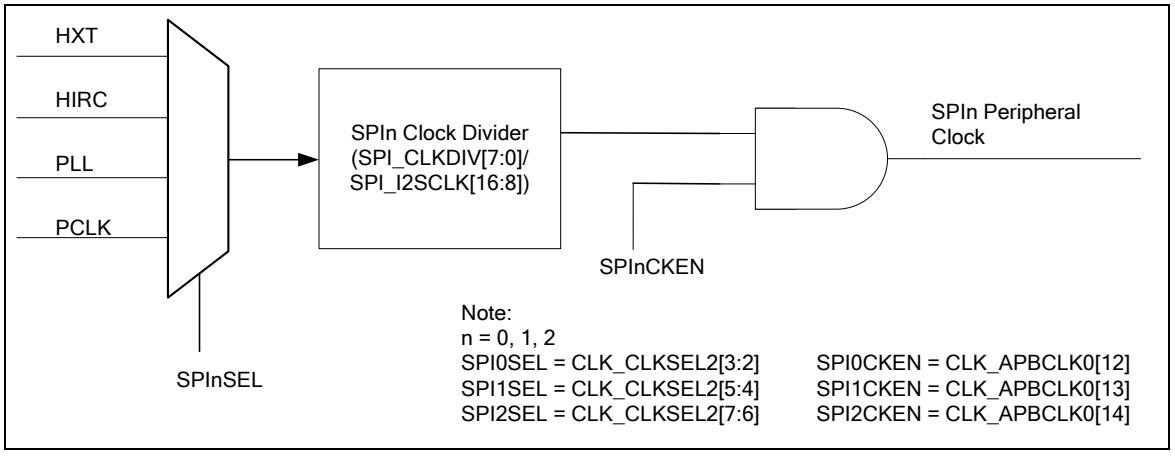


Figure 6.16-3 SPI Peripheral Clock

In Master mode, the frequency of the SPI bus clock is equal to the peripheral clock rate. In general, the SPI bus clock is denoted as SPI clock. In Slave mode, the SPI bus clock is provided by an off-chip master device. The frequency of SPI peripheral clock cannot be faster than the system clock rate regardless of Master or Slave mode. If the clock source of peripheral clock is different from the one of system clock, the frequency of SPI peripheral clock shall be slower than the system clock frequency regardless of Master or Slave mode.

In I^{2}S mode, the peripheral clock rate is equal to I^{2}S bit clock rate determined by SPI_I2SCLK

register.

Master/Slave mode

This SPI controller can be set as Master or Slave mode by setting the SLAVE (SPI_CTL[18]) to communicate with the off-chip SPI slave or master device. The application block diagrams in Master and Slave mode are shown below.

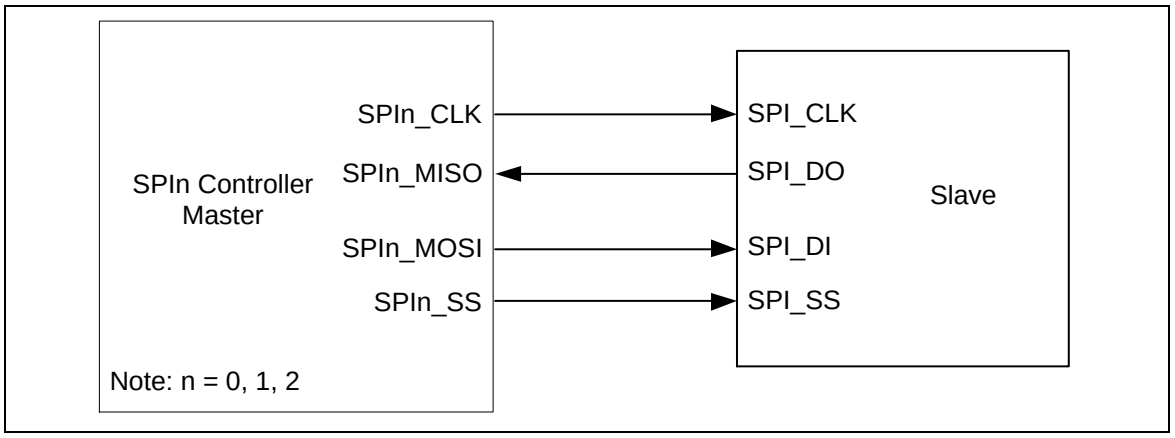


Figure 6.16-4 SPI Master Mode Application Block Diagram

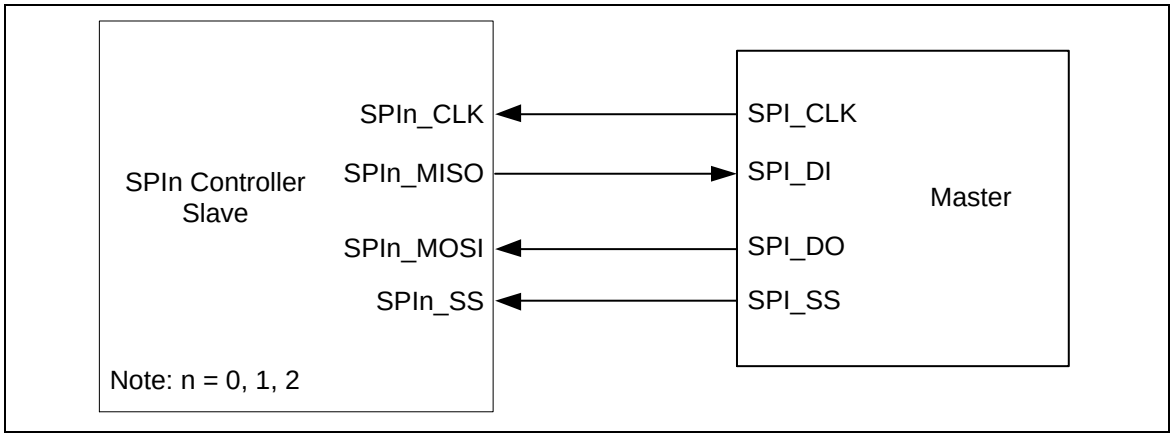


Figure 6.16-5 SPI Slave Mode Application Block Diagram

Slave Selection

In Master mode, the SPI controller can drive off-chip slave device through the slave select output pin SPIn_SS. In Slave mode, the off-chip master device drives the slave selection signal from the SPIn_SS input port to this SPI controller. The duration between the slave select active edge and the first SPI clock input shall over 3 SPI peripheral clock cycles of slave.

In Master/Slave mode, the active state of slave selection signal can be programmed to low or high active in SSACTPOL (SPI_SSCTL[2]). The selection of slave select conditions depends on what type of device is connected.

In Slave mode, to recognize the inactive state of the slave selection signal, the inactive period of the slave selection signal must be larger than or equal to 3 peripheral clock cycles between two successive transactions.

Timing Condition

The CLKPOL (SPI_CTL[3]) defines the SPI clock idle state. If CLKPOL = 1, the output of SPI clock is high at idle state; if CLKPOL = 0, it is low at idle state.

TXNEG (SPI_CTL[2]) defines the data transmitted out either on negative edge or on positive edge of SPI clock.

RXNEG (SPI_CTL[1]) defines the data received either on negative edge or on positive edge of SPI clock.

Note: The settings of TXNEG and RXNEG are mutual exclusive. In other words, do not transmit and receive data at the same clock edge.

Transmit/Receive Bit Length

The bit length of a transaction word is defined in DWIDTH (SPI_CTL[12:8]) and can be configured up to 32-bit length in a transaction word for transmitting and receiving.

When SPI controller finishes a transaction, i.e. receives or transmits a special count of bits defined in DWIDTH (SPI_CTL[12:8]), the unit transfer interrupt flag will be set to 1.

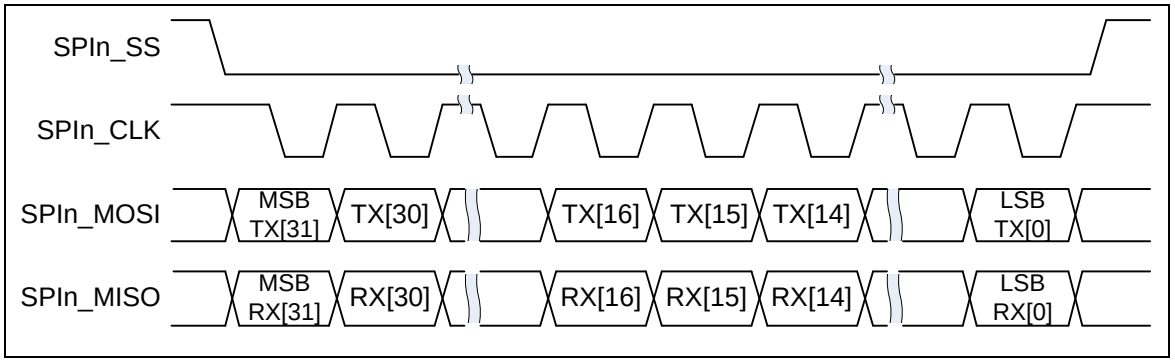


Figure 6.16-632-Bit in One Transaction (Master Mode)

LSB/MSB First

LSB (SPI_CTL[13]) defines the bit transfer sequence in a transaction. If the LSB (SPI_CTL[13]) is set to 1, the transfer sequence is LSB first. The bit 0 will be transferred firstly. If the LSB (SPI_CTL[13]) is cleared to 0, the transfer sequence is MSB first.

Suspend Interval

SUSPITV (SPI_CTL[7:4]) provide a configurable suspend interval, 0.5 ~ 15.5 SPI clock periods, between two successive transaction words in Master mode. The definition of the suspend interval is the interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value of SUSPITV is 0x3 (3.5 SPI clock cycles).

6.16.5.2 Automatic Slave Selection

In Master mode, if AUTOSS (SPI_SSCTL[3]) is set, the slave selection signal will be generated automatically and output to the SPIn_SS pin according to whether SS (SPI_SSCTL[0]) is enabled or not. The slave selection signal will be set to active state by the SPI controller when the SPI data transfer is started by writing to FIFO. It will be set to inactive state when SPI bus is idle. If SPI bus is not idle, i.e. TX FIFO, TX shift register or TX skew buffer is not empty, the slave

selection signal will be set to inactive state between transactions if the value of SUSPITV (SPI_CTL[7:4]) is greater than or equal to 3.

In Master mode, if the value of SUSPITV is less than 3 and the AUTOSS is set as 1, the slave selection signal will be kept at active state between two successive transactions.

If the AUTOSS bit is cleared, the slave selection output signal will be determined by the SS setting. The active state of the slave selection output signal is specified in SSACTPOL (SPI_SSCTL[2]).

The duration between the slave selection signal active edge and the first SPI bus clock edge is 1 SPI bus clock cycle and the duration between the last SPI bus clock and the slave selection signal inactive edge is 1.5 SPI bus clock cycle.

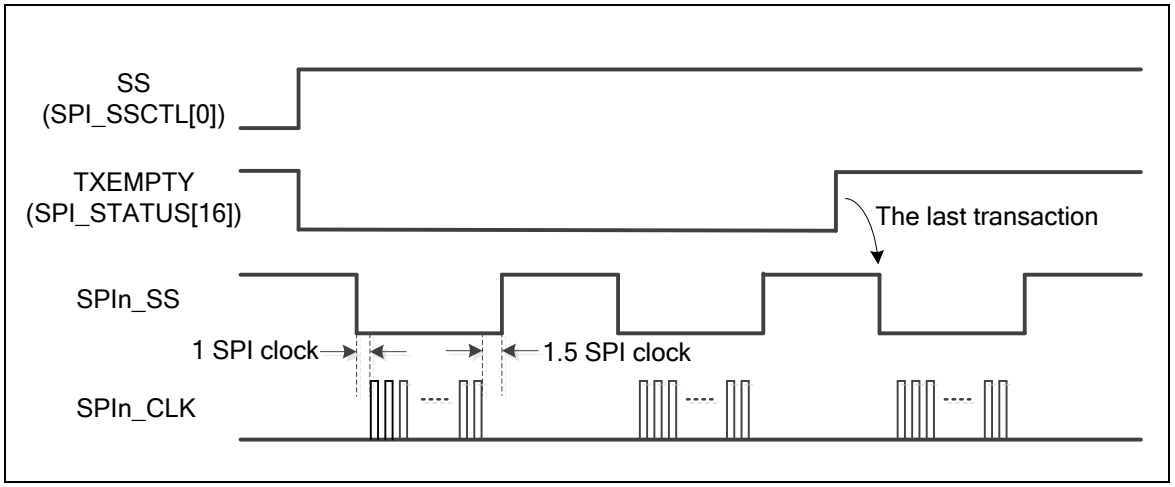


Figure 6.16-7 Automatic Slave Selection (SSACTPOL = 0, SPI_CYCLE > 0x2)

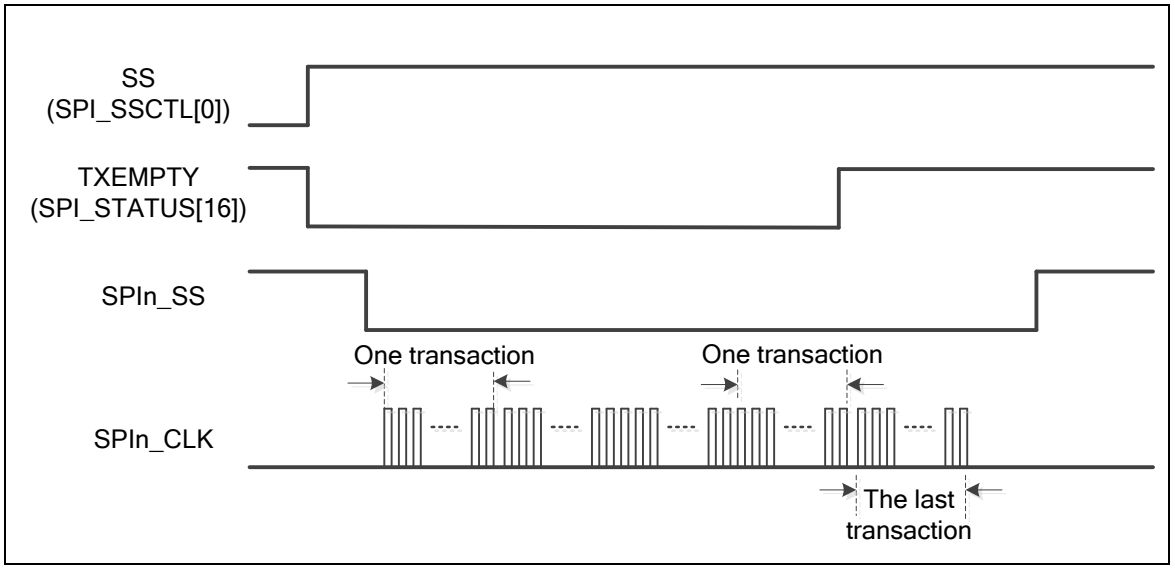


Figure 6.16-8 Automatic Selection (SSACTPOL = 0, SPI_CYCLE < 0x3)

6.16.5.3 Byte Reorder and Suspend Function

When the transfer is set as MSB first (LSB = 0) and the REORDER (SPI_CTL[19]) is set to 1, the data stored in the TX buffer and RX buffer will be rearranged in the order as [Byte0, Byte1, Byte2, Byte3] in 32-bit transfer (DWIDTH = 0). The sequence of transmitted/received data will be Byte0,

Byte1, Byte2, and then Byte3. If the DWIDTH is set as 24-bit transfer mode, the data in TX buffer and RX buffer will be rearranged as [unknown byte, Byte0, Byte1, Byte2]. The SPI controller will transmit/receive data with the sequence of Byte0, Byte1 and then Byte2. Each byte will be transmitted/received with MSB first. The rule of 16-bit mode is the same as above. Byte Reorder function is only available when DWIDTH is configured as 16, 24, and 32 bits.

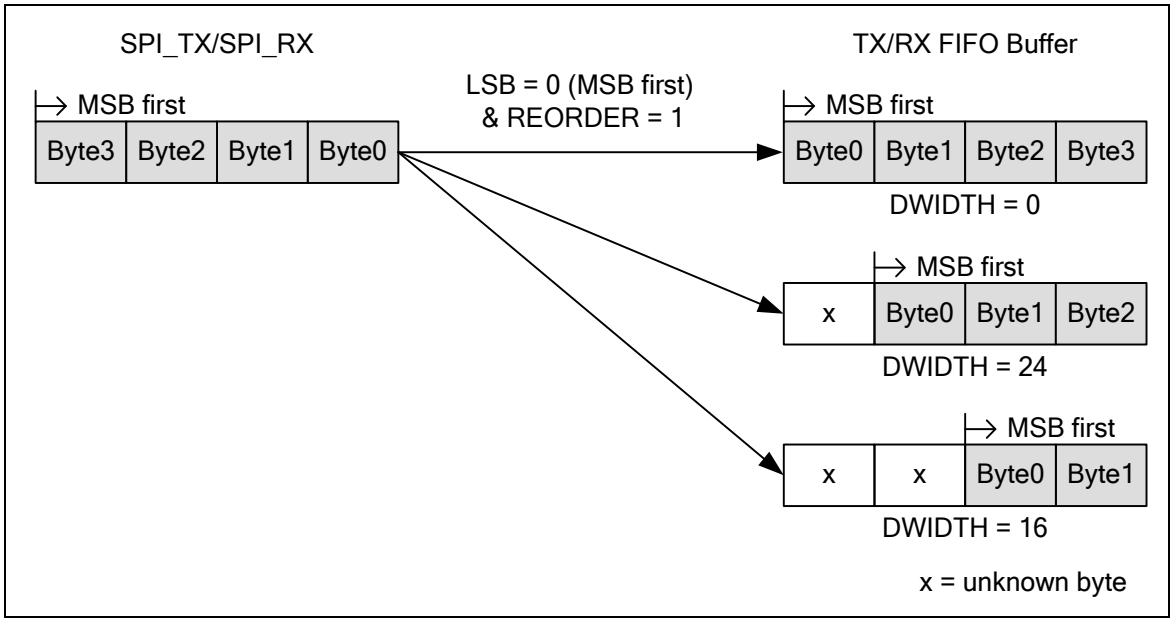


Figure 6.16-9 Byte Reorder Function

In Master mode, if REORDER (SPI_CTL[19]) is set to 1, a suspend interval of 0.5 ~ 15.5 SPI clock periods will be inserted by hardware between two successive bytes in a transaction word. The suspend interval is configured in SUSPITV (SPI_CTL[7:4]).

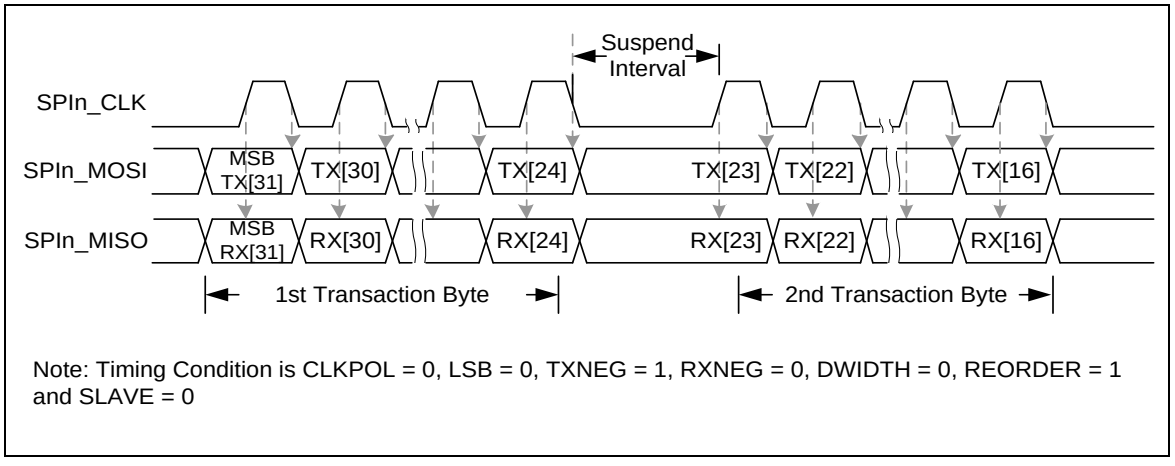


Figure 6.16-10 Timing Waveform for Byte Suspend

6.16.5.4 Slave 3-Wire Mode

When SLV3WIRE (SPI_SSCTL[4]) is set by software to enable the Slave 3-Wire mode, the SPI controller can work with no slave selection signal in Slave mode. The SLV3WIRE (SPI_SSCTL[4]) only takes effect in Slave mode. Only three pins, SPI0_CLK, SPI0_MISO, and SPI0_MOSI, are

required to communicate with a SPI master. The SPI0_SS pin can be configured as a GPIO. When the SLV3WIRE (SPI_SSCTL[4]) is set to 1, the SPI slave will be ready to transmit/receive data after the SPIEN (SPI_CTL[0]) is set to 1.

Note: This function is only supported in SPI0.

6.16.5.5 PDMA Transfer Function

SPI controller supports PDMA transfer function.

When TXPDMAEN (SPI_PDMACTL[0]) is set to 1, the controller will issue request to PDMA controller to start the PDMA transmission process automatically.

When RXPDMAEN (SPI_PDMACTL[1]) is set to 1, the controller will start the PDMA reception process. SPI controller will issue request to PDMA controller automatically when there is data in the RX FIFO buffer.

6.16.5.6 Two-Bit Transfer Mode

The SPI controller also supports 2-Bit Transfer mode when setting TWOBIT (SPI_CTL[16]) to 1. In 2-Bit Transfer mode, the SPI controller performs full duplex data transfer. In other words, the two serial data bits can be transmitted and received simultaneously.

For example, in Master mode, the even data (TX Data (n)) stored in the SPI_TX register will be transmitted through the SPI0_MOSI0 pin and the odd data (TX Data (n+1)) stored in the SPI_TX register will be transmitted through the SPI0_MOSI1 pin respectively. In the meanwhile, the even data received from SPI0_MISO0 pin will be written to RX FIFO prior to the odd data received from SPI0_MISO1 pin.

In Slave mode, the even and odd data stored in the SPI_TX register will be transmitted through the SPI0_MISO0 pin and SPI0_MISO1 pin respectively. In the meanwhile, the SPI0_RX register will store the even data received from the SPI0_MOSI0 pin and the odd data from SPI0_MOSI1 pin respectively. The data sequence of FIFO buffers is the same as the Master mode.

Note: This function is only supported in SPI0.

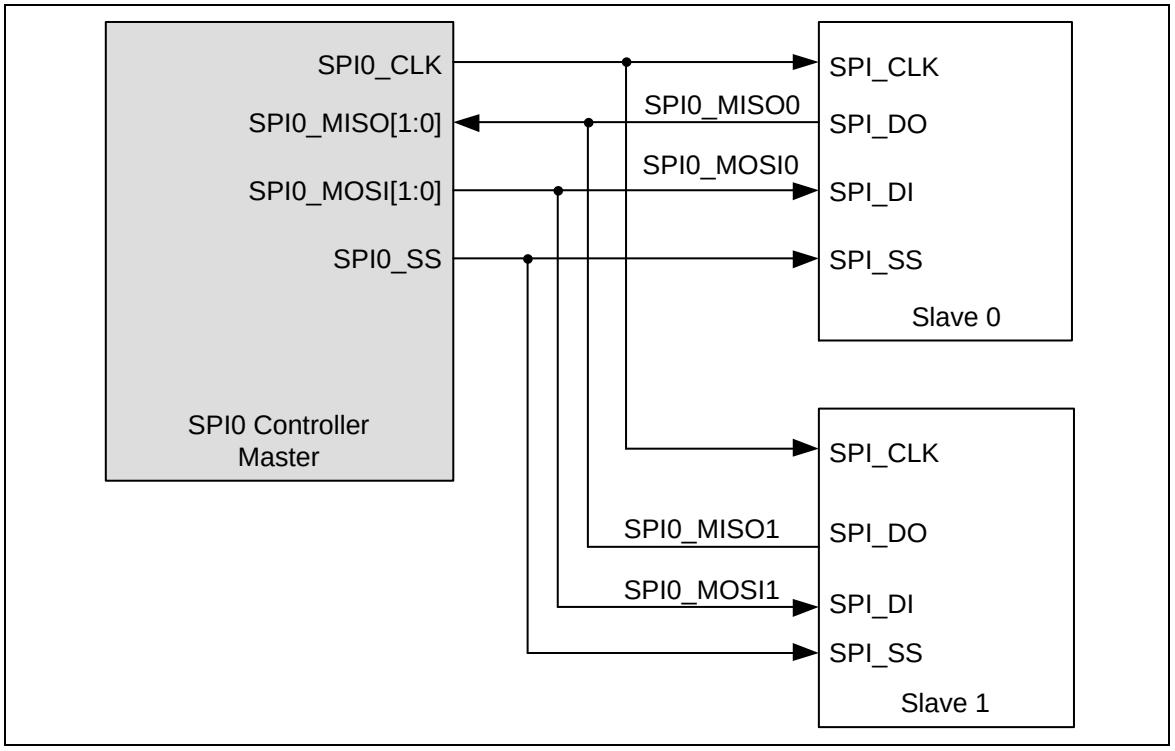


Figure 6.16-11 Two-Bit Transfer Mode System Architecture

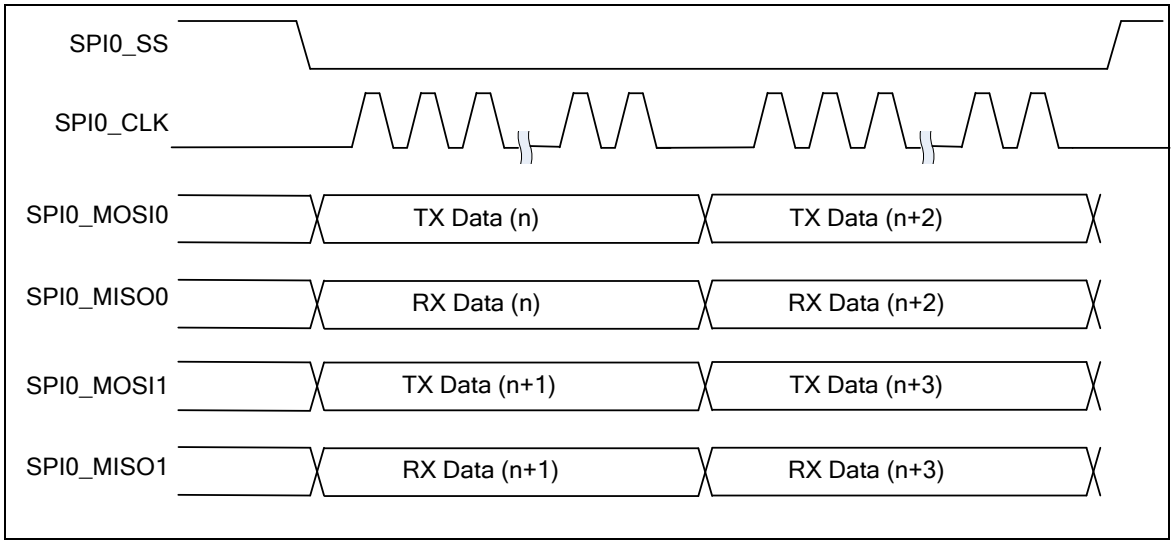


Figure 6.16-12 Two-Bit Transfer Mode Timing (Master Mode)

6.16.5.7 Dual I/O Mode

The SPI0 controller also supports Dual I/O transfer when setting the DUALIOEN ((SPI_CTL[21]) to 1. Many general SPI flashes support Dual I/O transfer. The QDIODIR (SPI_CTL[20]) is used to define the direction of the transfer data. When the QDIODIR bit is set to 1, the controller will send the data to external device. When the QDIODIR bit is set to 0, the controller will read the data from the external device. This function supports 8, 16, 24, and 32 bits of length.

The Dual I/O mode is not supported when the Slave 3-Wire mode or the Byte Reorder function is

enabled.

For Dual I/O mode, if both the DUALIOEN (SPI_CTL[21]) and QDIODIR (SPI_CTL[20]) are set as 1, the SPI0_MOSI0 is the even bit data output and the SPI0_MISO0 will be set as the odd bit data output. If the DUALIOEN (SPI_CTL[21]) is set as 1 and QDIODIR (SPI_CTL[20]) is set as 0, both the SPI0_MISO0 and SPI0_MOSI0 will be set as data input ports.

Note: This function is only supported in SPI0.

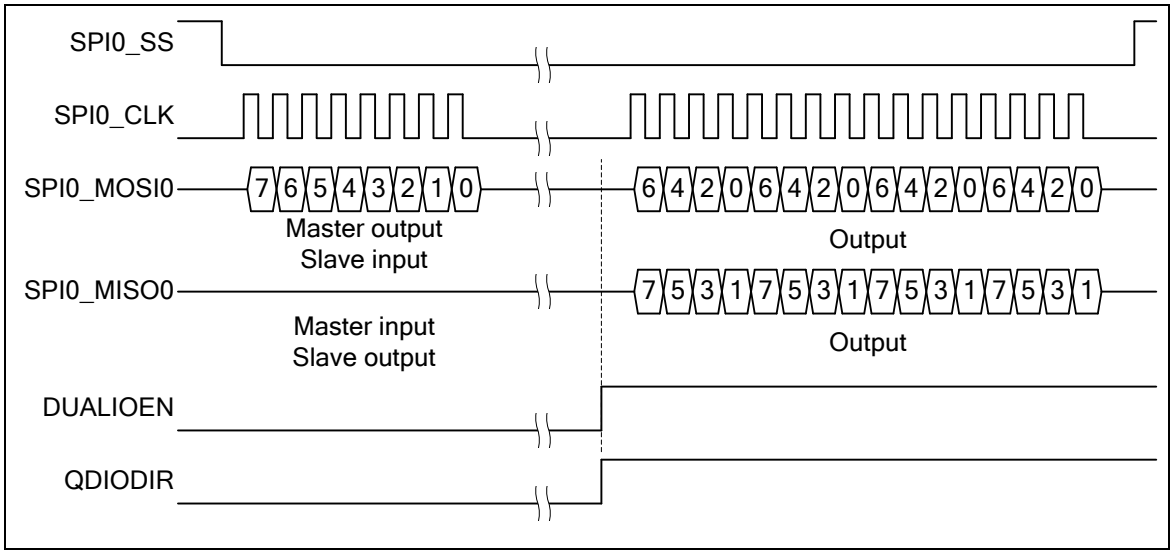


Figure 6.16-13 Bit Sequence of Dual Output Mode

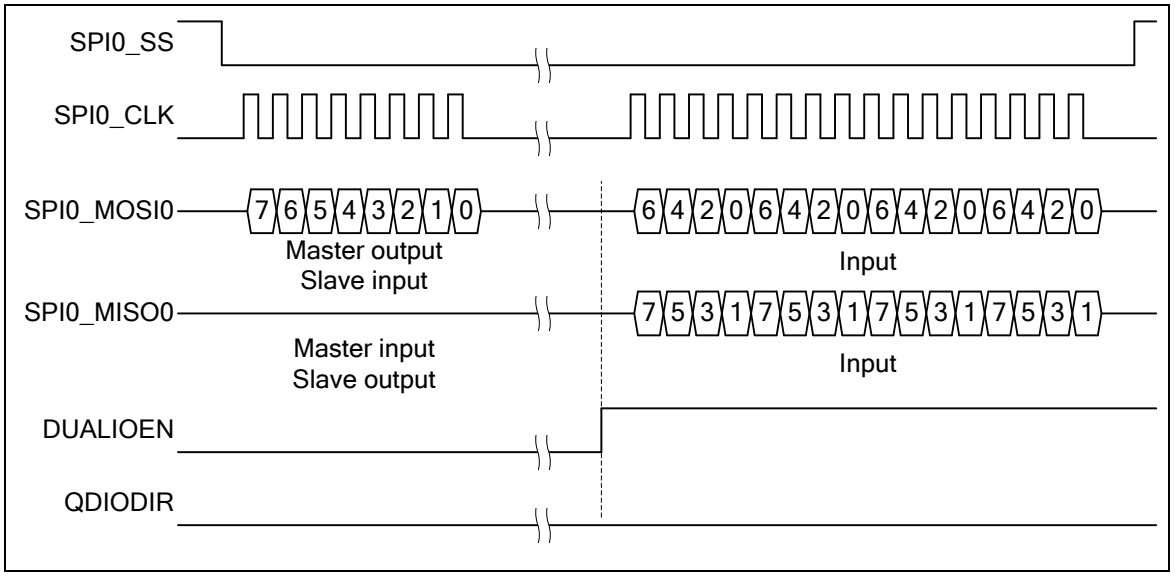


Figure 6.16-14 Bit Sequence of Dual Input Mode

6.16.5.8 Quad I/O Mode

The SPI0 controller also supports Quad I/O transfer when setting the QUADIOEN (SPI_CTL[22]) to 1. Many general SPI flashes support Quad I/O transfer. The QDIODIR bit (SPI_CTL[20]) is used to define the direction of the transfer data. When the QDIODIR (SPI_CTL[20]) is set to 1, the controller will send the data to external device. When the QDIODIR (SPI_CTL[20]) is set to 0, the

controller will read the data from the external device. This function supports 8, 16, 24, and 32 bits of length.

The Quad I/O mode is not supported when the Slave 3-Wire mode or the Byte Reorder function is enabled. The DUALIOEN (SPI_CTL[21]) and QUADIOEN (SPI_CTL[22]) shall not be set to 1 simultaneously.

For Quad I/O mode, if both the QUADIOEN (SPI_CTL[22]) and QDIODIR (SPI_CTL[20]) are set as 1, the SPI0_MOSI0 and SPI0_MOSI1 are the even bit data output and the SPI0_MISO0 and SPI0_MISO1 will be set as the odd bit data output. If the QUADIOEN (SPI_CTL[22]) is set as 1 and QDIODIR (SPI_CTL[20]) is set as 0, all the SPI0_MISO0, SPI0_MISO1, SPI0_MOSI0 and SPI0_MOSI1 pins will be set as data input ports.

Note: This function is only supported in SPI0.

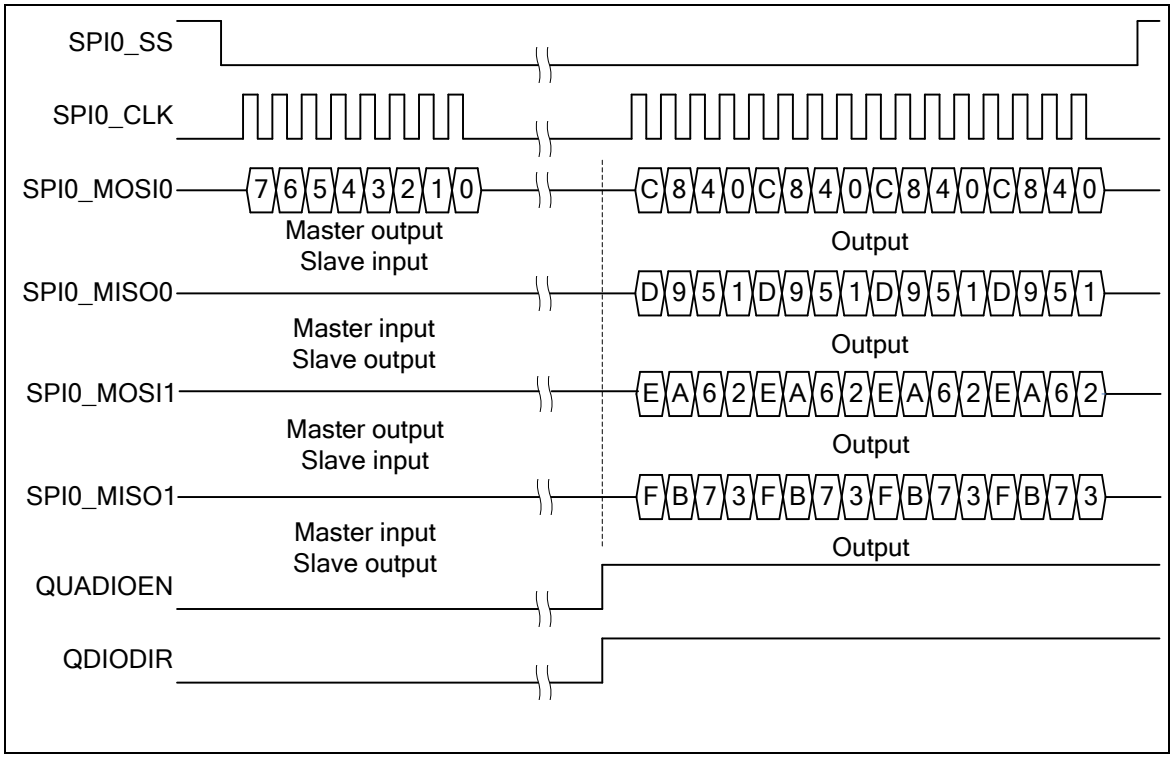


Figure 6.16-15 Bit Sequence of Quad Output Mode

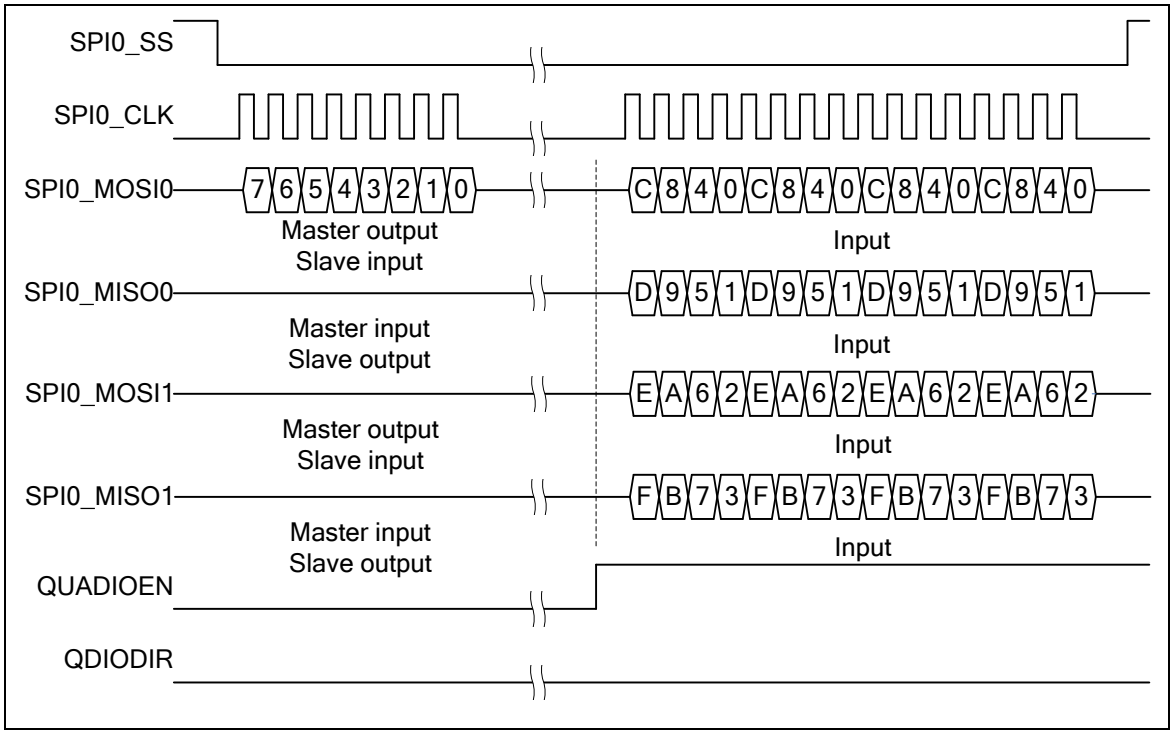


Figure 6.16-16 Bit Sequence of Quad Input Mode

6.16.5.9 FIFO Buffer Operation

The SPI controllers equip with 4/8 32-bit wide transmit and receive FIFO buffers.

The data stored in the transmit FIFO buffer will be read and sent out by the transmission control logic. If the transmit FIFO buffer is full, the TXFULL (SPI_STATUS[17]) will be set to 1. When the SPI transmission logic unit draws out the last datum of the transmit FIFO buffer, so that the transmit FIFO buffer is empty, the TXEMPTY (SPI_STATUS[16]) will be set to 1. Notice that the TXEMPTY (SPI_STATUS[16]) flag is set to 1 while the last transaction is still in progress. In Master mode, the BUSY (SPI_STATUS[0]) is set to 1 when the FIFO buffer is written any data or there is any transaction on the SPI bus. (e.g. the slave selection signal is active and the SPI controller is receiving data in slave mode). It will set to 0 when the transmit FIFO is empty and the current transaction has done. Thus, the status of BUSY (SPI_STATUS[0]) should be checked by software to make sure whether the SPI is in idle or not.

The receive control logic will store the SPI input data into the receive FIFO buffer. There are FIFO related status bits, like RXEMPTY (SPI_STATUS[8]) and RXFULL (SPI_STATUS[9]), to indicate the current status of RX FIFO buffer.

The transmitting and receiving threshold can be configured by setting TXTH (SPI_FIFOCTL[30:28]) and RXTH (SPI_FIFOCTL[26:24]). When the count of valid data stored in transmit FIFO buffer is less than or equal to TXTH (SPI_FIFOCTL[30:28]) setting, TXTHIF (SPI_STATUS[18]) will be set to 1. When the count of valid data stored in receive FIFO buffer is larger than RXTH (SPI_FIFOCTL[26:24]) setting, RXTHIF (SPI_STATUS[10]) will be set to 1.

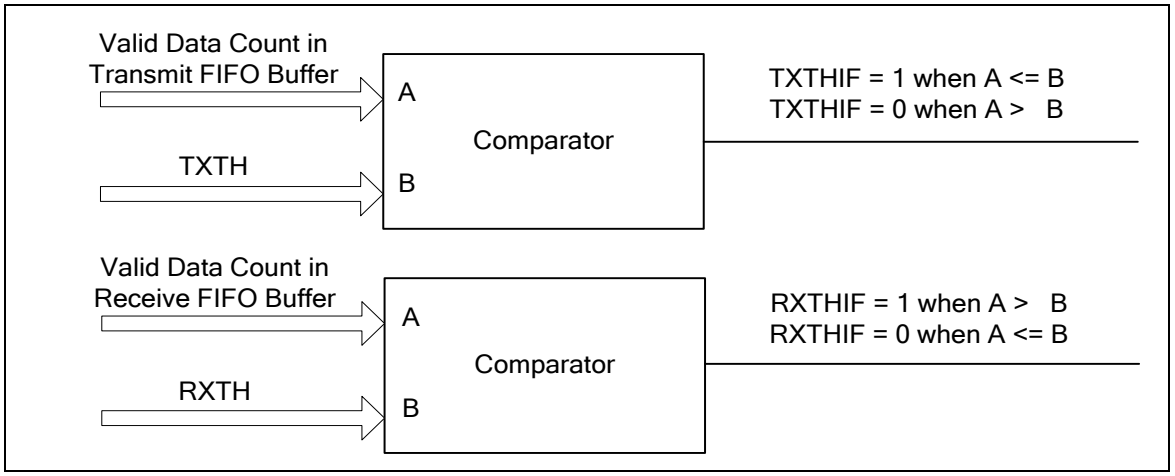


Figure 6.16-17 FIFO Threshold Comparator

In Master mode, the first datum is written to the SPI_TX register, the TXEMPTY flag (SPI_STATUS[16]) will be cleared to 0. The transmission will start after 1 APB clock cycle and 6 peripheral clock cycles. User can write the next data into SPI_TX register immediately. The SPI controller will insert a suspend interval between two successive transactions. The period of suspend interval is decided by the setting of SUSPITV (SPI_CTL[7:4]). If the SUSPITV (SPI_CTL[7:4]) equals 0, SPI controller can perform continuous transfer. User can write data into SPI_TX register as long as the TXFULL (SPI_STATUS[17]) is 0.

In the Example 1 of the Figure 6.16-18, it indicates the updated condition of TXEMPTY (SPI_STATUS[16]) and the relationship among the FIFO buffer, shift register and the skew buffer. The TXEMPTY (SPI_STATUS[16]) is set to 0 when the Data 0 is written into the FIFO buffer. The Data 0 will be loaded into the shift register by core logical and the TXEMPTY (SPI_STATUS[16]) will be to 1. The Data 0 in shift register will be shifted into skew buffer by bit for transmission until the transfer is done.

In the Example 2, it indicates the updated condition of TXFULL (SPI_STATUS[17]) when there are 8 data in the FIFO buffer and the next data of Data 9 does not be written into the FIFO buffer when the TXFULL = 1.

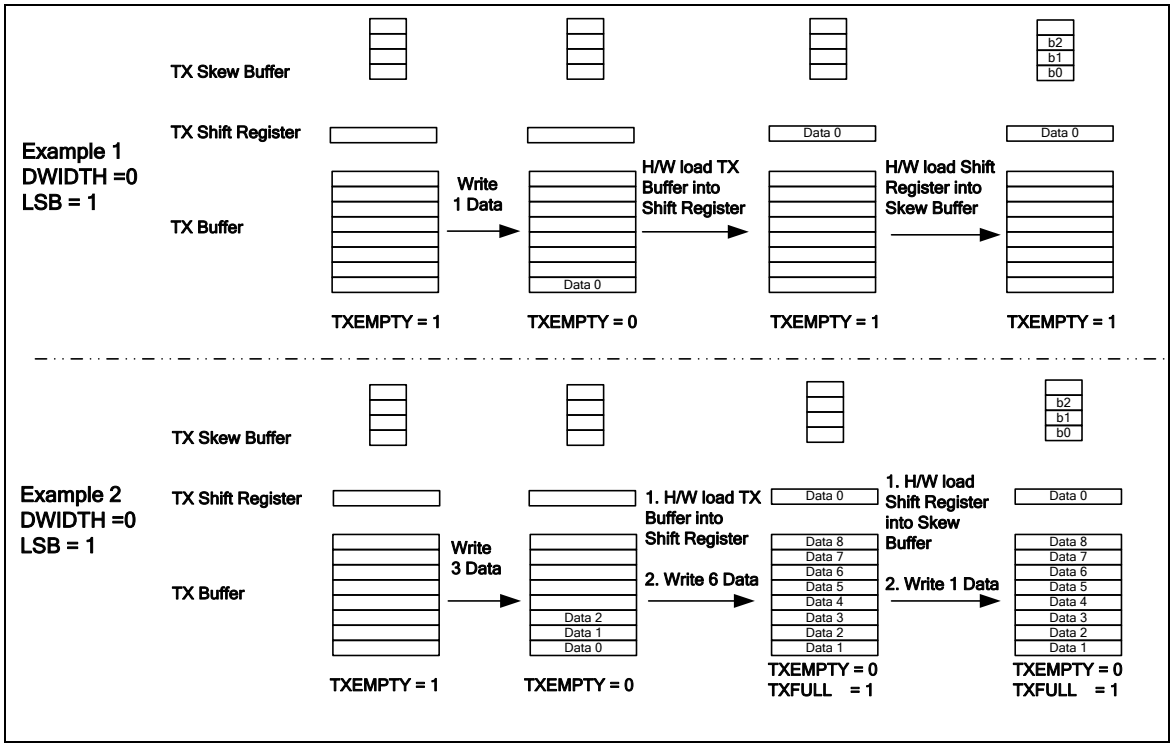


Figure 6.16-18 Transmit FIFO Buffer Example

The subsequent transactions will be triggered automatically if the transmitted data are updated in time. If the SPI_TX register does not be updated after all data transfer are done, the transfer will stop.

In Master mode, during receiving operation, the serial data are received from SPIn_MISO0/1 pin and stored to receive FIFO buffer.

The receive data (Data 0's b0, b1, …b31) is stored into skew buffer first according the serial clock (SPIn_CLK) and then is shifted into the shift register bit by bit. The core logic will load the data in shift register into FIFO buffer when the receive data bit reach the value of DWIDTH (SPI_CTL[12:8]). The RXEMPTY (SPI_STATUS[8]) will be cleared to 0 while the receive FIFO buffer contains unread data (see the Example 1 of Receive FIFO Buffer Example). The received data can be read by software from SPI_RX register as long as the RXEMPTY (SPI_STATUS[8]) is 0. If the receive FIFO buffer contains 8 unread data, the RXFULL (SPI_STATUS[9]) will be set to 1 (see the Example 2 of Receive FIFO Buffer Example).

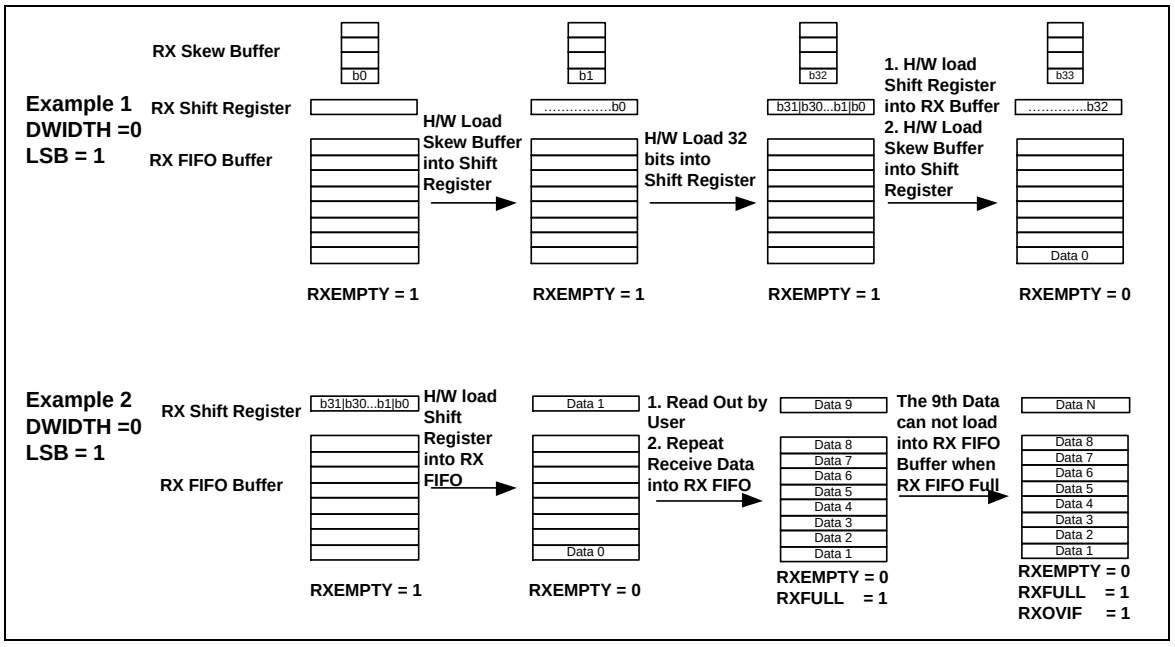


Figure 6.16-19 Receive FIFO Buffer Example

In Slave mode, during transmission operation, when data is written to the SPI_TX register by software, the data will be loaded into transmit FIFO buffer and the TXEMPTY (SPI_STATUS[16]) will be set to 0. The transmission will start when the slave device receives clock signal from master. Data can be written to SPI_TX register as long as the TXFULL (SPI_STATUS[17]) is 0. After all data have been drawn out by the SPI transmission logic unit and the SPI_TX register is not updated by software, the TXEMPTY (SPI_STATUS[16]) will be set to 1.

If there is no any data written to the SPI_TX register, the transmit underflow flag, TXUFIF (SPI_STATUS[19]) will be set to 1 when the slave selection signal is active. The output data will be held by TXUFPOL (SPI_FIFOCTL[6]) setting during this transfer until the slave selection signal goes to inactive state. When the transmit underflow event occurs, the slave under run flag, SLVURIF (SPI_STATUS[7]), will be set to 1 as SPIn_SS goes to inactive state.

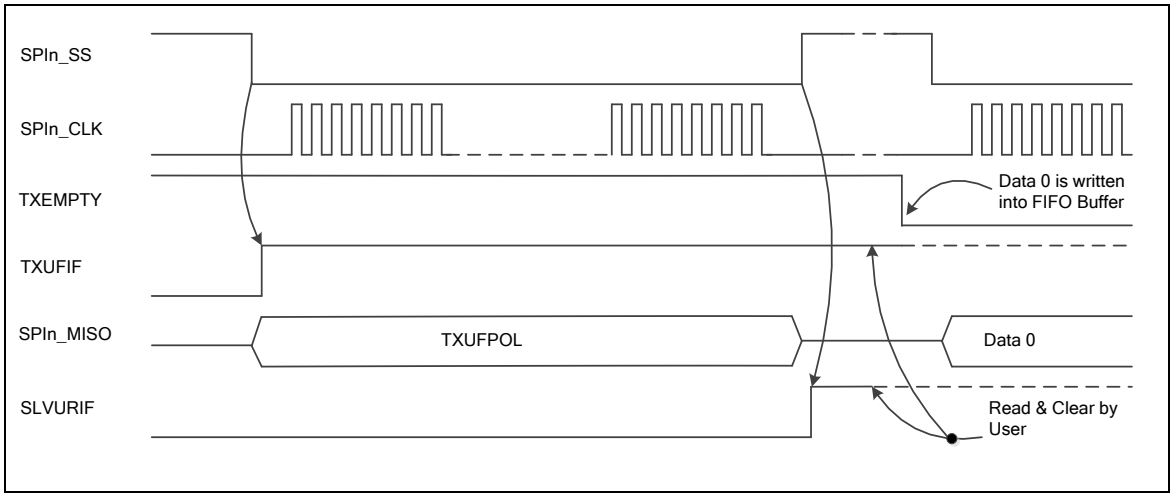


Figure 6.16-20 TX Underflow Event and Slave Under Run Event (Slave 3-Wire Mode Disabled)

In 2-Bit Transfer mode, the transmit data is loaded into shift register after 2 datum have been written into the TX FIFO buffer. It uses 2 shift registers and 2 4-level skew buffers concurrently. The detail timing of 2-Bit Transfer mode, please refer to the section of Two-Bit Transfer mode.

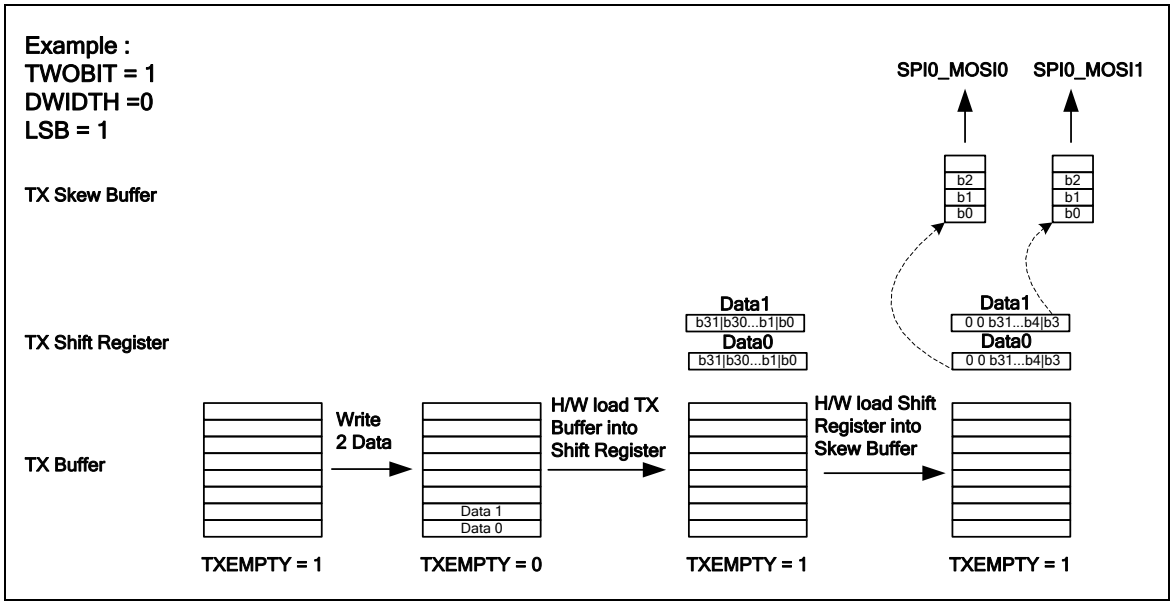


Figure 6.16-21 Two-Bit Transfer Mode FIFO Buffer Example

In Slave 3-Wire mode, the first 2-bit data is un-predicted (keep on the level of last bit in previously transfer) if the data is written into TX FIFO among 3 peripheral clock cycles before the SPI bus clock is presented. The other bits are held by TXUFPOL (SPI_FIFOCTL[6]) because there is TX underflow event. The written data will be transmitted in the next transfer.

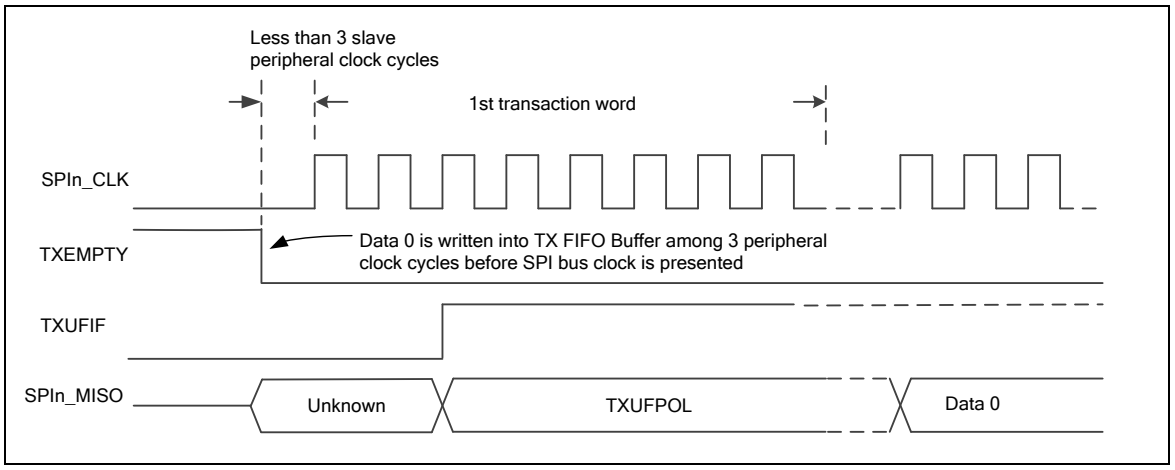


Figure 6.16-22 TX Underflow Event (Slave 3-Wire Mode Enabled)

In Slave mode, during receiving operation, the serial data is received from SPIn_MOSI0/1 pin and stored to SPI_RX register. The reception mechanism is similar to Master mode reception operation. If the receive FIFO buffer contains 8 unread data, the RXFULL (SPI_STATUS[9]) will be set to 1 and the RXOVIF (SPI_STATUS[11]) will be set 1 if there is more serial data is received from SPIn_MOSI and follow-up data will be dropped (refer to the Receive FIFO Buffer Example figure). If the receive bit count mismatch with the DWIDTH (SPI_CTL[12:8]) when the slave selection line goes to inactive state, the SLVBEIF (SPI_STATUS[6]) will be set to 1.

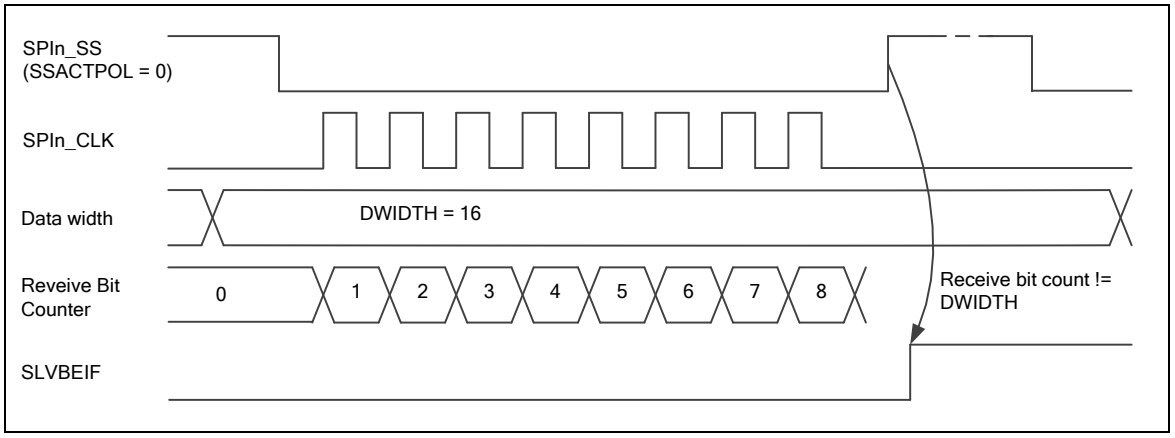


Figure 6.16-23 Slave Mode Bit Count Error

When the Slave selection signal is active and the value of SLVTOCNT (SPI_SSCTL[31:16]) is not 0, the Slave time-out counter in the SPI controller logic will start after the serial clock input. This counter will be cleared after one transaction done or the SLVTOCNT is set to 0. If the value of the time-out counter is greater than or equal to the value of SLVTOCNT before one transaction done, the slave time-out event occurs and the SLVTOIF (SPI_STATUS[5]) will be set to 1.

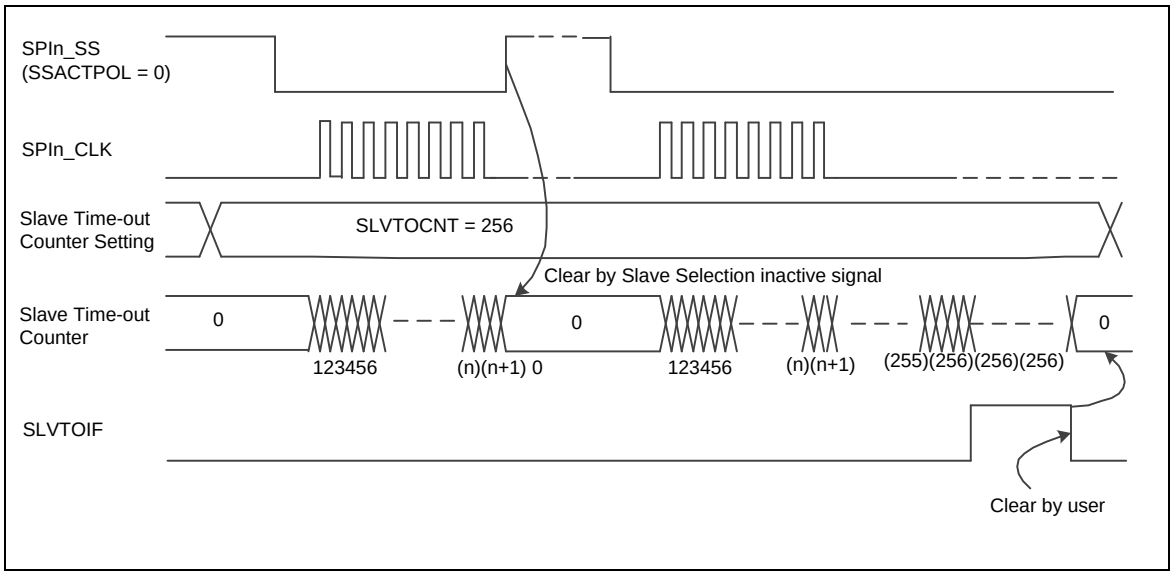


Figure 6.16-24 Slave Time-out Event

A receive time-out function is built-in in this controller. When the receive FIFO is not empty and no read operation in receive FIFO over 64 SPI clock period in Master mode or over 576 SPI peripheral clock period in Slave mode, the receive time-out occurs and the RXTOIF (SPI_STATUS[12]) be set to 1. When the receive FIFO is read by user, the time-out status will be cleared automatically.

Note: 8-level FIFO is only supported in SPI0, and 4-level FIFO is supported in SPI1 and SPI2.

6.16.5.10 Interrupt

- SPI unit transfer interrupt

As the SPI controller finishes a unit transfer, the unit transfer interrupt flag UNITIF (SPI_STATUS[1]) will be set to 1. The unit transfer interrupt event will generate an interrupt to CPU if the unit transfer interrupt enable bit UNITIEN (SPI_CTL[17]) is set. The unit transfer interrupt flag can be cleared only by writing 1 to it.

- SPI slave selection active/inactive interrupt

In Slave mode, the slave selection active/inactive interrupt flag, SSACTIF (SPI_STATUS[2]) and SSINAIF (SPI_STATUS[3]), will be set to 1 when the SPIEN (SPI_CTL[0]) and SLAVE (SPI_CTL[18]) are set to 1 and the slave selection signal goes to active/inactive state. The SPI controller will issue an interrupt if the SSINAIEN (SPI_SSCTL[13]) or SSACTIEN (SPI_SSCTL[12]) is set to 1.

- Slave time-out interrupt

In Slave mode, there is slave time-out function for user to know that there is serial clock input but one transaction is not finished over the period of SLVTOCNT (SPI_SSCTL[31:16]) basing on Slave peripheral clock.

When the slave selection signal is active and the value of SLVTOCNT (SPI_SSCTL[31:16]) is not 0, the slave time-out counter in the SPI controller logic will start after the serial clock input. This counter will be cleared after one transaction done or the SLVTOCNT (SPI_SSCTL[31:16]) is set to 0. If the value of the time-out counter is greater than or equal to the value of SLVTOCNT (SPI_SSCTL[31:16]) before one transaction done, the slave time-out event occurs and the SLVTOIF (SPI_STATUS[5]) will be set to 1. The SPI controller will issue an interrupt if the SLVTOIEN (SPI_SSCTL[5]) is set to 1.

- Slave bit count error interrupt

In Slave mode, if the transmit/receive bit count mismatch with the DWIDTH (SPI_CTL[12:8]) when the slave selection line goes to inactive state, the SLVBEIF (SPI_STATUS[6]) will be set to 1. The uncompleted transaction will be dropped from TX and RX shift registers. The SPI controller will issue an interrupt if the SLVBEIEN (SPI_SSCTL[8]) is set to 1.

Note: If the slave selection signal is active but there is no any serial clock input, the SLVBEIF (SPI_STATUS[6]) will be set to 1 when the slave selection signal goes to inactive state.

- TX underflow interrupt

In SPI Slave mode, if there is no any data is written to the SPI_TX register, the TXUFIF (SPI_STATUS[19]) will be set to 1 when the slave selection signal is active. The SPI controller will issue a TX underflow interrupt if the TXUFIEN (SPI_FIFOCTL[7]) is set to 1.

- Slave TX under run interrupt

If the TX underflow event occurs, the SLVURIF (SPI_STATUS[7]) will be set to 1 when SPIn_SS goes to inactive state. The SPI controller will issue a TX under run interrupt if the SLVURIEN (SPI_SSCTL[9]) is set to 1.

Note: In Slave 3-Wire mode, the slave selection signal is considered active all the time so that user shall poll the TXUFIF (SPI_STATUS[19]) to know if there is TX underflow event or not.

- Receive Overrun interrupt

In Slave mode, if the receive FIFO buffer contains 8 unread data, the RXFULL (SPI_STATUS[9]) flag will be set to 1 in SPI0 (4 unread data in SPI1/2) and the RXOVIF (SPI_STATUS[11]) will be set 1 if there is more serial data is received from SPI bus and follow-up data will be dropped. The SPI controller will issue an interrupt if the RXOVIEN (SPI_FIFOCTL[5]) is set to 1.

- Receive FIFO time-out interrupt

If there is a received data in the FIFO buffer and it is not read by software over 64 SPI peripheral clock periods in Master mode or over 576 SPI peripheral clock periods in Slave mode, it will send

a RX time-out interrupt to the system if the RX time-out interrupt enable bit, RXTOIEN (SPI_FIFOCTL[4]), is set to 1.

- Transmit FIFO interrupt

In FIFO mode, if the valid data count of the transmit FIFO buffer is less than or equal to the setting value of TXTH (SPI_FIFOCTL[30:28]), the transmit FIFO interrupt flag TXTHIF (SPI_STATUS[18]) will be set to 1. The SPI controller will generate a transmit FIFO interrupt to the system if the transmit FIFO interrupt enable bit, TXTHIEN (SPI_FIFOCTL[3]), is set to 1.

- Receive FIFO interrupt

In FIFO mode, if the valid data count of the receive FIFO buffer is larger than the setting value of RXTH (SPI_FIFOCTL[26:24]), the receive FIFO interrupt flag RXTHIF (SPI_STATUS[10]) will be set to 1. The SPI controller will generate a receive FIFO interrupt to the system if the receive FIFO interrupt enable bit, RXTHIEN (SPI_FIFOCTL[2]), is set to 1.

6.16.5.11 I²S Mode

The SPI1 and SPI2 controllers support I²S mode with PCM mode A, PCM mode B, MSB justified and I²S data format. The I2S_LRCLK signal indicates which audio channel is in transferring. The bit count of an audio channel is determined by WDWIDTH (SPI_I2SCTL[5:4]). The transfer sequence is always first from the most significance bit, MSB. Data are read on rising clock edge and are driven on falling clock edge of I2S_BCLK.

In I²S data format, the MSB is sent and latched on the second clock of an audio channel.

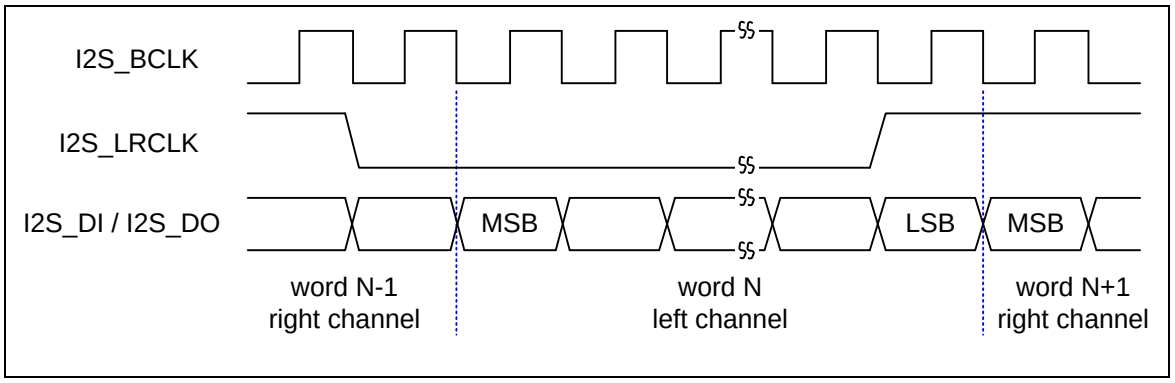


Figure 6.16-25 I²S Data Format Timing Diagram

In MSB justified data format, the MSB is sent and latched on the first clock of an audio channel.

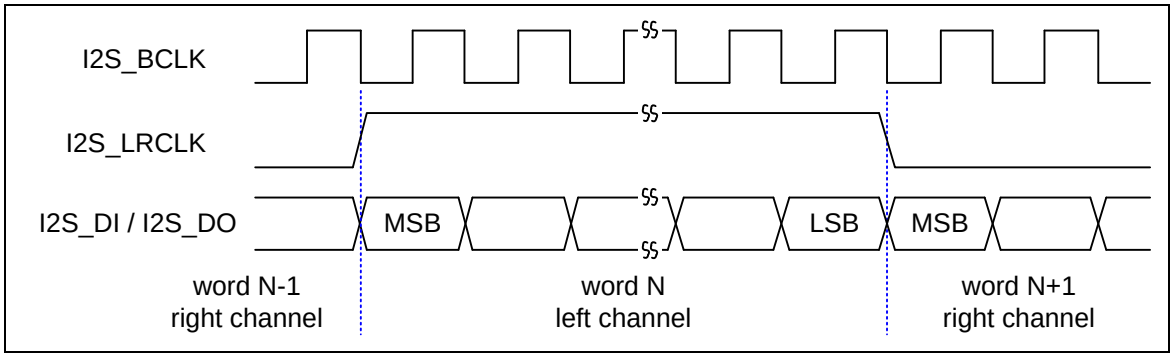


Figure 6.16-26 MSB Justified Data Format Timing Diagram

The I2S_LRCLK signal also supports PCM mode A and PCM mode B.

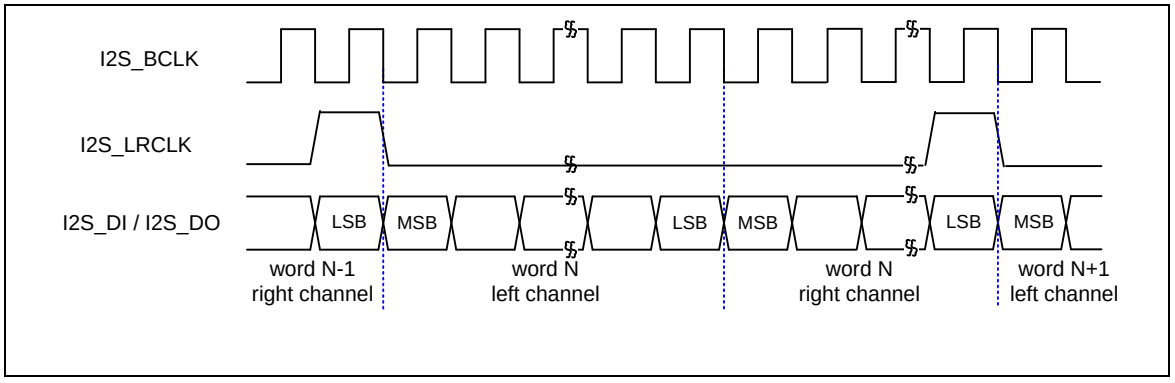


Figure 6.16-27 PCM Mode A Timing Diagram

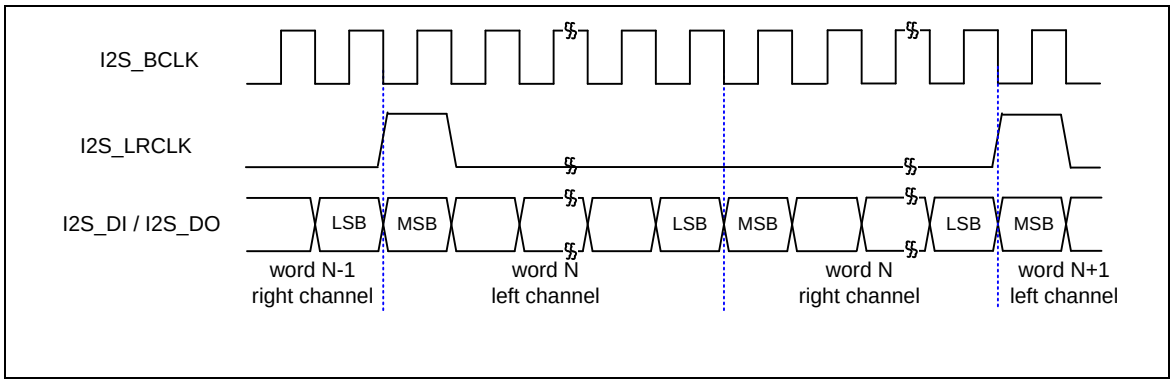


Figure 6.16-28 PCM Mode B Timing Diagram

6.16.5.12 I²S Mode FIFO operation

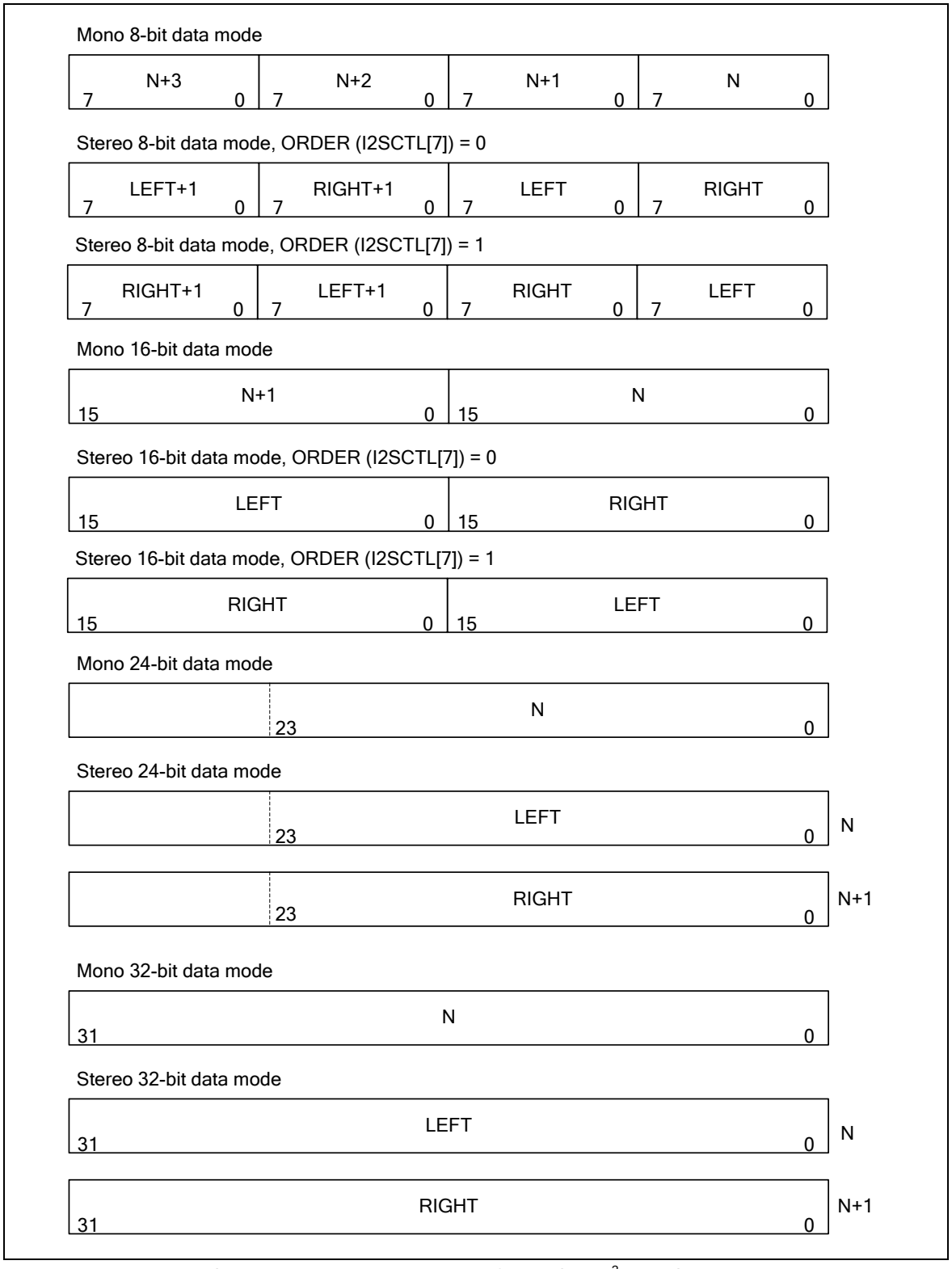


Figure 6.16-29 FIFO Contents for Various I²S Modes

6.16.6 Timing Diagram

The active state of slave selection signal can be defined by setting the SSACTPOL (SPI_SSCTL[2]). The SPI clock which is in idle state can be configured as high or low state by setting the CLKPOL (SPI_CTL[3]). It also provides the bit length of a transaction word in DWIDTH (SPI_CTL[12:8]), and transmitting/receiving data from MSB or LSB first in LSB bit (SPI_CTL[13]). User can also select which edge of SPI clock to transmit/receive data in TXNEG/RXNEG (SPI_CTL[2:1]). Four SPI timing diagrams for master/slave operations and the related settings are shown below.

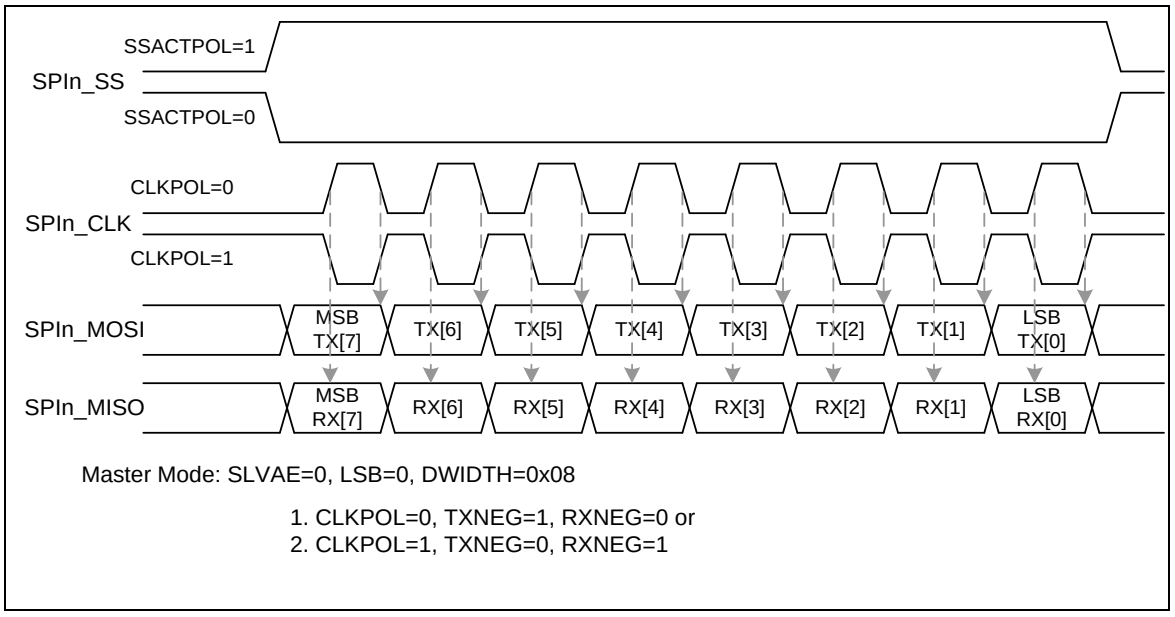


Figure 6.16-30 SPI Timing in Master Mode

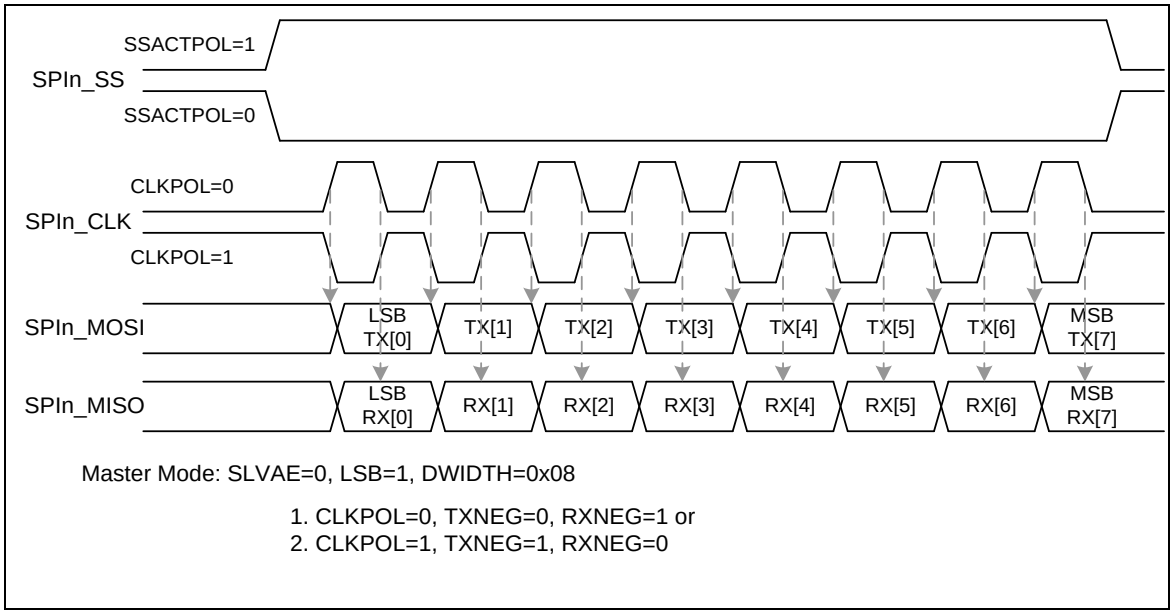


Figure 6.16-31 SPI Timing in Master Mode (Alternate Phase of SPIn_CLK)

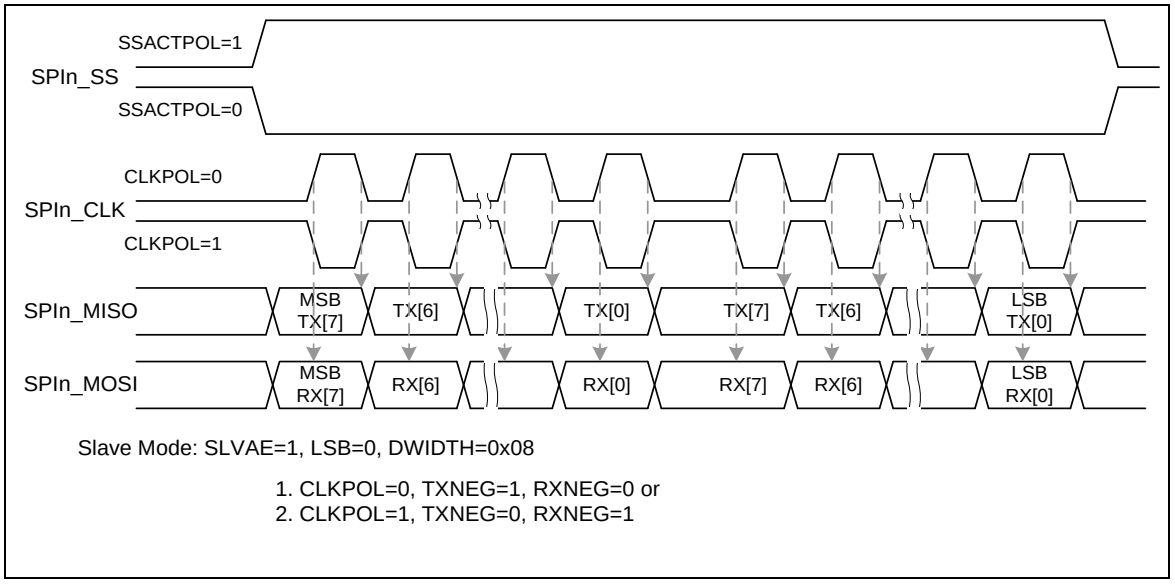


Figure 6.16-32 SPI Timing in Slave Mode

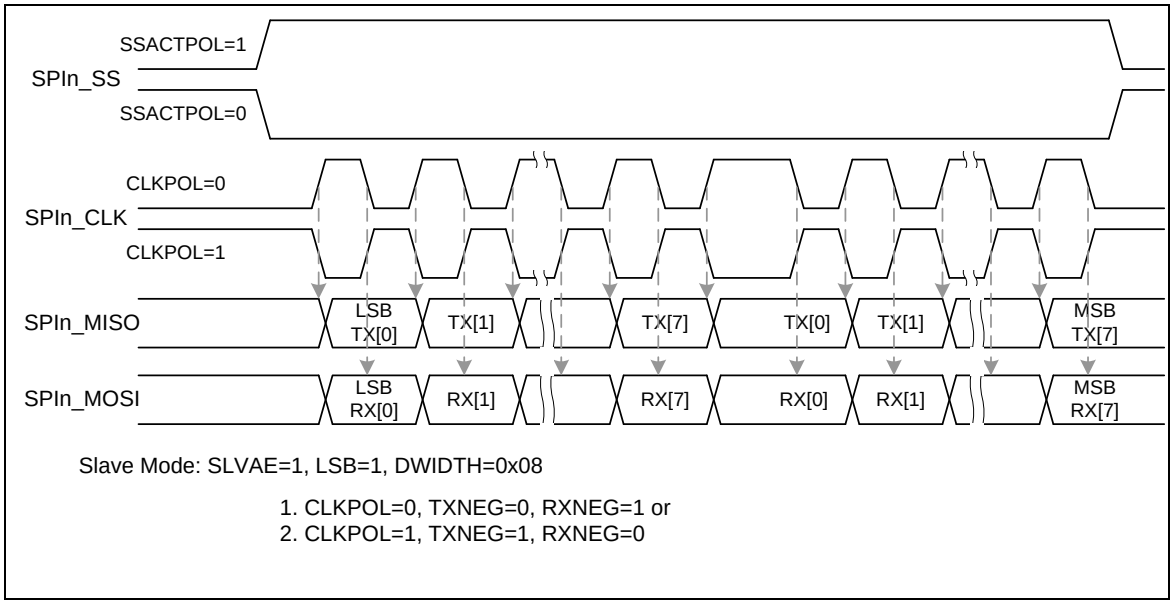


Figure 6.16-33 SPI Timing in Slave Mode (Alternate Phase of SPIn_CLK)

6.16.7 Programming Examples

Example 1: The SPI controller is set as a master to access an off-chip slave device with the following specifications:

- Data bit is latched on positive edge of SPI bus clock.
- Data bit is driven on negative edge of SPI bus clock.
- Data is transferred from MSB first.
- SPI bus clock is low at idle state.
- Only one byte of data to be transmitted/received in a transaction.
- Uses the SPI slave select pin to connect with an off-chip slave device. The slave selection signal is active low.

The operation flow is as follows:

1) Set DIVIDER (SPI_CLKDIV[7:0]) to determine the output frequency of SPI clock.
2) Write the SPI_SSCTL register a proper value for the related settings of Master mode:
    1. Clear AUTOSS (SPI_SSCTL[3]) to 0 to disable the Automatic Slave Selection function.
    2. Configure slave selection signal as active low by clearing SSACTPOL (SPI_SSCTL[2]) to 0.
    3. Enable slave selection signal by setting SS (SPI_SSCTL[0]) to 1 to active the off-chip slave device.
3) Write the related settings into the SPI_CTL register to control the SPI master actions.
    1. Configure this SPI controller as master device by setting SLAVE (SPI_CTL[18]) to 0.
    2. Force the SPI clock to low at ilde state by clearing CLKPOL (SPI_CTL[3]) to 0.
    3. Select data transmitted on negative edge of SPI bus clock by setting TXNEG (SPI_CTL[2]) to 1.
    4. Select data latched on positive edge of SPI bus clock by clearing RXNEG (SPI_CTL[1]) to 0.
    5. Set the bit length of a transaction as 8-bit in DWIDTH bit field (SPI_CTL[12:8] = 0x08).
    6. Set MSB transfer first by clearing MSB (SPI_CTL[13]) to 0.
4) Set SPIEN (SPI_CTL[0]) to 1 to enable the data transfer with the SPI interface.
5) If this SPI master attempts to transmit (write) one byte data to the off-chip slave device, write the byte data that will be transmitted into the SPI_TX register.
6) Waiting for SPI interrupt if the UNITIEN (SPI_CTL[17]) is set to 1, or just polling the unit transfer interrupt flag UNITIF (SPI_STATUS[1]).
7) Read out the received one byte data from SPI_RX register.
8) Go to 5) to continue another data transfer or set SS (SPI_SSCTL[0]) to 0 to inactivate the off-chip slave device.

Example 2: The SPI controller is set as a slave device and connects with an off-chip master device. The off-chip master device communicates with the on-chip SPI slave controller through the SPI interface with the following specifications:

- Data bit is latched on positive edge of SPI bus clock.
- Data bit is driven on negative edge of SPI bus clock.
- Data is transferred from LSB first.
- SPI bus clock is high at idle state.
- Only one byte of data to be transmitted/received in a transaction.
- Slave selection signal is active high.

The operation flow is as follows:

2) Write the SPI_SSCTL register a proper value for the related settings of Slave mode.

   Select high level for the input of slave selection signal by setting SSACTPOL (SPI_SSCTL[2]) to 1.

3) Write the related settings into the SPI_CTL register to control this SPI slave actions

   1. Set the SPI controller as slave device by setting SLAVE (SPI_CTL[18]) to 1.
   2. Set the SPI clock to high at idle state by setting CLKPOL (SPI_CTL[3]) to 1.
   3. Select data transmitted on negative edge of SPI bus clock by setting TXNEG (SPI_CTL[2]) to 1.
   4. Select data latched on positive edge of SPI bus clock by clearing RXNEG (SPI_CTL[1]) to 0.
   5. Set the bit length of a transaction as 8-bit in DWIDTH bit field (SPI_CTL[12:8] = 0x08).
   6. Set LSB transfer first by setting LSB (SPI_CTL[13]) to 1.

4) Set the SPIEN (SPI_CTL[0]) to 1. Wait for the slave select trigger input and SPI clock input from the off-chip master device to start the data transfer.

5) If this SPI slave attempts to transmit (be read) one byte data to the off-chip master device, write the byte data that will be transmitted into the SPI_TX register.

6) If this SPI slave just only attempts to receive (be written) one byte data from the off-chip master device and does not care what data will be transmitted, the SPI_TX register does not need to be updated by software.

7) Waiting for SPI interrupt if the UNITIEN (SPI_CTL[17]) is set to 1, or just polling the unit transfer interrupt flag UNITIF (SPI_STATUS[1]).

8) Read out the received one byte data from SPI_RX register.

9) Go to 5) to continue another data transfer or stop data transfer.

6.16.8 Register Map

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

Register	Offset	R/W	Description	Reset Value
SPI Base Address: **SPIn_BA = 0x4006_0000 + (0x1000 * n)** **n= 0,1,2**				
SPI_CTL	SPIn_BA+0x00	R/W	SPI Control Register	0x0000_0034
SPI_CLKDIV	SPIn_BA+0x04	R/W	SPI Clock Divider Register	0x0000_0000
SPI_SSCTL	SPIn_BA+0x08	R/W	SPI Slave Select Control Register	0x0000_0000
SPI_PDMACTL	SPIn_BA+0x0C	R/W	SPI PDMA Control Register	0x0000_0000
SPI_FIFOCTL	SPIn_BA+0x10	R/W	SPI FIFO Control Register	0x4400_0000
SPI_STATUS	SPIn_BA+0x14	R/W	SPI Status Register	0x0005_0110
SPI_TX	SPIn_BA+0x20	W	Data Transmit Register	0x0000_0000
SPI_RX	SPIn_BA+0x30	R	Data Receive Register	0x0000_0000
SPI_I2SCTL	SPIn_BA+0x60	R/W	I^2S Control Register	0x0000_0000
SPI_I2SCLK	SPIn_BA+0x64	R/W	I^2S Clock Divider Control Register	0x0000_0000
SPI_I2SSTS	SPIn_BA+0x68	R/W	I^2S Status Register	0x0005_0100

Note: SPIn_BA indicates each set of SPI, and there are SPI0_BA, SPI1_BA, and SPI2_BA.

6.16.9 Register Description

SPI Control Register (SPI_CTL)

Register	Offset	R/W	Description	Reset Value
SPI_CTL	SPIn_BA+0x00	R/W	SPI Control Register	0x0000_0034

Note: Not supported in I^2S mode.

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved	QUADIOEN	DUALIOEN	QDIODIR	REORDER	SLAVE	UNITIEN	TWOBIT
15	14	13	12	11	10	9	8
Reserved		LSB	DWIDTH				
7	6	5	4	3	2	1	0
SUSPITV				CLKPOL	TXNEG	RXNEG	SPIEN

Bits	Description	
[31:23]	**Reserved**	Reserved.
[22]	**QUADIOEN**	**Quad I/O Mode Enable Bit (Only Supported in SPI0)** 0 = Quad I/O mode Disabled. 1 = Quad I/O mode Enabled.
[21]	**DUALIOEN**	**Dual I/O Mode Enable Bit (Only Supported in SPI0)** 0 = Dual I/O mode Disabled. 1 = Dual I/O mode Enabled.
[20]	**QDIODIR**	**Quad or Dual I/O Mode Direction Control (Only Supported in SPI0)** 0 = Quad or Dual Input mode. 1 = Quad or Dual Output mode.
[19]	**REORDER**	**Byte Reorder Function Enable Bit** 0 = Byte Reorder function Disabled. 1 = Byte Reorder function Enabled. A byte suspend interval will be inserted among each byte. The period of the byte suspend interval depends on the setting of SUSPITV. **Note:** 1. Byte Reorder function is only available if DWIDTH is defined as 16, 24, and 32 bits. 2. Byte Reorder function is not supported when the Quad or Dual I/O mode is enabled.
[18]	**SLAVE**	**Slave Mode Control** 0 = Master mode. 1 = Slave mode.
[17]	**UNITIEN**	**Unit Transfer Interrupt Enable Bit** 0 = SPI unit transfer interrupt Disabled. 1 = SPI unit transfer interrupt Enabled.

[16]	**TWOBIT**	**2-bit Transfer Mode Enable Bit (Only Supported in SPI0)** 0 = 2-Bit Transfer mode Disabled. 1 = 2-Bit Transfer mode Enabled. **Note:** When 2-Bit Transfer mode is enabled, the first serial transmitted bit data is from the first FIFO buffer data, and the 2nd serial transmitted bit data is from the second FIFO buffer data. As the same as transmitted function, the first received bit data is stored into the first FIFO buffer and the 2nd received bit data is stored into the second FIFO buffer at the same time.
[15:14]	**Reserved**	Reserved.
[13]	**LSB**	**Send LSB First** 0 = The MSB, which bit of transmit/receive register depends on the setting of DWIDTH, is transmitted/received first. 1 = The LSB, bit 0 of the SPI TX register, is sent first to the SPI data output pin, and the first bit received from the SPI data input pin will be put in the LSB position of the RX register (bit 0 of SPI_RX).
[12:8]	**DWIDTH**	**Data Width** This field specifies how many bits can be transmitted/received in one transaction. The minimum bit length is 8 bits and can up to 32 bits. DWIDTH = 0x08 … 8 bits. DWIDTH = 0x09 … 9 bits. …… DWIDTH = 0x1F … 31 bits. DWIDTH = 0x00 … 32 bits.
[7:4]	**SUSPITV**	**Suspend Interval (Master Only)** The four bits provide configurable suspend interval between two successive transmit/receive transaction in a transfer. The definition of the suspend interval is the interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value is 0x3. The period of the suspend interval is obtained according to the following equation. (SUSPITV[3:0] + 0.5) * period of SPICLK clock cycle Example: SUSPITV = 0x0 … 0.5 SPICLK clock cycle. SUSPITV = 0x1 … 1.5 SPICLK clock cycle. …… SUSPITV = 0xE … 14.5 SPICLK clock cycle. SUSPITV = 0xF … 15.5 SPICLK clock cycle.
[3]	**CLKPOL**	**Clock Polarity** 0 = SPI bus clock is idle low. 1 = SPI bus clock is idle high.
[2]	**TXNEG**	**Transmit on Negative Edge** 0 = Transmitted data output signal is changed on the rising edge of SPI bus clock. 1 = Transmitted data output signal is changed on the falling edge of SPI bus clock.
[1]	**RXNEG**	**Receive on Negative Edge** 0 = Received data input signal is latched on the rising edge of SPI bus clock. 1 = Received data input signal is latched on the falling edge of SPI bus clock.
[0]	**SPIEN**	**SPI Transfer Control Enable Bit** In Master mode, the transfer will start when there is a data in the FIFO buffer after this is set to 1. In Slave mode, this device is ready to receive data when this bit is set to 1. 0 = Transfer control Disabled. 1 = Transfer control Enabled.

		Note: Before changing the configurations of SPI_CTL, SPI_CLKDIV, SPI_SSCTL and SPI_FIFOCTL registers, user shall clear the SPIEN (SPI_CTL[0]) and confirm the SPIENSTS (SPI_STATUS[15]) is 0.

SPI Clock Divider Register (SPI_CLKDIV)

Register	Offset	R/W	Description	Reset Value
SPI_CLKDIV	SPIn_BA+0x04	R/W	SPI Clock Divider Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
DIVIDER							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**DIVIDER**	**Clock Divider** The value in this field is the frequency divider for generating the peripheral clock, f_{spi_eclk}, and the SPI bus clock of SPI master. The frequency is obtained according to the following equation. $f_{spi\_eclk} = \frac{f_{spi\_clock\_src}}{(DIVIDER+1)}$ where $f_{spi\_clock\_src}$ is the peripheral clock source, which is defined in the clock control register, CLK_CLKSEL2. **Note:** Not supported in I^2S mode.

SPI Slave Select Control Register (SPI_SSCTL)

Register	Offset	R/W	Description	Reset Value
SPI_SSCTL	SPIn_BA+0x08	R/W	SPI Slave Select Control Register	0x0000_0000

Note: Not supported in I^2S mode.

31	30	29	28	27	26	25	24
SLVTOCNT							
23	22	21	20	19	18	17	16
SLVTOCNT							
15	14	13	12	11	10	9	8
Reserved		SSINAIEN	SSACTIEN	Reserved		SLVURIEN	SLVBEIEN
7	6	5	4	3	2	1	0
Reserved	SLVTORST	SLVTOIEN	SLV3WIRE	AUTOSS	SSACTPOL	Reserved	SS

Bits	Description	
[31:16]	**SLVTOCNT**	**Slave Mode Time-out Period (Only Supported in SPI0)** In Slave mode, these bits indicate the time-out period when there is bus clock input during slave select active. The clock source of the time-out counter is Slave peripheral clock. If the value is 0, it indicates the slave mode time-out function is disabled.
[15:14]	**Reserved**	Reserved.
[13]	**SSINAIEN**	**Slave Select Inactive Interrupt Enable Bit** 0 = Slave select inactive interrupt Disabled. 1 = Slave select inactive interrupt Enabled.
[12]	**SSACTIEN**	**Slave Select Active Interrupt Enable Bit** 0 = Slave select active interrupt Disabled. 1 = Slave select active interrupt Enabled.
[11:10]	**Reserved**	Reserved.
[9]	**SLVURIEN**	**Slave Mode TX Under Run Interrupt Enable Bit** 0 = Slave mode TX under run interrupt Disabled. 1 = Slave mode TX under run interrupt Enabled.
[8]	**SLVBEIEN**	**Slave Mode Bit Count Error Interrupt Enable Bit** 0 = Slave mode bit count error interrupt Disabled. 1 = Slave mode bit count error interrupt Enabled.
[7]	**Reserved**	Reserved.
[6]	**SLVTORST**	**Slave Mode Time-out Reset Control (Only Supported in SPI0)** 0 = When Slave mode time-out event occurs, the TX and RX control circuit will not be reset. 1 = When Slave mode time-out event occurs, the TX and RX control circuit will be reset by hardware.
[5]	**SLVTOIEN**	**Slave Mode Time-out Interrupt Enable Bit (Only Supported in SPI0)** 0 = Slave mode time-out interrupt Disabled. 1 = Slave mode time-out interrupt Enabled.

[4]	**SLV3WIRE**	**Slave 3-wire Mode Enable Bit (Only Supported in SPI0)** Slave 3-wire mode is only available in SPI0. In Slave 3-wire mode, the SPI controller can work with 3-wire interface including SPI0_CLK, SPI0_MISO and SPI0_MOSI pins. 0 = 4-wire bi-direction interface. 1 = 3-wire bi-direction interface.
[3]	**AUTOSS**	**Automatic Slave Selection Function Enable Bit (Master Only)** 0 = Automatic slave selection function Disabled. Slave selection signal will be asserted/de-asserted according to SS (SPI_SSCTL[0]). 1 = Automatic slave selection function Enabled.
[2]	**SSACTPOL**	**Slave Selection Active Polarity** This bit defines the active polarity of slave selection signal (SPIn_SS). 0 = The slave selection signal SPIn_SS is active low. 1 = The slave selection signal SPIn_SS is active high.
[1]	**Reserved**	Reserved.
[0]	**SS**	**Slave Selection Control (Master Only)** If AUTOSS bit is cleared to 0, 0 = set the SPIn_SS line to inactive state. 1 = set the SPIn_SS line to active state. If the AUTOSS bit is set to 1, 0 = Keep the SPIn_SS line at inactive state. 1 = SPIn_SS line will be automatically driven to active state for the duration of data transfer, and will be driven to inactive state for the rest of the time. The active state of SPIn_SS is specified in SSACTPOL (SPI_SSCTL[2]).

SPI PDMA Control Register (SPI_PDMACTL)

Register	Offset	R/W	Description	Reset Value
SPI_PDMACTL	SPIn_BA+0x0C	R/W	SPI PDMA Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					PDMARST	RXPDMAEN	TXPDMAEN

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**PDMARST**	**PDMA Reset** 0 = No effect. 1 = Reset the PDMA control logic of the SPI controller. This bit will be automatically cleared to 0.
[1]	**RXPDMAEN**	**Receive PDMA Enable Bit** 0 = Receive PDMA function Disabled. 1 = Receive PDMA function Enabled.
[0]	**TXPDMAEN**	**Transmit PDMA Enable Bit** 0 = Transmit PDMA function Disabled. 1 = Transmit PDMA function Enabled. **Note:** In SPI master mode with full duplex transfer, if both TX and RX PDMA functions are enabled, RX PDMA function cannot be enabled prior to TX PDMA function. User can enable TX PDMA function firstly or enable both functions simultaneously.

SPI FIFO Control Register (SPI_FIFOCTL)

Register	Offset	R/W	Description	Reset Value
SPI_FIFOCTL	SPIn_BA+0x10	R/W	SPI FIFO Control Register	0x4400_0000

31	30	29	28	27	26	25	24
Reserved	TXTH			Reserved	RXTH		
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						TXFBCLR	RXFBCLR
7	6	5	4	3	2	1	0
TXUFIEN	TXUFPOL	RXOVIEN	RXTOIEN	TXTHIEN	RXTHIEN	TXRST	RXRST

Bits	Description	
[31]	**Reserved**	Reserved.
[30:28]	**TXTH**	**Transmit FIFO Threshold** If the valid data count of the transmit FIFO buffer is less than or equal to the TXTH setting, the TXTHIF bit will be set to 1, else the TXTHIF bit will be cleared to 0. In SPI0, TXTH is a 3-bit wide configuration; in SPI1 and SPI2, 2-bit wide only (SPI_FIFOCTL[29:28]).
[27]	**Reserved**	Reserved.
[26:24]	**RXTH**	**Receive FIFO Threshold** If the valid data count of the receive FIFO buffer is larger than the RXTH setting, the RXTHIF bit will be set to 1, else the RXTHIF bit will be cleared to 0. In SPI0, RXTH is a 3-bit wide configuration; in SPI1 and SPI2, 2-bit wide only (SPI_FIFOCTL[25:24]).
[23:10]	**Reserved**	Reserved.
[9]	**TXFBCLR**	**Transmit FIFO Buffer Clear** 0 = No effect. 1 = Clear transmit FIFO pointer. The TXFULL bit will be cleared to 0 and the TXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 1 system clock after it is set to 1. **Note:** The TX shift register will not be cleared.
[8]	**RXFBCLR**	**Receive FIFO Buffer Clear** 0 = No effect. 1 = Clear receive FIFO pointer. The RXFULL bit will be cleared to 0 and the RXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 1 system clock after it is set to 1. **Note:** The RX shift register will not be cleared.
[7]	**TXUFIEN**	**TX Underflow Interrupt Enable Bit** In Slave mode, when TX underflow event occurs, this interrupt flag will be set to 1. 0 = Slave TX underflow interrupt Disabled. 1 = Slave TX underflow interrupt Enabled.

[6]	**TXUFPOL**	**TX Underflow Data Polarity** 0 = The SPI data out is keep 0 if there is TX underflow event in Slave mode. 1 = The SPI data out is keep 1 if there is TX underflow event in Slave mode. **Note 1:** The TX underflow event occurs if there is not any data in TX FIFO when the slave selection signal is active. **Note 2:** This bit should be set as 0 in I^2S mode.
[5]	**RXOVIEN**	**Receive FIFO Overrun Interrupt Enable Bit** 0 = Receive FIFO overrun interrupt Disabled. 1 = Receive FIFO overrun interrupt Enabled.
[4]	**RXTOIEN**	**Slave Receive Time-out Interrupt Enable Bit** 0 = Receive time-out interrupt Disabled. 1 = Receive time-out interrupt Enabled.
[3]	**TXTHIEN**	**Transmit FIFO Threshold Interrupt Enable Bit** 0 = TX FIFO threshold interrupt Disabled. 1 = TX FIFO threshold interrupt Enabled.
[2]	**RXTHIEN**	**Receive FIFO Threshold Interrupt Enable Bit** 0 = RX FIFO threshold interrupt Disabled. 1 = RX FIFO threshold interrupt Enabled.
[1]	**TXRST**	**Transmit Reset** 0 = No effect. 1 = Reset transmit FIFO pointer and transmit circuit. The TXFULL bit will be cleared to 0 and the TXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 3 system clock cycles + 2 peripheral clock cycles after it is set to 1. User can read TXRXRST (SPI_STATUS[23]) to check if reset is accomplished or not. **Note:** If there is slave receive time-out event, the TXRST will be set to 1 when the SLVTORST (SPI_SSCTL[6]) is enabled.
[0]	**RXRST**	**Receive Reset** 0 = No effect. 1 = Reset receive FIFO pointer and receive circuit. The RXFULL bit will be cleared to 0 and the RXEMPTY bit will be set to 1. This bit will be cleared to 0 by hardware about 3 system clock cycles + 2 peripheral clock cycles after it is set to 1. User can read TXRXRST (SPI_STATUS[23]) to check if reset is accomplished or not. **Note:** If there is slave receive time-out event, the RXRST will be set 1 when the SLVTORST (SPI_SSCTL[6]) is enabled.

SPI Status Register (SPI_STATUS)

Register	Offset	R/W	Description	Reset Value
SPI_STATUS	SPIn_BA+0x14	R/W	SPI Status Register	0x0005_0110

Note: Not supported in I^2S mode.

31	30	29	28	27	26	25	24
TXCNT				RXCNT			
23	22	21	20	19	18	17	16
TXRXRST	Reserved			TXUFIF	TXTHIF	TXFULL	TXEMPTY
15	14	13	12	11	10	9	8
SPIENSTS	Reserved		RXTOIF	RXOVIF	RXTHIF	RXFULL	RXEMPTY
7	6	5	4	3	2	1	0
SLVURIF	SLVBEIF	SLVTOIF	SSLINE	SSINAIF	SSACTIF	UNITIF	BUSY

Bits	Description	
[31:28]	**TXCNT**	**Transmit FIFO Data Count (Read Only)** This bit field indicates the valid data count of transmit FIFO buffer.
[27:24]	**RXCNT**	**Receive FIFO Data Count (Read Only)** This bit field indicates the valid data count of receive FIFO buffer.
[23]	**TXRXRST**	**TX or RX Reset Status (Read Only)** 0 = The reset function of TXRST or RXRST is done. 1 = Doing the reset function of TXRST or RXRST. **Note:** Both the reset operations of TXRST and RXRST need 3 system clock cycles + 2 peripheral clock cycles. User can check the status of this bit to monitor the reset function is doing or done.
[22:20]	**Reserved**	Reserved.
[19]	**TXUFIF**	**TX Underflow Interrupt Flag** When the TX underflow event occurs, this bit will be set to 1, the state of data output pin depends on the setting of TXUFPOL. 0 = No effect. 1 = No data in Transmit FIFO and TX shift register when the slave selection signal is active. **Note 1:** This bit will be cleared by writing 1 to it. **Note 2:** If reset slave's transmission circuit when slave selection signal is active, this flag will be set to 1 after 3 system clock cycles + 2 peripheral clock cycles since the reset operation is done.
[18]	**TXTHIF**	**Transmit FIFO Threshold Interrupt Flag (Read Only)** 0 = The valid data count within the transmit FIFO buffer is larger than the setting value of TXTH. 1 = The valid data count within the transmit FIFO buffer is less than or equal to the setting value of TXTH.
[17]	**TXFULL**	**Transmit FIFO Buffer Full Indicator (Read Only)** 0 = Transmit FIFO buffer is not full. 1 = Transmit FIFO buffer is full.

[16]	**TXEMPTY**	**Transmit FIFO Buffer Empty Indicator (Read Only)** 0 = Transmit FIFO buffer is not empty. 1 = Transmit FIFO buffer is empty.
[15]	**SPIENSTS**	**SPI Enable Status (Read Only)** 0 = The SPI controller is disabled. 1 = The SPI controller is enabled. **Note:** The SPI peripheral clock is asynchronous with the system clock. In order to make sure the SPI control logic is disabled, this bit indicates the real status of SPI controller.
[14:13]	**Reserved**	Reserved.
[12]	**RXTOIF**	**Receive Time-out Interrupt Flag** 0 = No receive FIFO time-out event. 1 = Receive FIFO buffer is not empty and no read operation on receive FIFO buffer over 64 SPI clock period in Master mode or over 576 peripheral clock period in Slave mode. When the received FIFO buffer is read by software, the time-out status will be cleared automatically. **Note:** This bit will be cleared by writing 1 to it.
[11]	**RXOVIF**	**Receive FIFO Overrun Interrupt Flag** When the receive FIFO buffer is full, the follow-up data will be dropped and this bit will be set to 1. 0 = Receive FIFO does not over run. 1 = Receive FIFO over run. **Note:** This bit will be cleared by writing 1 to it.
[10]	**RXTHIF**	**Receive FIFO Threshold Interrupt Flag (Read Only)** 0 = The valid data count within the RX FIFO buffer is smaller than or equal to the setting value of RXTH. 1 = The valid data count within the receive FIFO buffer is larger than the setting value of RXTH.
[9]	**RXFULL**	**Receive FIFO Buffer Full Indicator (Read Only)** 0 = Receive FIFO buffer is not full. 1 = Receive FIFO buffer is full.
[8]	**RXEMPTY**	**Receive FIFO Buffer Empty Indicator (Read Only)** 0 = Receive FIFO buffer is not empty. 1 = Receive FIFO buffer is empty.
[7]	**SLVURIF**	**Slave Mode TX Under Run Interrupt Flag** In Slave mode, if TX underflow event occurs and the slave select line goes to inactive state, this interrupt flag will be set to 1. 0 = No Slave TX under run event. 1 = Slave TX under run event occurs. **Note:** This bit will be cleared by writing 1 to it.
[6]	**SLVBEIF**	**Slave Mode Bit Count Error Interrupt Flag** In Slave mode, when the slave select line goes to inactive state, if bit counter is mismatch with DWIDTH, this interrupt flag will be set to 1. 0 = No Slave mode bit count error event. 1 = Slave mode bit count error event occurs. **Note:** If the slave select active but there is no any bus clock input, the SLVBCEIF also be set when the slave select goes to inactive state. This bit will be cleared by writing 1 to it.
[5]	**SLVTOIF**	**Slave Time-out Interrupt Flag (Only Supported in SPI0)** When the slave select is active and the value of SLVTOCNT is not 0, as the bus clock is

		detected, the slave time-out counter in SPI controller logic will be started. When the value of time-out counter is greater than or equal to the value of SLVTOCNT (SPI_SSCTL[31:16]) before one transaction is done, the slave time-out interrupt event will be asserted. 0 = Slave time-out is not active. 1 = Slave time-out is active. **Note:** This bit will be cleared by writing 1 to it.
[4]	**SSLINE**	**Slave Select Line Bus Status (Read Only)** 0 = The slave select line status is 0. 1 = The slave select line status is 1. **Note:** This bit is only available in Slave mode. If SSACTPOL (SPI_SSCTL[2]) is set 0, and the SSLINE is 1, the SPI slave select is in inactive status.
[3]	**SSINAIF**	**Slave Select Inactive Interrupt Flag** 0 = Slave select inactive interrupt be cleared or not occurrs. 1 = Slave select inactive interrupt event occurrs. **Note:** Only available in Slave mode. This bit will be cleared by writing 1 to it.
[2]	**SSACTIF**	**Slave Select Active Interrupt Flag** 0 = Slave select active interrupt be cleared or not occurrs. 1 = Slave select active interrupt event occurrs. **Note:** Only available in Slave mode. This bit will be cleared by writing 1 to it.
[1]	**UNITIF**	**Unit Transfer Interrupt Flag** 0 = No transaction has been finished since this bit was cleared to 0. 1 = SPI controller has finished one unit transfer. **Note:** This bit will be cleared by writing 1 to it.
[0]	**BUSY**	**Busy Status (Read Only)** 0 = SPI controller is in idle state. 1 = SPI controller is in busy state. The following listing are the bus busy conditions: a. SPI_CTL[0] = 1 and the TXEMPTY = 0. b. For SPI Master mode, the TXEMPTY = 1 but the current transaction is not finished yet. c. For SPI Slave mode, the SPI_CTL[0] = 1 and there is serial clock input into the SPI core logic when slave select is active. d. For SPI Slave mode, the SPI_CTL[0] = 1 and the transmit buffer or transmit shift register is not empty even if the slave select is inactive.

SPI Data Transmit Register (SPI_TX)

Register	Offset	R/W	Description	Reset Value
SPI_TX	SPIn_BA+0x20	W	Data Transmit Register	0x0000_0000

31	30	29	28	27	26	25	24
TX							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TX							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TX							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TX							

Bits	Description	
[31:0]	**TX**	**Data Transmit Register** The data transmit registers pass through the transmitted data into the 8-/4-level transmit FIFO buffer. The number of valid bits depends on the setting of DWIDTH (SPI_CTL[12:8]) in SPI mode or WDWIDTH (SPI_I2SCTL[5:4]) in I^2S mode. For example, if DWIDTH is set to 0x08, the bits TX[7:0] will be transmitted. If DWIDTH is set to 0x00, the SPI controller will perform a 32-bit transfer. **Note:** In Master mode, SPI controller will start to transfer after 1 APB clock cycle and 6 peripheral clock cycles after user writes to this register.

SPI Data Receive Register (SPI_RX)

Register	Offset	R/W	Description	Reset Value
SPI_RX	SPIn_BA+0x30	R	Data Receive Register	0x0000_0000

31	30	29	28	27	26	25	24
RX							
23	22	21	20	19	18	17	16
RX							
15	14	13	12	11	10	9	8
RX							
7	6	5	4	3	2	1	0
RX							

Bits	Description	
[31:0]	RX	**Data Receive Register** There are 8-/4-level FIFO buffers in this controller. The data receive register holds the data received from SPI data input pin. If the RXEMPTY (SPI_STATUS[8] or SPI_I2SSTS[8]) is not set to 1, the receive FIFO buffers can be accessed through software by reading this register. This is a read only register.

I²S Control Register (SPI_I2SCTL)

Register	Offset	R/W	Description	Reset Value
SPI_I2SCTL	SPIn_BA+0x60	R/W	I²S Control Register	0x0000_0000

Note: Not supported in SPI mode.

31	30	29	28	27	26	25	24
Reserved		FORMAT		Reserved		LZCIEN	RZCIEN
23	22	21	20	19	18	17	16
RXLCH	Reserved					LZCEN	RZCEN
15	14	13	12	11	10	9	8
MCLKEN	Reserved						SLAVE
7	6	5	4	3	2	1	0
ORDER	MONO	WDWIDTH		MUTE	RXEN	TXEN	I2SEN

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:28]	**FORMAT**	**Data Format Selection** 00 = I²S data format. 01 = MSB justified data format. 10 = PCM mode A. 11 = PCM mode B.
[27:26]	**Reserved**	Reserved.
[25]	**LZCIEN**	**Left Channel Zero-cross Interrupt Enable Bit** Interrupt occurs if this bit is set to 1 and left channel zero-cross event occurs. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[24]	**RZCIEN**	**Right Channel Zero-cross Interrupt Enable Bit** Interrupt occurs if this bit is set to 1 and right channel zero-cross event occurs. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[23]	**RXLCH**	**Receive Left Channel Enable Bit** When monaural format is selected (MONO = 1), I²S controller will receive right channel data if RXLCH is set to 0, and receive left channel data if RXLCH is set to 1. 0 = Receive right channel data in Mono mode. 1 = Receive left channel data in Mono mode.
[22:18]	**Reserved**	Reserved.
[17]	**LZCEN**	**Left Channel Zero Cross Detection Enable Bit** If this bit is set to 1, when left channel data sign bit changes or next shift data bits are all 0 then LZCIF flag in SPI_I2SSTS register will be set to 1. This function is only available in transmit operation. 0 = Left channel zero cross detection Disabled. 1 = Left channel zero cross detection Enabled.

[16]	**RZCEN**	**Right Channel Zero Cross Detection Enable Bit** If this bit is set to 1, when right channel data sign bit change or next shift data bits are all 0 then RZCIF flag in SPI_I2SSTS register will be set to 1. This function is only available in transmit operation. 0 = Right channel zero cross detection Disabled. 1 = Right channel zero cross detection Enabled.
[15]	**MCLKEN**	**Master Clock Enable Bit** If MCLKEN is set to 1, I^2S controller will generate master clock on SPIn_I2SMCLK pin for external audio devices. 0 = Master clock Disabled. 1 = Master clock Enabled.
[14:9]	**Reserved**	Reserved.
[8]	**SLAVE**	**Slave Mode** I^2S can operate as master or slave. For Master mode, I2Sn_BCLK and I2Sn_LRCLK pins are output mode and send bit clock from this chip to Audio CODEC chip. In Slave mode, I2Sn_BCLK and I2Sn_LRCLK pins are input mode and I2Sn_BCLK and I2Sn_LRCLK signals are received from outer Audio CODEC chip. 0 = Master mode. 1 = Slave mode.
[7]	**ORDER**	**Stereo Data Order in FIFO** 0 = Left channel data at high byte. 1 = Left channel data at low byte.
[6]	**MONO**	**Monaural Data** 0 = Data is stereo format. 1 = Data is monaural format.
[5:4]	**WDWIDTH**	**Word Width** 00 = data size is 8-bit. 01 = data size is 16-bit. 10 = data size is 24-bit. 11 = data size is 32-bit.
[3]	**MUTE**	**Transmit Mute Enable Bit** 0 = Transmit data is shifted out from buffer. 1 = Transmit data is zero.
[2]	**RXEN**	**Receive Enable Bit** 0 = Data receive Disabled. 1 = Data receive Enabled.
[1]	**TXEN**	**Transmit Enable Bit** 0 = Data transmit Disabled. 1 = Data transmit Enabled.
[0]	**I2SEN**	**I^2S Controller Enable Bit** 0 = I^2S controller Disabled. 1 = I^2S controller Enabled. **Note:** If set this bit to 1, I2Sn_BCLK will start to output in master mode.

I²S Clock Divider Control Register (SPI_I2SCLK)

Register	Offset	R/W	Description	Reset Value
SPI_I2SCLK	SPIn_BA+0x64	R/W	I²S Clock Divider Control Register	0x0000_0000

Note: Not supported in SPI mode.

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							BCLKDIV
15	14	13	12	11	10	9	8
BCLKDIV							
7	6	5	4	3	2	1	0
Reserved		MCLKDIV					

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16:8]	**BCLKDIV**	**Bit Clock Divider** The I²S controller will generate bit clock in Master mode. The bit clock rate, F_I2SBCLK, is determined by the following expression. F_I2SBCLK = F_I2SCLK /(2x(BCLKDIV + 1)) , where F_I2SCLK is the frequency of I²S source clock determined by SPInSEL setting of CLK_CLKSEL2 register.
[7:6]	**Reserved**	Reserved.
[5:0]	**MCLKDIV**	**Master Clock Divider** If MCLKEN is set to 1, I²S controller will generate master clock for external audio devices. The master clock rate, F_I2SMCLK, is determined by the following expressions. If MCLKDIV >= 1, F_I2SMCLK = F_I2SCLK/(2x(MCLKDIV)). If MCLKDIV = 0, F_I2SMCLK = F_I2SCLK. F_I2SCLK is the frequency of I²S source clock determined by SPInSEL setting of CLK_CLKSEL2 register. In general, the master clock rate is 256 times sampling clock rate.

I²S Status Register (SPI_I2SSTS)

Register	Offset	R/W	Description	Reset Value
SPI_I2SSTS	SPIn_BA+0x68	R/W	I²S Status Register	0x0005_0100

Note: Not supported in SPI mode.

31	30	29	28	27	26	25	24
Reserved	TXCNT			Reserved	RXCNT		
23	22	21	20	19	18	17	16
TXRXRST	Reserved	LZCIF	RZCIF	TXUFIF	TXTHIF	TXFULL	TXEMPTY
15	14	13	12	11	10	9	8
I2SENSTS	Reserved		RXTOIF	RXOVIF	RXTHIF	RXFULL	RXEMPTY
7	6	5	4	3	2	1	0
Reserved			RIGHT	Reserved			

Bits	Description	
[31]	**Reserved**	Reserved.
[30:28]	**TXCNT**	**Transmit FIFO Data Count (Read Only)** This bit field indicates the valid data count of transmit FIFO buffer.
[27]	**Reserved**	Reserved.
[26:24]	**RXCNT**	**Receive FIFO Data Count (Read Only)** This bit field indicates the valid data count of receive FIFO buffer.
[23]	**TXRXRST**	**TX or RX Reset Status (Read Only)** 0 = The reset function of TXRST or RXRST is done. 1 = Doing the reset function of TXRST or RXRST. **Note:** Both the reset operations of TXRST and RXRST need 3 system clock cycles + 3 peripheral clock cycles. User can check the status of this bit to monitor the reset function is doing or done.
[22]	**Reserved**	Reserved.
[21]	**LZCIF**	**Left Channel Zero Cross Interrupt Flag** 0 = No zero cross event occurred on left channel. 1 = Zero cross event occurred on left channel.
[20]	**RZCIF**	**Right Channel Zero Cross Interrupt Flag** 0 = No zero cross event occurred on right channel. 1 = Zero cross event occurred on right channel.
[19]	**TXUFIF**	**Transmit FIFO Underflow Interrupt Flag** When the transmit FIFO buffer is empty and there is no datum written into the FIFO buffer, if there is more bus clock input, this bit will be set to 1. **Note:** This bit will be cleared by writing 1 to it.
[18]	**TXTHIF**	**Transmit FIFO Threshold Interrupt Flag (Read Only)** 0 = The valid data count within the transmit FIFO buffer is larger than the setting value of TXTH. 1 = The valid data count within the transmit FIFO buffer is less than or equal to the setting

		value of TXTH. **Note:** If TXTHIEN = 1 and TXTHIF = 1, the SPI/I^{2}S controller will generate a SPI interrupt request.
[17]	**TXFULL**	**Transmit FIFO Buffer Full Indicator (Read Only)** 0 = Transmit FIFO buffer is not full. 1 = Transmit FIFO buffer is full.
[16]	**TXEMPTY**	**Transmit FIFO Buffer Empty Indicator (Read Only)** 0 = Transmit FIFO buffer is not empty. 1 = Transmit FIFO buffer is empty.
[15]	**I2SENSTS**	**I^{2}S Enable Status (Read Only)** 0 = The SPI/I^{2}S control logic is disabled. 1 = The SPI/I^{2}S control logic is enabled. **Note:** The SPI/I^{2}S peripheral clock is asynchronous with the system clock. In order to make sure the SPI/I^{2}S controller logic is disabled, this bit indicates the real status of SPI/I^{2}S controller logic for user.
[14:13]	**Reserved**	Reserved.
[12]	**RXTOIF**	**Receive Time-out Interrupt Flag** 0 = No receive FIFO time-out event. 1 = Receive FIFO buffer is not empty and no read operation on receive FIFO buffer over 64 SPI clock period in Master mode or over 576 peripheral clock period in Slave mode. When the received FIFO buffer is read by software, the time-out status will be cleared automatically. **Note:** This bit will be cleared by writing 1 to it.
[11]	**RXOVIF**	**Receive FIFO Overrun Interrupt Flag** When the receive FIFO buffer is full, the follow-up data will be dropped and this bit will be set to 1. **Note:** This bit will be cleared by writing 1 to it.
[10]	**RXTHIF**	**Receive FIFO Threshold Interrupt Flag (Read Only)** 0 = The valid data count within the Rx FIFO buffer is smaller than or equal to the setting value of RXTH. 1 = The valid data count within the receive FIFO buffer is larger than the setting value of RXTH. **Note:** If RXTHIEN = 1 and RXTHIF = 1, the SPI/I^{2}S controller will generate a SPI interrupt request.
[9]	**RXFULL**	**Receive FIFO Buffer Full Indicator (Read Only)** 0 = Receive FIFO buffer is not full. 1 = Receive FIFO buffer is full.
[8]	**RXEMPTY**	**Receive FIFO Buffer Empty Indicator (Read Only)** 0 = Receive FIFO buffer is not empty. 1 = Receive FIFO buffer is empty.
[7:5]	**Reserved**	Reserved.
[4]	**RIGHT**	**Right Channel (Read Only)** This bit indicates the current transmit data is belong to which channel. 0 = Left channel. 1 = Right channel.
[3:0]	**Reserved**	Reserved.

6.17 USB Device Controller (USBD)

6.17.1 Overview

There is one set of USB 2.0 full-speed device controller and transceiver in this device. It is compliant with USB 2.0 full-speed device specification and supports Control/Bulk/Interrupt/ Isochronous transfer types.

In this device controller, there are two main interfaces: the APB bus and USB bus which comes from the USB PHY transceiver. For the APB bus, the CPU can program control registers through it. There are 512 bytes internal SRAM as data buffer in this controller. For IN or OUT transfer, it is necessary to write data to SRAM or read data from SRAM through the APB interface or SIE. User needs to set the effective starting address of SRAM for each endpoint buffer through buffer segmentation register (USBD_BUFSEGx).

There are 8 endpoints in this controller. Each of the endpoint can be configured as IN or OUT endpoint. All the operations including Control, Bulk, Interrupt and Isochronous transfer are implemented in this block. The block of “Endpoint Control” is also used to manage the data sequential synchronization, endpoint state, current start address, transaction status, and data buffer status for each endpoint.

There are four different interrupt events in this controller. They are the no-event-wake-up, device plug-in or plug-out event, USB events, like IN ACK, OUT ACK etc, and BUS events, like suspend and resume, etc. Any event will cause an interrupt, and users just need to check the related event flags in interrupt event status register (USBD_INTSTS) to acknowledge what kind of interrupt occurring, and then check the related USB Endpoint Status Register (USBD_EPSTS) to acknowledge what kind of event occurring in this endpoint.

A software-disconnect function is also supported for this USB controller. It is used to simulate the disconnection of this device from the host. If user enables SE0 bit (USBD_SE0), the USB controller will force the output of USB_D+ and USB_D- to level low and its function is disabled. After disable the SE0 bit, host will enumerate this USB device again.

For more information on the Universal Serial Bus, please refer to *Universal Serial Bus Specification Revision 1.1.*

6.17.2 Features

- Compliant with USB 2.0 Full-Speed specification
- Provides 1 interrupt vector with 4 different interrupt events (NEVWK, VBUSDET, USB and BUS)
- Supports Control/Bulk/Interrupt/Isochronous transfer types
- Supports suspend function when no bus activity existing for 3 ms
- Supports 8 endpoints for configurable Control/Bulk/Interrupt/Isochronous transfer types and maximum 512 bytes buffer size
- Provides remote wake-up capability

6.17.3 Block Diagram

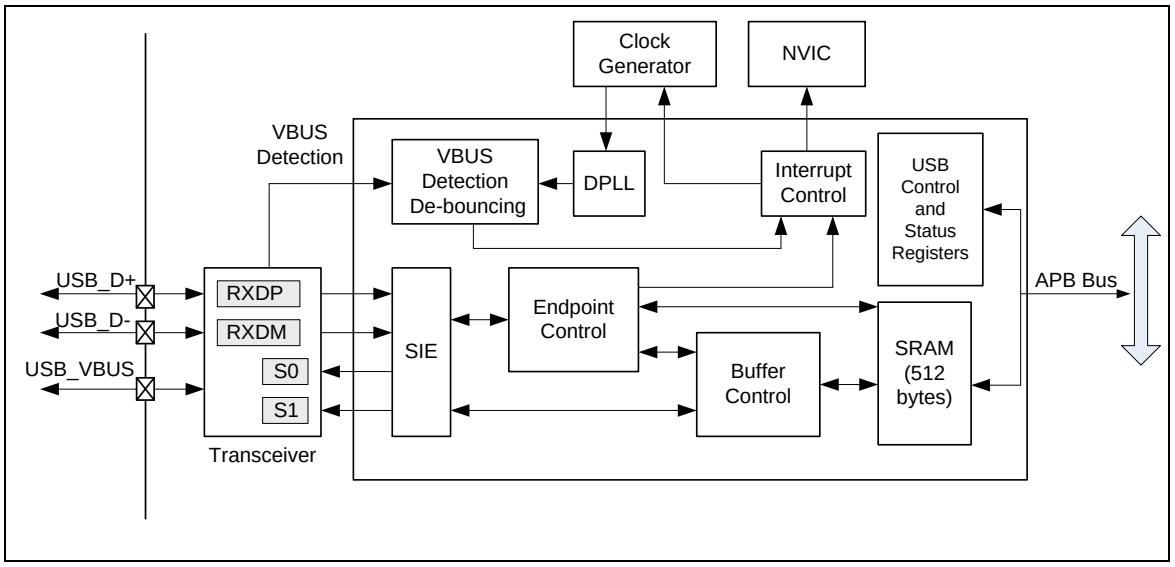


Figure 6.17-1 USB Block Diagram

6.17.4 Basic Configuration

The role of USB frame is determined by USBROLE (SYS_USBPHY[1:0]). The internal USB 3.3V LDO can be enabled by LDO33EN (SYS_USBPHY[8]). These two configurations are write-protection bits. Before writing to these bits, user must disable the register protection function. Refer to the description of SYS_REGLCTL register for details. The USBD clock source is derived from PLL. User has to set the PLL related configurations before USB device controller is enabled. Set the USBDCKEN (CLK_APBCLK0[27]) bit to enable USBD clock and 4-bit pre-scaler USBDIV (CLK_CLKDIV0[7:4]) to generate the proper USBD clock rate.

6.17.5 Functional Description

6.17.5.1 Serial Interface Engine (SIE)

The SIE is the front-end of the device controller and handles most of the USB packet protocol. The SIE typically comprehends signaling up to the transaction level. The functions that it handles could include:

- Packet recognition and transaction sequencing
- SOP, EOP, RESET and RESUME signal detection/generation
- Clock/Data separation
- NRZI Data encoding/decoding and bit stuffing/de-stuffing
- CRC generation and checking (for Token and Data)
- Packet ID (PID) generation and checking/decoding
- Serial-Parallel/Parallel-Serial conversion

6.17.5.2 Endpoint Control

This controller supports 8 endpoints. Each of the endpoint can be configured as Control, Bulk, Interrupt or Isochronous transfer type. All the operations including Control, Bulk, Interrupt and Isochronous transfer are implemented in this block. It is also used to manage the data sequential synchronization, endpoint state control, current endpoint start address, current transaction status, and data buffer status in each endpoint.

6.17.5.3 Digital Phase Lock Loop (DPLL)

The bit rate of USB data is 12 MHz. The DPLL uses the 48 MHz which comes from the clock controller to lock the input data RXDP and RXDM. The 12 MHz bit rate clock is also converted from DPLL.

6.17.5.4 VBUS Detection De-bouncing

A USB device may be plugged-in or plugged-out from the USB host. To monitor the state of a USB device when it is detached from the USB host, the device controller provides hardware de-bouncing for USB VBUS detection interrupt to avoid bounce problems on USB plug-in or unplug. VBUS detection interrupt appears about 10 ms later than USB plug-in or plug-out. User can acknowledge USB plug-in/plug-out by reading USBD_VBUSDET register. The VBUSDET flag represents the current state on the bus without de-bouncing. If VBUSDET is 1, it means the USB cable is plugged-in. If user polls the flag to check USB state, software de-bouncing must be added if needed.

6.17.5.5 Interrupt control

This USB provides 1 interrupt vector with 4 interrupt events (NEVWK, VBUSDET, USB and BUS). The NEVWK event occurs after waking up the system from Power-down mode (The power mode function is defined in system power-down control register, CLK_PWRCTL). The VBUSDET event is used for USB plug-in or unplug. The USB event notifies users of some USB requests, such as IN ACK, OUT ACK., and the BUS event notifies users of some bus events, such as suspend and, resume. The related bits must be set in the interrupt enable register (USBD_INTEN) of USB Device Controller to enable USB interrupts.

NEVWK interrupt is only presented when no the other USB interrupt events happened more than 20ms after the chip is waked up from Power-down mode. After the chip enters Power-down mode, any change on USB_VBUS, USB_D+ and USB_D- can wake up this chip if USB wake-up function is enabled. If this change is not intentionally, no interrupt but NEVWK interrupt will occur. After waking up by USB, this interrupt will occur when no the other USB interrupt events are presented for more than 20ms. The Figure 6.17-2 is the control flow of wake-up interrupt.

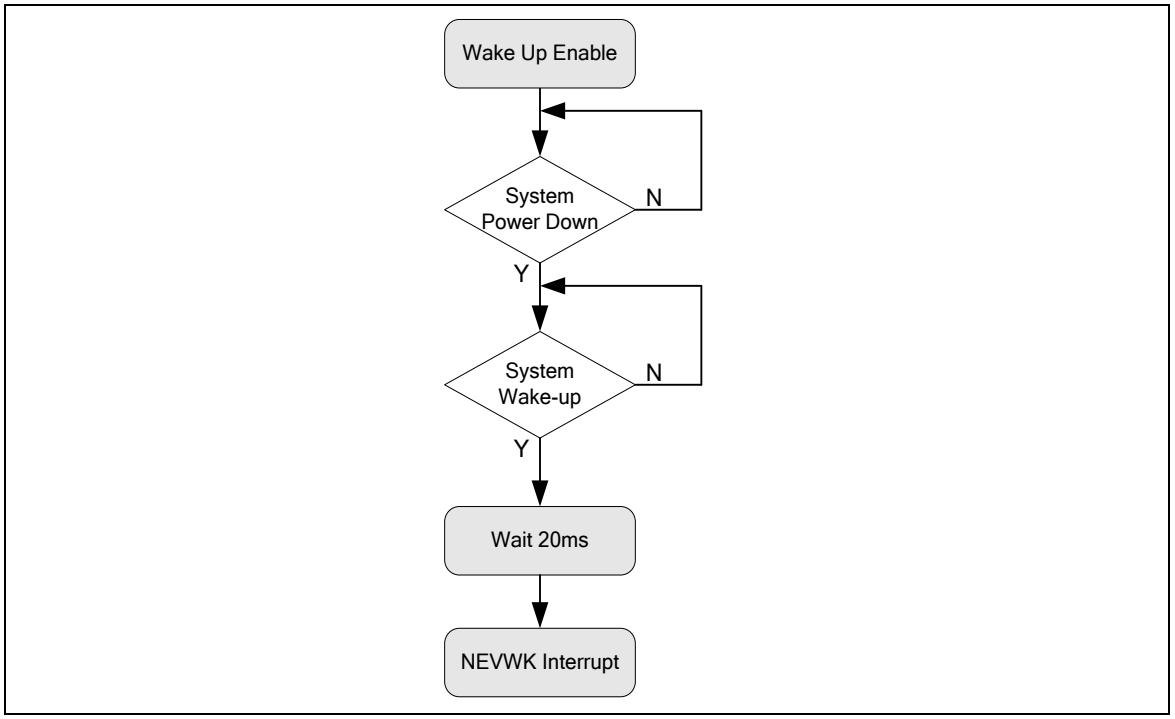


Figure 6.17-2 NEVWK Interrupt Operation Flow

The USB interrupt is used to notify users of any USB event on the bus, and user can read EPSTS (USBD_EPSTS[31:8]) and EPEVT7~0 (USBD_INTSTS[23:16]) to take necessary responses.

Same as USB interrupt, BUS interrupt notifies users of some bus events, like USB reset, suspend, time-out and resume. A user can read USBD_ATTR to acknowledge bus events.

6.17.5.6 Power Saving

User can write 0 to USBD_ATTR[4] to disable PHY under special circumstances, like suspend, to conserve power.

6.17.5.7 Buffer Control

There is 512 bytes SRAM in the controller and the 8 endpoints share this buffer. User shall configure each endpoint's effective starting address in the buffer segmentation register before the USB function active. The "Buffer Control" block is used to control each endpoint's effective starting address and its SRAM size is defined in the USBD_MXPLDx register.

Figure 6.17-3 depicts the starting address for each endpoint according the content of USBD_BUFSEGx and USBD_MXPLDx registers. If the USBD_BUFSEG0 is programmed as 0x08h and USBD_MXPLD0 is set as 0x40h, the SRAM size of endpoint 0 is start from USBD_BA+0x108h and end in USBD_BA+0x148h. (**Note:** The USBD SRAM base is USBD_BA+0x100h).

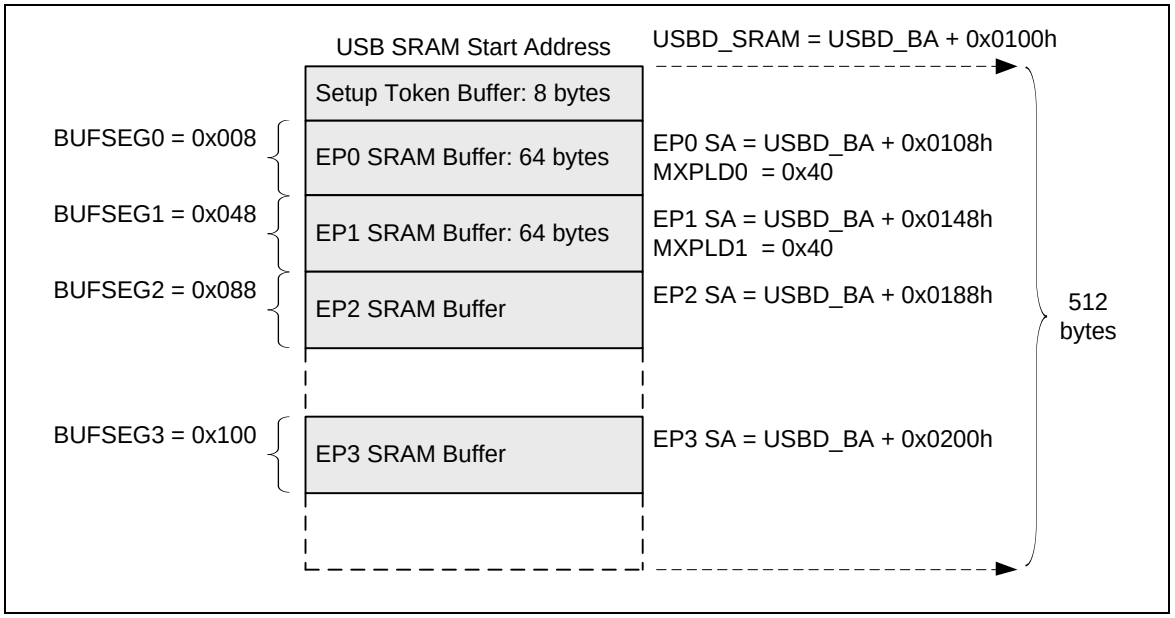


Figure 6.17-3 Endpoint SRAM Structure

6.17.5.8 Handling Transactions with USB Device Peripheral

User can use interrupt or polling USBD_INTSTS to monitor the USB transactions. When transactions occur, USBD_INTSTS will be set by hardware and send an interrupt request to CPU (if related interrupt enabled), or user can polling USBD_INTSTS to get these events without interrupt. The following is the control flow with interrupt enabled.

When USB host has requested data from a device controller, user needs to prepare related data in the specified endpoint buffer in advance. After buffering the required data, user needs to write the actual data length in the specified USBD_MXPLDx register. Once this register is written, the internal signal “In_Rdy” will be asserted and the buffering data will be transmitted immediately after receiving associated IN token from Host. Note that after transferring the specified data, the signal “In_Rdy” will de-assert automatically by hardware.

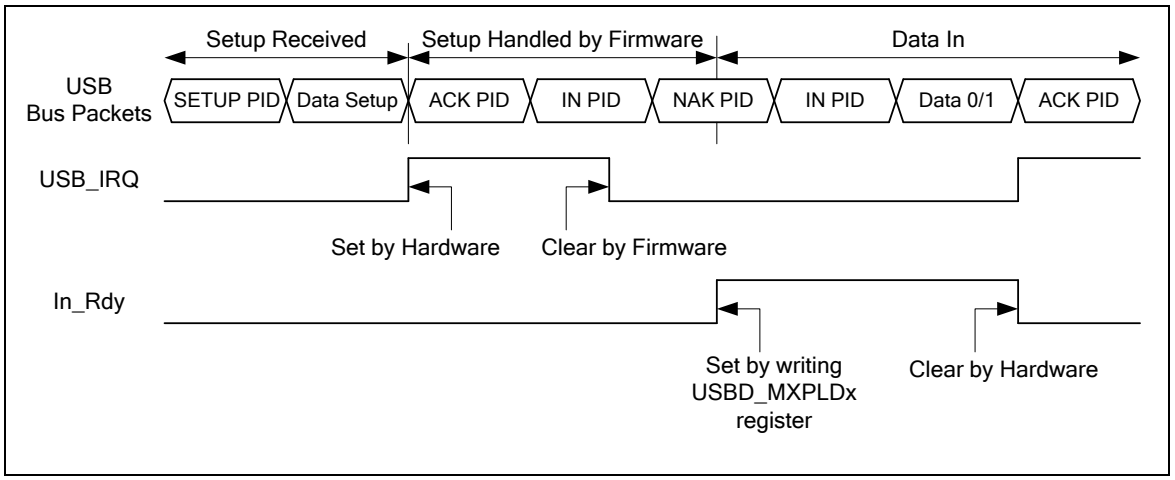


Figure 6.17-4 Setup Transaction Followed by Data IN Transaction

Alternatively, when USB host wants to transmit data to the OUT endpoint in the device controller, hardware will buffer these data to the specified endpoint buffer. After this transaction is completed, hardware will record the data length in specified USBD_MXPLDx register and de-assert the internal signal “Out_Rdy”. This will avoid hardware accepting next transaction until user moves out the current data in the related endpoint buffer. Once users have processed this transaction, the specified USBD_MXPLDx register needs to be written by firmware to assert the signal “Out_Rdy” again to accept the next transaction.

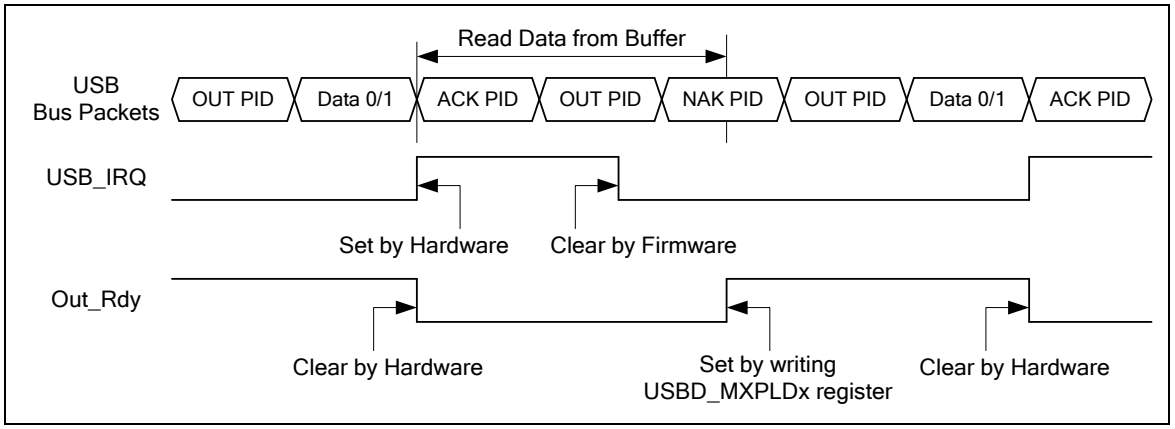


Figure 6.17-5 Data Out Transfer

6.17.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
USBD Base Address: **USBD_BA = 0x400C_0000**				
USBD_INTEN	USBD_BA+0x000	R/W	USB Device Interrupt Enable Register	0x0000_0000
USBD_INTSTS	USBD_BA+0x004	R/W	USB Device Interrupt Event Status Register	0x0000_0000
USBD_FADDR	USBD_BA+0x008	R/W	USB Device Function Address Register	0x0000_0000
USBD_EPSTS	USBD_BA+0x00C	R	USB Device Endpoint Status Register	0x0000_0000
USBD_ATTR	USBD_BA+0x010	R/W	USB Device Bus Status and Attribution Register	0x0000_0040
USBD_VBUSDET	USBD_BA+0x014	R	USB Device VBUS Detection Register	0x0000_0000
USBD_STBUFSEG	USBD_BA+0x018	R/W	USB Setup Token Buffer Segmentation Register	0x0000_0000
USBD_SE0	USBD_BA+0x090	R/W	USB Device Drive SE0 Control Register	0x0000_0001
USBD_BUFSEG0	USBD_BA+0x500	R/W	USB Endpoint 0 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD0	USBD_BA+0x504	R/W	USB Endpoint 0 Maximal Payload Register	0x0000_0000
USBD_CFG0	USBD_BA+0x508	R/W	USB Endpoint 0 Configuration Register	0x0000_0000
USBD_CFGP0	USBD_BA+0x50C	R/W	USB Endpoint 0 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG1	USBD_BA+0x510	R/W	USB Endpoint 1 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD1	USBD_BA+0x514	R/W	USB Endpoint 1 Maximal Payload Register	0x0000_0000
USBD_CFG1	USBD_BA+0x518	R/W	USB Endpoint 1 Configuration Register	0x0000_0000
USBD_CFGP1	USBD_BA+0x51C	R/W	USB Endpoint 1 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG2	USBD_BA+0x520	R/W	USB Endpoint 2 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD2	USBD_BA+0x524	R/W	USB Endpoint 2 Maximal Payload Register	0x0000_0000
USBD_CFG2	USBD_BA+0x528	R/W	USB Endpoint 2 Configuration Register	0x0000_0000
USBD_CFGP2	USBD_BA+0x52C	R/W	USB Endpoint 2 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG3	USBD_BA+0x530	R/W	USB Endpoint 3 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD3	USBD_BA+0x534	R/W	USB Endpoint 3 Maximal Payload Register	0x0000_0000
USBD_CFG3	USBD_BA+0x538	R/W	USB Endpoint 3 Configuration Register	0x0000_0000
USBD_CFGP3	USBD_BA+0x53C	R/W	USB Endpoint 3 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG4	USBD_BA+0x540	R/W	USB Endpoint 4 Buffer Segmentation Register	0x0000_0000

USBD_MXPLD4	USBD_BA+0x544	R/W	USB Endpoint 4 Maximal Payload Register	0x0000_0000
USBD_CFG4	USBD_BA+0x548	R/W	USB Endpoint 4 Configuration Register	0x0000_0000
USBD_CFGP4	USBD_BA+0x54C	R/W	USB Endpoint 4 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG5	USBD_BA+0x550	R/W	USB Endpoint 5 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD5	USBD_BA+0x554	R/W	USB Endpoint 5 Maximal Payload Register	0x0000_0000
USBD_CFG5	USBD_BA+0x558	R/W	USB Endpoint 5 Configuration Register	0x0000_0000
USBD_CFGP5	USBD_BA+0x55C	R/W	USB Endpoint 5 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG6	USBD_BA+0x560	R/W	USB Endpoint 6 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD6	USBD_BA+0x564	R/W	USB Endpoint 6 Maximal Payload Register	0x0000_0000
USBD_CFG6	USBD_BA+0x568	R/W	USB Endpoint 6 Configuration Register	0x0000_0000
USBD_CFGP6	USBD_BA+0x56C	R/W	USB Endpoint 6 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_BUFSEG7	USBD_BA+0x570	R/W	USB Endpoint 7 Buffer Segmentation Register	0x0000_0000
USBD_MXPLD7	USBD_BA+0x574	R/W	USB Endpoint 7 Maximal Payload Register	0x0000_0000
USBD_CFG7	USBD_BA+0x578	R/W	USB Endpoint 7 Configuration Register	0x0000_0000
USBD_CFGP7	USBD_BA+0x57C	R/W	USB Endpoint 7 Set Stall and Clear In/Out Ready Control Register	0x0000_0000

Memory Type	**Address**	**Size**	**Description**
USBD_BA = 0x400C_0000			
USBD_SRAM	USBD_BA+0x100 ~ USBD_BA+0x2FF	512 Bytes	The SRAM is used for the entire endpoints buffer. Refer to section 6.17.5.7 for the endpoint SRAM structure and its description.

6.17.7 Register Description

USB Interrupt Enable Register (USBD_INTEN)

Register	Offset	R/W	Description	Reset Value
USBD_INTEN	USBD_BA+0x000	R/W	USB Device Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
INNAKEN	**Reserved**						**WKEN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**NEVWKIEN**	**VBDETIEN**	**USBIEN**	**BUSIEN**

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**INNAKEN**	**Active NAK Function and Its Status in IN Token** 0 = When device responds NAK after receiving IN token, IN NAK status will not be updated to USBD_EPSTS register, so that the USB interrupt event will not be asserted. 1 = IN NAK status will be updated to USBD_EPSTS register and the USB interrupt event will be asserted, when the device responds NAK after receiving IN token.
[14:9]	**Reserved**	Reserved.
[8]	**WKEN**	**Wake-up Function Enable Bit** 0 = USB wake-up function Disabled. 1 = USB wake-up function Enabled.
[7:4]	**Reserved**	Reserved.
[3]	**NEVWKIEN**	**USB No-event-wake-up Interrupt Enable Bit** 0 = No-event-wake-up Interrupt Disabled. 1 = No-event-wake-up Interrupt Enabled.
[2]	**VBDETIEN**	**VBUS Detection Interrupt Enable Bit** 0 = VBUS detection Interrupt Disabled. 1 = VBUS detection Interrupt Enabled.
[1]	**USBIEN**	**USB Event Interrupt Enable Bit** 0 = USB event interrupt Disabled. 1 = USB event interrupt Enabled.
[0]	**BUSIEN**	**Bus Event Interrupt Enable Bit** 0 = BUS event interrupt Disabled. 1 = BUS event interrupt Enabled.

USB Interrupt Event Status Register (USBD_INTSTS)

Register	Offset	R/W	Description	Reset Value
USBD_INTSTS	USBD_BA+0x004	R/W	USB Device Interrupt Event Status Register	0x0000_0000

31	30	29	28	27	26	25	24
SETUP	Reserved						
23	22	21	20	19	18	17	16
EPEVT7	EPEVT6	EPEVT5	EPEVT4	EPEVT3	EPEVT2	EPEVT1	EPEVT0
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				NEVWKIF	VBDETIF	USBIF	BUSIF

Bits	Description	
[31]	**SETUP**	**Setup Event Status** 0 = No Setup event. 1 = Setup event occurred, cleared by write 1 to USBD_INTSTS[31].
[30:24]	**Reserved**	Reserved.
[23]	**EPEVT7**	**Endpoint 7's USB Event Status** 0 = No event occurred in endpoint 7. 1 = USB event occurred on Endpoint 7, check USBD_EPSTS[31:29] to know which kind of USB event was occurred, cleared by write 1 to USBD_INTSTS[23] or USBD_INTSTS[1].
[22]	**EPEVT6**	**Endpoint 6's USB Event Status** 0 = No event occurred in endpoint 6. 1 = USB event occurred on Endpoint 6, check USBD_EPSTS[28:26] to know which kind of USB event was occurred, cleared by write 1 to USBD_INTSTS[22] or USBD_INTSTS[1].
[21]	**EPEVT5**	**Endpoint 5's USB Event Status** 0 = No event occurred in endpoint 5. 1 = USB event occurred on Endpoint 5, check USBD_EPSTS[25:23] to know which kind of USB event was occurred, cleared by write 1 to USBD_INTSTS[21] or USBD_INTSTS[1].
[20]	**EPEVT4**	**Endpoint 4's USB Event Status** 0 = No event occurred in endpoint 4. 1 = USB event occurred on Endpoint 4, check USBD_EPSTS[22:20] to know which kind of USB event was occurred, cleared by write 1 to USBD_INTSTS[20] or USBD_INTSTS[1].
[19]	**EPEVT3**	**Endpoint 3's USB Event Status** 0 = No event occurred in endpoint 3. 1 = USB event occurred on Endpoint 3, check USBD_EPSTS[19:17] to know which kind of USB event was occurred, cleared by write 1 to USBD_INTSTS[19] or USBD_INTSTS[1].

[18]	**EPEVT2**	**Endpoint 2's USB Event Status** 0 = No event occurred in endpoint 2. 1 = USB event occurred on Endpoint 2, check USBD_EPSTS[16:14] to know which kind of USB event was occurred, cleared by write 1 to USBD_INTSTS[18] or USBD_INTSTS[1].
[17]	**EPEVT1**	**Endpoint 1's USB Event Status** 0 = No event occurred in endpoint 1. 1 = USB event occurred on Endpoint 1, check USBD_EPSTS[13:11] to know which kind of USB event was occurred, cleared by write 1 to USBD_INTSTS[17] or USBD_INTSTS[1].
[16]	**EPEVT0**	**Endpoint 0's USB Event Status** 0 = No event occurred in endpoint 0. 1 = USB event occurred on Endpoint 0, check USBD_EPSTS[10:8] to know which kind of USB event was occurred, cleared by write 1 to USBD_INTSTS[16] or USBD_INTSTS[1].
[15:4]	**Reserved**	Reserved.
[3]	**NEVWKIF**	**No-event-wake-up Interrupt Status** 0 = NEVWK event does not occur. 1 = No-event-wake-up event occurred, cleared by write 1 to USBD_INTSTS[3].
[2]	**VBDETIF**	**VBUS Detection Interrupt Status** 0 = There is not attached/detached event in the USB. 1 = There is attached/detached event in the USB bus and it is cleared by write 1 to USBD_INTSTS[2].
[1]	**USBIF**	**USB Event Interrupt Status** The USB event includes the SETUP Token, IN Token, OUT ACK, ISO IN or ISO OUT events in the bus. 0 = No USB event occurred. 1 = USB event occurred, check EPSTS0~5[2:0] to know which kind of USB event was occurred, cleared by write 1 to USBD_INTSTS[1] or EPSTS0~7 and SETUP (USBD_INTSTS[31]).
[0]	**BUSIF**	**BUS Interrupt Status** The BUS event means that there is one of the suspense or the resume function in the bus. 0 = No BUS event occurred. 1 = Bus event occurred; check USBD_ATTR[3:0] to know which kind of bus event was occurred, cleared by write 1 to USBD_INTSTS[0].

USB Device Function Address Register (USBD_FADDR)

A 7-bit value is used as the address of a device on the USB BUS.

Register	Offset	R/W	Description	Reset Value
USBD_FADDR	USBD_BA+0x008	R/W	USB Device Function Address Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**FADDR**						

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6:0]	**FADDR**	USB Device Function Address

USB Endpoint Status Register (USBD_EPSTS)

Register	Offset	R/W	Description	Reset Value
USBD_EPSTS	USBD_BA+0x00C	R	USB Device Endpoint Status Register	0x0000_0000

31	30	29	28	27	26	25	24
EPSTS7			EPSTS6			EPSTS5	
23	22	21	20	19	18	17	16
EPSTS5	EPSTS4			EPSTS3			EPSTS2
15	14	13	12	11	10	9	8
EPSTS2		EPSTS1			EPSTS0		
7	6	5	4	3	2	1	0
OV	Reserved						

Bits	Description	
[31:29]	**EPSTS7**	**Endpoint 7 Status** These bits are used to indicate the current status of this endpoint 000 = In ACK. 001 = In NAK. 010 = Out Packet Data0 ACK. 011 = Setup ACK. 110 = Out Packet Data1 ACK. 111 = Isochronous transfer end.
[28:26]	**EPSTS6**	**Endpoint 6 Status** These bits are used to indicate the current status of this endpoint 000 = In ACK. 001 = In NAK. 010 = Out Packet Data0 ACK. 011 = Setup ACK. 110 = Out Packet Data1 ACK. 111 = Isochronous transfer end.
[25:23]	**EPSTS5**	**Endpoint 5 Status** These bits are used to indicate the current status of this endpoint 000 = In ACK. 001 = In NAK. 010 = Out Packet Data0 ACK. 011 = Setup ACK. 110 = Out Packet Data1 ACK. 111 = Isochronous transfer end.
[22:20]	**EPSTS4**	**Endpoint 4 Status** These bits are used to indicate the current status of this endpoint 000 = In ACK. 001 = In NAK.

		010 = Out Packet Data0 ACK. 011 = Setup ACK. 110 = Out Packet Data1 ACK. 111 = Isochronous transfer end.
[19:17]	**EPSTS3**	**Endpoint 3 Status** These bits are used to indicate the current status of this endpoint 000 = In ACK. 001 = In NAK. 010 = Out Packet Data0 ACK. 011 = Setup ACK. 110 = Out Packet Data1 ACK. 111 = Isochronous transfer end.
[16:14]	**EPSTS2**	**Endpoint 2 Status** These bits are used to indicate the current status of this endpoint 000 = In ACK. 001 = In NAK. 010 = Out Packet Data0 ACK. 011 = Setup ACK. 110 = Out Packet Data1 ACK. 111 = Isochronous transfer end.
[13:11]	**EPSTS1**	**Endpoint 1 Status** These bits are used to indicate the current status of this endpoint 000 = In ACK. 001 = In NAK. 010 = Out Packet Data0 ACK. 011 = Setup ACK. 110 = Out Packet Data1 ACK. 111 = Isochronous transfer end.
[10:8]	**EPSTS0**	**Endpoint 0 Status** These bits are used to indicate the current status of this endpoint 000 = In ACK. 001 = In NAK. 010 = Out Packet Data0 ACK. 011 = Setup ACK. 110 = Out Packet Data1 ACK. 111 = Isochronous transfer end.
[7]	**OV**	**Overrun** It indicates that the received data is over the maximum payload number or not. 0 = No overrun. 1 = Out Data is more than the Max Payload in MXPLD register or the Setup Data is more than 8 bytes.
[6:0]	**Reserved**	Reserved.

USB Bus Status and Attribution Register (USBD_ATTR)

Register	Offset	R/W	Description	Reset Value
USBD_ATTR	USBD_BA+0x010	R/W	USB Device Bus Status and Attribution Register	0x0000_0040

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**BYTEM**	**Reserved**	**DPPUEN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
USBEN	**Reserved**	**RWAKEUP**	**PHYEN**	**TOUT**	**RESUME**	**SUSPEND**	**USBRST**

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10]	**BYTEM**	**CPU Access USB SRAM Size Mode Selection** 0 = Word mode: The size of the transfer from CPU to USB SRAM can be Word only. 1 = Byte mode: The size of the transfer from CPU to USB SRAM can be Byte only.
[9]	**Reserved**	Reserved.
[8]	**DPPUEN**	**Pull-up Resistor on USB_DP Enable Bit** 0 = Pull-up resistor in USB_D+ bus Disabled. 1 = Pull-up resistor in USB_D+ bus Active.
[7]	**USBEN**	**USB Controller Enable Bit** 0 = USB Controller Disabled. 1 = USB Controller Enabled.
[6]	**Reserved**	Reserved.
[5]	**RWAKEUP**	**Remote Wake-up** 0 = Release the USB bus from K state. 1 = Force USB bus to K (USB_D+ low and USB_D- high) state, used for remote wake-up.
[4]	**PHYEN**	**PHY Transceiver Function Enable Bit** 0 = PHY transceiver function Disabled. 1 = PHY transceiver function Enabled.
[3]	**TOUT**	**Time-out Status** 0 = No time-out. 1 = No Bus response more than 18 bits time. **Note:** This bit is read only.
[2]	**RESUME**	**Resume Status** 0 = No bus resume. 1 = Resume from suspend.

		Note: This bit is read only.
[1]	**SUSPEND**	**Suspend Status** 0 = Bus no suspend. 1 = Bus idle more than 3 ms, either cable is plugged off or host is sleeping. **Note:** This bit is read only.
[0]	**USBRST**	**USB Reset Status** 0 = Bus no reset. 1 = Bus reset when SE0 (single-ended 0) more than 2.5us. **Note:** This bit is read only.

USB Device VBUS Detection Register (USBD_VBUSDET)

Register	Offset	R/W	Description	Reset Value
USBD_VBUSDET	USBD_BA+0x014	R	USB Device VBUS Detection Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**VBUSDET**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**VBUSDET**	**Device VBUS Detection** 0 = Controller is not attached to the USB host. 1 = Controller is attached to the USB host.

USB Setup Token Buffer Segmentation Register (USBD_STBUFSEG)

Register	Offset	R/W	Description	Reset Value
USBD_STBUFSEG	USBD_BA+0x018	R/W	USB Setup Token Buffer Segmentation Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**STBUFSEG**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
STBUFSEG					**Reserved**		

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:3]	**STBUFSEG**	**SETUP Token Buffer Segmentation** It is used to indicate the offset address for the SETUP token with the USB Device SRAM starting address The effective starting address is USBD_SRAM address + {STBUFSEG, 3'b000} Where the USBD_SRAM address = USBD_BA+0x100h. **Note:** It is used for SETUP token only.
[2:0]	**Reserved**	Reserved.

USB Device Drive SE0 Register (USBD_SE0)

Register	Offset	R/W	Description	Reset Value
USBD_SE0	USBD_BA+0x090	R/W	USB Device Drive SE0 Control Register	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**SE0**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**SE0**	**Drive Single Ended Zero in USB Bus** The Single Ended Zero (SE0) is when both lines (USB_D+ and USB_D-) are being pulled low. 0 = Normal operation. 1 = Force USB PHY transceiver to drive SE0.

USB Endpoint Buffer Segmentation Register (USB_BUFSEGx)

Register	Offset	R/W	Description	Reset Value
USBD_BUFSEG0	USBD_BA+0x500	R/W	USB Endpoint 0 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG1	USBD_BA+0x510	R/W	USB Endpoint 1 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG2	USBD_BA+0x520	R/W	USB Endpoint 2 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG3	USBD_BA+0x530	R/W	USB Endpoint 3 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG4	USBD_BA+0x540	R/W	USB Endpoint 4 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG5	USBD_BA+0x550	R/W	USB Endpoint 5 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG6	USBD_BA+0x560	R/W	USB Endpoint 6 Buffer Segmentation Register	0x0000_0000
USBD_BUFSEG7	USBD_BA+0x570	R/W	USB Endpoint 7 Buffer Segmentation Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**BUFSEG**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BUFSEG					**Reserved**		

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:3]	**BUFSEG**	**Endpoint Buffer Segmentation** It is used to indicate the offset address for each endpoint with the USB SRAM starting address The effective starting address of the endpoint is USBD_SRAM address + {BUFSEG[8:3], 3'b000} Where the USBD_SRAM address = USBD_BA+0x100h. Refer to the section 6.17.5.7 for the endpoint SRAM structure and its description.
[2:0]	**Reserved**	Reserved.

USB Endpoint Maximal Payload Register (USB_MXPLDx)

Register	Offset	R/W	Description	Reset Value
USBD_MXPLD0	USBD_BA+0x504	R/W	USB Endpoint 0 Maximal Payload Register	0x0000_0000
USBD_MXPLD1	USBD_BA+0x514	R/W	USB Endpoint 1 Maximal Payload Register	0x0000_0000
USBD_MXPLD2	USBD_BA+0x524	R/W	USB Endpoint 2 Maximal Payload Register	0x0000_0000
USBD_MXPLD3	USBD_BA+0x534	R/W	USB Endpoint 3 Maximal Payload Register	0x0000_0000
USBD_MXPLD4	USBD_BA+0x544	R/W	USB Endpoint 4 Maximal Payload Register	0x0000_0000
USBD_MXPLD5	USBD_BA+0x554	R/W	USB Endpoint 5 Maximal Payload Register	0x0000_0000
USBD_MXPLD6	USBD_BA+0x564	R/W	USB Endpoint 6 Maximal Payload Register	0x0000_0000
USBD_MXPLD7	USBD_BA+0x574	R/W	USB Endpoint 7 Maximal Payload Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**MXPLD**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MXPLD							

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8:0]	**MXPLD**	**Maximal Payload** Define the data length which is transmitted to host (IN token) or the actual data length which is received from the host (OUT token). It also used to indicate that the endpoint is ready to be transmitted out IN token or received in OUT token. (1) When the register is written by CPU, For IN token, the value of MXPLD is used to define the data length to be transmitted and indicate the data buffer is ready. For OUT token, it means that the controller is ready to receive data from the host and the value of MXPLD is the maximal data length comes from host. (2) When the register is read by CPU, For IN token, the value of MXPLD is indicated by the data length be transmitted to host For OUT token, the value of MXPLD is indicated the actual data length received from host. **Note:** Once MXPLD is written, the data packets will be transmitted/received immediately after IN/OUT token arrived.

USB Endpoint Configuration Register (USB_CFGx)

Register	Offset	R/W	Description	Reset Value
USBD_CFG0	USBD_BA+0x508	R/W	USB Endpoint 0 Configuration Register	0x0000_0000
USBD_CFG1	USBD_BA+0x518	R/W	USB Endpoint 1 Configuration Register	0x0000_0000
USBD_CFG2	USBD_BA+0x528	R/W	USB Endpoint 2 Configuration Register	0x0000_0000
USBD_CFG3	USBD_BA+0x538	R/W	USB Endpoint 3 Configuration Register	0x0000_0000
USBD_CFG4	USBD_BA+0x548	R/W	USB Endpoint 4 Configuration Register	0x0000_0000
USBD_CFG5	USBD_BA+0x558	R/W	USB Endpoint 5 Configuration Register	0x0000_0000
USBD_CFG6	USBD_BA+0x568	R/W	USB Endpoint 6 Configuration Register	0x0000_0000
USBD_CFG7	USBD_BA+0x578	R/W	USB Endpoint 7 Configuration Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved						**CSTALL**	**Reserved**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DSQSYNC	**STATE**		**ISOCH**	**EPNUM**			

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**CSTALL**	**Clear STALL Response** 0 = Disable the device to clear the STALL handshake in setup stage. 1 = Clear the device to response STALL handshake in setup stage.
[8]	**Reserved**	Reserved.
[7]	**DSQSYNC**	**Data Sequence Synchronization** 0 = DATA0 PID. 1 = DATA1 PID. **Note:** It is used to specify the DATA0 or DATA1 PID in the following IN token transaction. hardware will toggle automatically in IN token base on the bit.
[6:5]	**STATE**	**Endpoint STATE** 00 = Endpoint is Disabled. 01 = Out endpoint. 10 = IN endpoint. 11 = Undefined.
[4]	**ISOCH**	**Isochronous Endpoint**

		This bit is used to set the endpoint as Isochronous endpoint, no handshaking. 0 = No Isochronous endpoint. 1 = Isochronous endpoint.
[3:0]	**EPNUM**	**Endpoint Number** These bits are used to define the endpoint number of the current endpoint

USB Endpoint Extra Configuration Register (USB_CFGPx)

Register	Offset	R/W	Description	Reset Value
USBD_CFGP0	USBD_BA+0x50C	R/W	USB Endpoint 0 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP1	USBD_BA+0x51C	R/W	USB Endpoint 1 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP2	USBD_BA+0x52C	R/W	USB Endpoint 2 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP3	USBD_BA+0x53C	R/W	USB Endpoint 3 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP4	USBD_BA+0x54C	R/W	USB Endpoint 4 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP5	USBD_BA+0x55C	R/W	USB Endpoint 5 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP6	USBD_BA+0x56C	R/W	USB Endpoint 6 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USBD_CFGP7	USBD_BA+0x57C	R/W	USB Endpoint 7 Set Stall and Clear In/Out Ready Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**SSTALL**	**CLRRDY**

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**SSTALL**	**Set STALL** 0 = Disable the device to response STALL. 1 = Set the device to respond STALL automatically.
[0]	**CLRRDY**	**Clear Ready** When the USBD_MXPLDx register is set by user, it means that the endpoint is ready to transmit or receive data. If the user wants to disable this transaction before the transaction start, users can set this bit to 1 to disable it and it is auto clear to 0. For IN token 0 = No effect. 1 =Clear the IN token had ready to transmit the data to USB host. For OUT token 0 = No effect.

		1 = Clear the OUT token had ready to receive the data from USB host. This bit is write 1 only and is always 0 when it is read back.

6.18 USB 1.1 Host Controller (USBH)

6.18.1 Overview

This chip is equipped with a USB 1.1 Host Controller (USBH) that supports Open Host Controller Interface (OpenHCI, OHCI) Specification, a register-level description of a host controller, to manage the devices and data transfer of Universal Serial Bus (USB).

The USBH supports an integrated Root Hub with a USB port, a DMA for real-time data transfer between system memory and USB bus, port power control and port over current detection.

The USBH is responsible for detecting the connect and disconnect of USB devices, managing data transfer, collecting status and activity of USB bus, providing power control and detecting over current of attached USB devices.

6.18.2 Features

- Supports Universal Serial Bus (USB) Specification Revision 1.1.
- Supports Open Host Controller Interface (OpenHCI) Specification Revision 1.0.
- Supports both full-speed (12Mbps) and low-speed (1.5Mbps) USB devices.
- Supports Control, Bulk, Interrupt and Isochronous transfers.
- Supports an integrated Root Hub.
- Supports a USB host port shared with USB device (OTG function).
- Supports port power control and port over current detection.
- Supports DMA for real-time data transfer.

6.18.3 Block Diagram

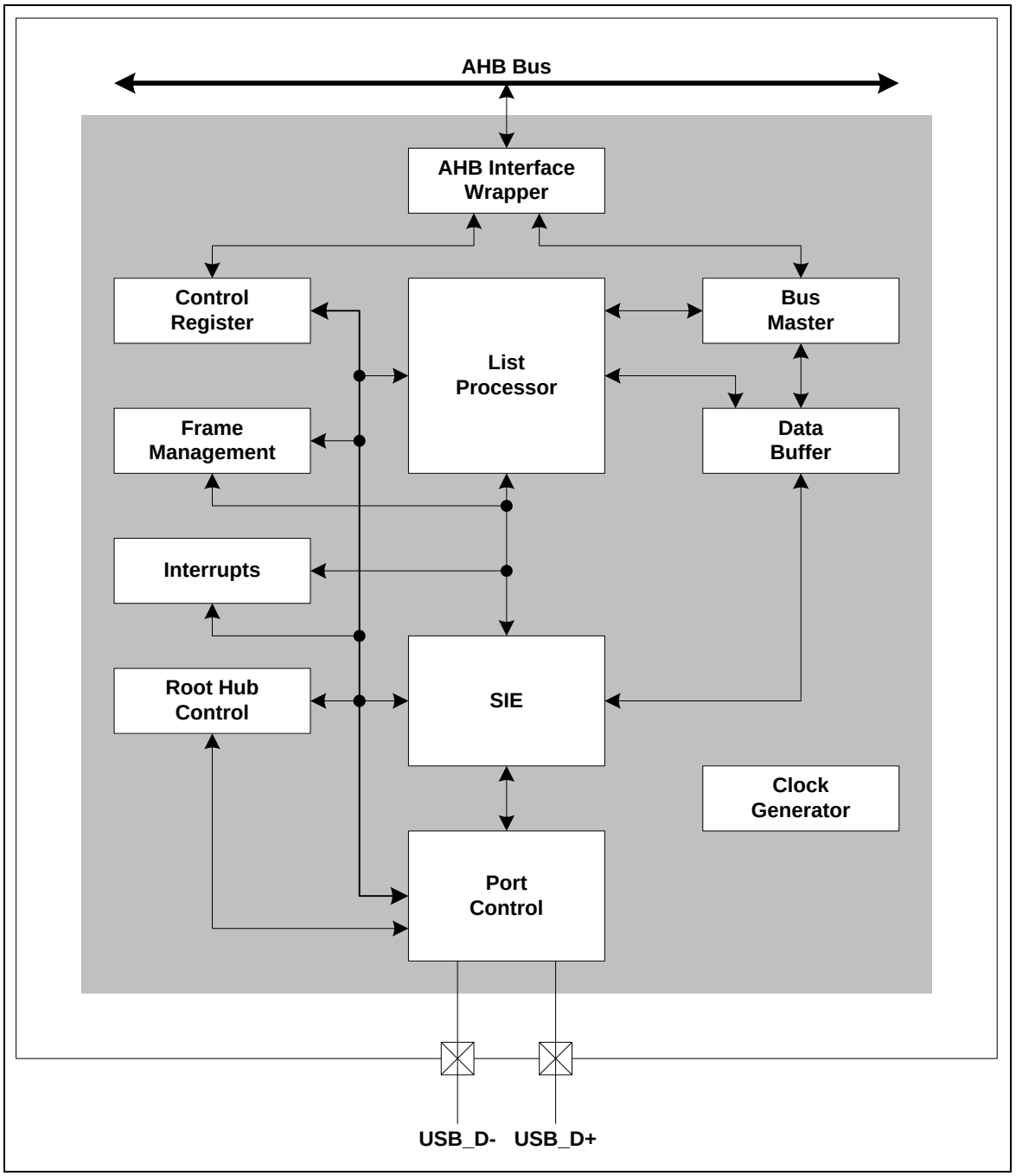


Figure 6.18-1 USB 1.1 Host Controller Block Diagram

6.18.4 Basic Configuration

The USBH clock source is derived from PLL. User has to set the PLL related configurations before USB host controller is enabled. Set the USBHCKEN (CLK_AHBCLK[4]) bit to enable USBH clock and 4-bit pre-scaler USBDIV (CLK_CLKDIV0[7:4]) to generate the proper USBH clock rate. The proper USBH clock rate is 48 MHz.

6.18.5 Functional Description

6.18.5.1 AHB Interface

The OpenHCI Host Controller is connected to the system by the AHB bus. The design requires both master and slave bus operations. As a master, the Host Controller is responsible for running cycles on the AHB bus to access EDs and TDs as well as transferring data between memory and the local data buffer. As a slave, the Host Controller monitors the cycles on the AHB bus and determines when to respond to these cycles. Configuration and non-real-time control access to the Host Controller operational registers are through the AHB bus slave interface.

6.18.5.2 Host Controller

The host controller includes 5 functional blocks, including List Processing, Frame Management, Interrupt Processing, Host Controller Bus Master and Data Buffer.

The List Processor manages the data structures from the Host Controller Driver and coordinates all activity within the Host Controller.

The Frame Management is responsible for managing the frame specific tasks required by the USB specification and the OpenHCI specification. These tasks are:

1) Management of the OpenHCI frame specific Operational Registers
2) Operation of the Largest Data Packet Counter.
3) Performing frame qualifications on USB Transaction requests to the SIE.
4) Generate SOF token requests to the SIE.

Interrupts are the communication method for HC-initiated communication with the Host Controller Driver. There are several events that may trigger an interrupt from the Host Controller. Each specific event sets a specific bit in the HcInterruptStatus register.

The Host Controller Bus Master is the central block in the data path. The Host Controller Bus Master coordinates all access to the AHB Interface. There are two sources of bus mastering within Host Controller: the List Processor and the Data Buffer Engine.

The Data Buffer serves as the data interface between the Bus Master and the SIE. It is a combination of a 64-byte latched based bi-directional asynchronous FIFO and a single DWORD AHB Holding Register.

6.18.5.3 USB Interface

The USB interface includes the integrated Root Hub with an USB port, Port 1 as well as the Serial Interface Engine (SIE) and USB clock generator. The interface combines responsibility for executing bus transactions requested by the HC as well as the hub and port management specified by USB.

The SIE is responsible for managing all transactions to the USB. It controls the bus protocol, packet generation/extraction, data parallel-to-serial conversion, CRC coding, bit stuffing, and NRZI encoding. All transactions on the USB are requested from the List Processor and Frame Manager.

The Root Hub is a collection of ports that are individually controlled and a hub that maintains control/status over functions common to all ports.

6.18.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
USBH Base Address: **USBH_BA = 0x4000_9000**				
HCREVISION	USBH_BA+0x000	R	Host Controller Revision Register	0x0000_0110
HCCONTROL	USBH_BA+0x004	R/W	Host Controller Control Register	0x0000_0000
HCCOMMANDSTATUS	USBH_BA+0x008	R/W	Host Controller CMD Status Register	0x0000_0000
HCINTERRUPTSTATUS	USBH_BA+0x00C	R/W	Host Controller Interrupt Status Register	0x0000_0000
HCINTERRUPTENABLE	USBH_BA+0x010	R/W	Host Controller Interrupt Enable Register	0x0000_0000
HCINTERRUPTDISABLE	USBH_BA+0x014	R/W	Host Controller Interrupt Disable Register	0x0000_0000
HCHCCA	USBH_BA+0x018	R/W	Host Controller Communication Area Register	0x0000_0000
HCPERIODCURRENTED	USBH_BA+0x01C	R/W	Host Controller Period Current ED Register	0x0000_0000
HCCONTROLHEADED	USBH_BA+0x020	R/W	Host Controller Control Head ED Register	0x0000_0000
HCCONTROLCURRENTED	USBH_BA+0x024	R/W	Host Controller Control Current ED Register	0x0000_0000
HCBULKHEADED	USBH_BA+0x028	R/W	Host Controller Bulk Head ED Register	0x0000_0000
HCBULKCURRENTED	USBH_BA+0x02C	R/W	Host Controller Bulk Current ED Register	0x0000_0000
HCDONEHEAD	USBH_BA+0x030	R/W	Host Controller Done Head Register	0x0000_0000
HCFMINTERVAL	USBH_BA+0x034	R/W	Host Controller Frame Interval Register	0x0000_2EDF
HCFMREMAINING	USBH_BA+0x038	R	Host Controller Frame Remaining Register	0x0000_0000
HCFMNUMBER	USBH_BA+0x03C	R	Host Controller Frame Number Register	0x0000_0000
HCPERIODICSTART	USBH_BA+0x040	R/W	Host Controller Periodic Start Register	0x0000_0000
HCLSTHRESHOLD	USBH_BA+0x044	R/W	Host Controller Low-speed Threshold Register	0x0000_0628
HCRHDESCRIPTORA	USBH_BA+0x048	R/W	Host Controller Root Hub Descriptor A Register	0x0000_0902
HCRHDESCRIPTORB	USBH_BA+0x04C	R/W	Host Controller Root Hub Descriptor B Register	0x0000_0000
HCRHSTATUS	USBH_BA+0x050	R/W	Host Controller Root Hub Status Register	0x0000_0000

HCRHPORTSTATUS1	USBH_BA+0x054	R/W	Host Controller Root Hub Port Status [1]	0x0000_0000
HCPHYCONTROL	USBH_BA+0x200	R/W	Host Controller PHY Control Regsiter	0x0000_0000
HCMISCCONTROL	USBH_BA+0x204	R/W	Host Controller Miscellaneous Control Register	0x0000_0000

6.18.7 Register Description

Host Controller Revision Register (HcRevision)

Register	Offset	R/W	Description	Reset Value
HCREVISION	USBH_BA+0x000	R	Host Controller Revision Register	0x0000_0110

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
REV							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**REV**	**Revision Number** Indicates the Open HCI Specification revision number implemented by the Hardware. Host Controller supports 1.1 specification. (X.Y = XYh).

Host Controller Control Register (HcControl)

Register	Offset	R/W	Description	Reset Value
HCCONTROL	USBH_BA+0x004	R/W	Host Controller Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
HCFS		BLE	CLE	IE	PLE	CBSR	

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:6]	**HCFS**	**Host Controller Functional State** This field sets the Host Controller state. The Controller may force a state change from USBSUSPEND to USBRESUME after detecting resume signaling from a downstream port. States are: 00 = USBRESET. 01 = USBRESUME. 10 = USBOPERATIONAL. 11 = USBSUSPEND.
[5]	**BLE**	**Bulk List Enable Bit** 0 = Processing of the Bulk list after next SOF (Start-Of-Frame) Disabled. 1 = Processing of the Bulk list in the next frame Enabled.
[4]	**CLE**	**Control List Enable Bit** 0 = Processing of the Control list after next SOF (Start-Of-Frame) Disabled. 1 = Processing of the Control list in the next frame Enabled.
[3]	**IE**	**Isochronous List Enable Bit** Both ISOEn and PLE (HcControl[2]) high enables Host Controller to process the Isochronous list. Either ISOEn or PLE (HcControl[2]) is low disables Host Controller to process the Isochronous list. 0 = Processing of the Isochronous list after next SOF (Start-Of-Frame) Disabled. 1 = Processing of the Isochronous list in the next frame Enabled, if the PLE (HcControl[2]) is high, too.

[2]	**PLE**	**Periodic List Enable Bit** When set, this bit enables processing of the Periodic (interrupt and isochronous) list. The Host Controller checks this bit prior to attempting any periodic transfers in a frame. 0 = Processing of the Periodic (Interrupt and Isochronous) list after next SOF (Start-Of-Frame) Disabled. 1 = Processing of the Periodic (Interrupt and Isochronous) list in the next frame Enabled. **Note:** To enable the processing of the Isochronous list, user has to set both PLE and IE (HcControl[3]) high.
[1:0]	**CBSR**	**Control Bulk Service Ratio** This specifies the service ratio between Control and Bulk EDs. Before processing any of the non-periodic lists, HC must compare the ratio specified with its internal count on how many nonempty Control EDs have been processed, in determining whether to continue serving another Control ED or switching to Bulk EDs. The internal count will be retained when crossing the frame boundary. In case of reset, HCD is responsible for restoring this Value. 00 = Number of Control EDs over Bulk EDs served is 1:1. 01 = Number of Control EDs over Bulk EDs served is 2:1. 10 = Number of Control EDs over Bulk EDs served is 3:1. 11 = Number of Control EDs over Bulk EDs served is 4:1.

Host Controller CMD Status Register (HcCommandStatus)

Register	Offset	R/W	Description	Reset Value
HCCOMMANDSTATUS	USBH_BA+0x008	R/W	Host Controller CMD Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						SOC	
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved					BLF	CLF	HCR

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17:16]	**SOC**	**Schedule Overrun Count** These bits are incremented on each scheduling overrun error. It is initialized to 00b and wraps around at 11b. This will be incremented when a scheduling overrun is detected even if SO (HcInterruptStatus[0]) has already been set.
[15:3]	**Reserved**	Reserved.
[2]	**BLF**	**Bulk List Filled** Set high to indicate there is an active TD on the Bulk list. This bit may be set by either software or the Host Controller and cleared by the Host Controller each time it begins processing the head of the Bulk list. 0 = No active TD found or Host Controller begins to process the head of the Bulk list. 1 = An active TD added or found on the Bulk list.
[1]	**CLF**	**Control List Filled** Set high to indicate there is an active TD on the Control List. It may be set by either software or the Host Controller and cleared by the Host Controller each time it begins processing the head of the Control List. 0 = No active TD found or Host Controller begins to process the head of the Control list. 1 = An active TD added or found on the Control list.
[0]	**HCR**	**Host Controller Reset** This bit is set to initiate the software reset of Host Controller. This bit is cleared by the Host Controller, upon completed of the reset operation. This bit, when set, didn't reset the Root Hub and no subsequent reset signaling be asserted to its downstream ports. 0 = Host Controller is not in software reset state. 1 = Host Controller is in software reset state.

Host Controller Interrupt Status Register (HcInterruptStatus)

Register	Offset	R/W	Description	Reset Value
HCINTERRUPTSTATUS	USBH_BA+0x00C	R/W	Host Controller Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**RHSC**	**FNO**	**Reserved**	**RD**	**SF**	**WDH**	**SO**

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**RHSC**	**Root Hub Status Change** This bit is set when the content of HcRhStatus or the content of HcRhPortStatus1 register has changed. 0 = The content of HcRhStatus and the content of HcRhPortStatus1 register didn't change. 1 = The content of HcRhStatus or the content of HcRhPortStatus1 register has changed.
[5]	**FNO**	**Frame Number Overflow** This bit is set when bit 15 of Frame Number changes from 1 to 0 or from 0 to 1. 0 = The bit 15 of Frame Number didn't change. 1 = The bit 15 of Frame Number changes from 1 to 0 or from 0 to 1.
[4]	**Reserved**	Reserved.
[3]	**RD**	**Resume Detected** Set when Host Controller detects resume signaling on a downstream port. 0 = No resume signaling detected on a downstream port. 1 = Resume signaling detected on a downstream port.
[2]	**SF**	**Start of Frame** Set when the Frame Management functional block signals a 'Start of Frame' event. Host Control generates a SOF token at the same time. 0 =.Not the start of a frame. 1 =.Indicate the start of a frame and Host Controller generates a SOF token.
[1]	**WDH**	**Write Back Done Head** Set after the Host Controller has written HcDoneHead to HccaDoneHead. Further updates of the HccaDoneHead will not occur until this bit has been cleared. 0 =.Host Controller didn't update HccaDoneHead. 1 =.Host Controller has written HcDoneHead to HccaDoneHead.

[0]	**SO**	**Scheduling Overrun** Set when the List Processor determines a Schedule Overrun has occurred. 0 = Schedule Overrun didn't occur. 1 = Schedule Overrun has occurred.

Host Controller Interrupt Enable Register (HcInterruptEnable)

Register	Offset	R/W	Description	Reset Value
HCINTERRUPTENABLE	USBH_BA+0x010	R/W	Host Controller Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
MIE	Reserved						
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	RHSC	FNO	Reserved	RD	SF	WDH	SO

Bits	Description	
[31]	**MIE**	**Master Interrupt Enable Bit** This bit is a global interrupt enable. A write of '1' allows interrupts to be enabled via the specific enable bits listed above. Write Operation: 0 = No effect. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Enabled if the corresponding bit in HcInterruptEnable is high. Read Operation: 0 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Disabled even if the corresponding bit in HcInterruptEnable is high. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Enabled if the corresponding bit in HcInterruptEnable is high.
[30:7]	**Reserved**	Reserved.
[6]	**RHSC**	**Root Hub Status Change Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Enabled. Read Operation: 0 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Disabled. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Enabled.

[5]	**FNO**	**Frame Number Overflow Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to FNO (HcInterruptStatus[5]) Enabled. Read Operation: 0 = Interrupt generation due to FNO (HcInterruptStatus[5]) Disabled. 1 = Interrupt generation due to FNO (HcInterruptStatus[5]) Enabled.
[4]	**Reserved**	Reserved.
[3]	**RD**	**Resume Detected Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to RD (HcInterruptStatus[3]) Enabled. Read Operation: 0 = Interrupt generation due to RD (HcInterruptStatus[3]) Disabled. 1 = Interrupt generation due to RD (HcInterruptStatus[3]) Enabled.
[2]	**SF**	**Start of Frame Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to SF (HcInterruptStatus[2]) Enabled. Read Operation: 0 = Interrupt generation due to SF (HcInterruptStatus[2]) Disabled. 1 = Interrupt generation due to SF (HcInterruptStatus[2]) Enabled.
[1]	**WDH**	**Write Back Done Head Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to WDH (HcInterruptStatus[1]) Enabled. Read Operation: 0 = Interrupt generation due to WDH (HcInterruptStatus[1]) Disabled. 1 = Interrupt generation due to WDH (HcInterruptStatus[1]) Enabled.
[0]	**SO**	**Scheduling Overrun Enable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to SO (HcInterruptStatus[0]) Enabled. Read Operation: 0 = Interrupt generation due to SO (HcInterruptStatus[0]) Disabled. 1 = Interrupt generation due to SO (HcInterruptStatus[0]) Enabled.

Host Controller Interrupt Disable Register (HcInterruptDisable)

Register	Offset	R/W	Description	Reset Value
HCINTERRUPTDISABLE	USBH_BA+0x014	R/W	Host Controller Interrupt Disable Register	0x0000_0000

31	30	29	28	27	26	25	24
MIE	Reserved						
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	RHSC	FNO	Reserved	RD	SF	WDH	SO

Bits	Description	
[31]	**MIE**	**Master Interrupt Disable Bit** Global interrupt disable. Writing '1' to disable all interrupts. Write Operation: 0 = No effect. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Disabled if the corresponding bit in HcInterruptEnable is high. Read Operation: 0 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Disabled even if the corresponding bit in HcInterruptEnable is high. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Enabled if the corresponding bit in HcInterruptEnable is high.
[30:7]	**Reserved**	Reserved.
[6]	**RHSC**	**Root Hub Status Change Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Disabled. Read Operation: 0 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Disabled. 1 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Enabled.
[5]	**FNO**	**Frame Number Overflow Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to FNO (HcInterruptStatus[5]) Disabled. Read Operation: 0 = Interrupt generation due to FNO (HcInterruptStatus[5]) Disabled. 1 = Interrupt generation due to FNO (HcInterruptStatus[5]) Enabled.

[4]	**Reserved**	Reserved.
[3]	**RD**	**Resume Detected Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to RD (HcInterruptStatus[3]) Disabled. Read Operation: 0 = Interrupt generation due to RD (HcInterruptStatus[3]) Disabled. 1 = Interrupt generation due to RD (HcInterruptStatus[3]) Enabled.
[2]	**SF**	**Start of Frame Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to SF (HcInterruptStatus[2]) Disabled. Read Operation: 0 = Interrupt generation due to SF (HcInterruptStatus[2]) Disabled. 1 = Interrupt generation due to SF (HcInterruptStatus[2]) Enabled.
[1]	**WDH**	**Write Back Done Head Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to WDH (HcInterruptStatus[1]) Disabled. Read Operation: 0 = Interrupt generation due to WDH (HcInterruptStatus[1]) Disabled. 1 = Interrupt generation due to WDH (HcInterruptStatus[1]) Enabled.
[0]	**SO**	**Scheduling Overrun Disable Bit** Write Operation: 0 = No effect. 1 = Interrupt generation due to SO (HcInterruptStatus[0]) Disabled. Read Operation: 0 = Interrupt generation due to SO (HcInterruptStatus[0]) Disabled. 1 = Interrupt generation due to SO (HcInterruptStatus[0]) Enabled.

Host Controller Communication Area Register (HcHCCA)

Register	Offset	R/W	Description	Reset Value
HCHCCA	USBH_BA+0x018	R/W	Host Controller Communication Area Register	0x0000_0000

31	30	29	28	27	26	25	24
HCCA							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
HCCA							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
HCCA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:8]	**HCCA**	**Host Controller Communication Area** Pointer to indicate base address of the Host Controller Communication Area (HCCA).
[7:0]	**Reserved**	Reserved.

Host Controller Period Current ED Register (HcPeriodCurrentED)

Register	Offset	R/W	Description	Reset Value
HCPERIODCURRENTED	USBH_BA+0x01C	R/W	Host Controller Period Current ED Register	0x0000_0000

31	30	29	28	27	26	25	24
PCED							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PCED							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PCED							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PCED				**Reserved**			

Bits	Description	
[31:4]	**PCED**	**Periodic Current ED** Pointer to indicate physical address of the current Isochronous or Interrupt Endpoint Descriptor.
[3:0]	**Reserved**	Reserved.

Host Controller Control Head ED Register (HcControlHeadED)

Register	Offset	R/W	Description	Reset Value
HCCONTROLHEADED	USBH_BA+0x020	R/W	Host Controller Control Head ED Register	0x0000_0000

31	30	29	28	27	26	25	24
CHED							
23	22	21	20	19	18	17	16
CHED							
15	14	13	12	11	10	9	8
CHED							
7	6	5	4	3	2	1	0
CHED				Reserved			

Bits	Description	
[31:4]	CHED	**Control Head ED** Pointer to indicate physical address of the first Endpoint Descriptor of the Control list.
[3:0]	Reserved	Reserved.

Host Controller Control Current ED Register (HcControlCurrentED)

Register	Offset	R/W	Description	Reset Value
HCCONTROLCURRENTED	USBH_BA+0x024	R/W	Host Controller Control Current ED Register	0x0000_0000

31	30	29	28	27	26	25	24
CCED							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CCED							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CCED							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CCED				**Reserved**			

Bits	Description	
[31:4]	**CCED**	**Control Current Head ED** Pointer to indicate the physical address of the current Endpoint Descriptor of the Control list.
[3:0]	**Reserved**	Reserved.

Host Controller Bulk Head ED Register (HcBulkHeadED)

Register	Offset	R/W	Description	Reset Value
HCBULKHEADED	USBH_BA+0x028	R/W	Host Controller Bulk Head ED Register	0x0000_0000

31	30	29	28	27	26	25	24
BHED							
23	22	21	20	19	18	17	16
BHED							
15	14	13	12	11	10	9	8
BHED							
7	6	5	4	3	2	1	0
BHED				Reserved			

Bits	Description	
[31:4]	**BHED**	**Bulk Head ED** Pointer to indicate the physical address of the first Endpoint Descriptor of the Bulk list.
[3:0]	**Reserved**	Reserved.

Host Controller Bulk Current Head ED Register (HcBulkCurrentED)

Register	Offset	R/W	Description	Reset Value
HCBULKCURRENTED	USBH_BA+0x02C	R/W	Host Controller Bulk Current ED Register	0x0000_0000

31	30	29	28	27	26	25	24
BCED							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
BCED							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
BCED							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BCED				**Reserved**			

Bits	Description	
[31:4]	**BCED**	**Bulk Current Head ED** Pointer to indicate the physical address of the current endpoint of the Bulk list.
[3:0]	**Reserved**	Reserved.

Host Controller Done Head Register (HcDoneHead)

Register	Offset	R/W	Description	Reset Value
HCDONEHEAD	USBH_BA+0x030	R/W	Host Controller Done Head Register	0x0000_0000

31	30	29	28	27	26	25	24
DH							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DH							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DH							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DH				**Reserved**			

Bits	Description	
[31:4]	**DH**	**Done Head** Pointer to indicate the physical address of the last completed Transfer Descriptor that was added to the Done queue.
[3:0]	**Reserved**	Reserved.

Host Controller Frame Interval Register (HcFmInterval)

Register	Offset	R/W	Description	Reset Value
HCFMINTERVAL	USBH_BA+0x034	R/W	Host Controller Frame Interval Register	0x0000_2EDF

31	30	29	28	27	26	25	24
FIT	**FSMPS**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
FSMPS							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**FI**					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FI							

Bits	Description	
[31]	**FIT**	**Frame Interval Toggle** This bit is toggled by Host Controller Driver when it loads a new value into FI (HcFmInterval[13:0]). 0 = Host Controller Driver didn't load new value into FI (HcFmInterval[13:0]). 1 = Host Controller Driver loads a new value into FI (HcFmInterval[13:0]).
[30:16]	**FSMPS**	**FS Largest Data Packet** This field specifies a value that is loaded into the Largest Data Packet Counter at the beginning of each frame.
[15:14]	**Reserved**	Reserved.
[13:0]	**FI**	**Frame Interval** This field specifies the length of a frame as (bit times - 1). For 12,000 bit times in a frame, a value of 11,999 is stored here.

Host Controller Frame Remaining Register (HcFmRemaining)

Register	Offset	R/W	Description	Reset Value
HCFMREMAINING	USBH_BA+0x038	R	Host Controller Frame Remaining Register	0x0000_0000

31	30	29	28	27	26	25	24
FRT	Reserved						
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		FR					
7	6	5	4	3	2	1	0
FR							

Bits	Description	
[31]	**FRT**	**Frame Remaining Toggle** This bit is loaded from the FIT (HcFmInterval[31]) whenever FR (HcFmRemaining[13:0]) reaches 0.
[30:14]	**Reserved**	Reserved.
[13:0]	**FR**	**Frame Remaining** When the Host Controller is in the USBOPERATIONAL state, this 14-bit field decrements each 12 MHz clock period. When the count reaches 0, (end of frame) the counter reloads with Frame Interval. In addition, the counter loads when the Host Controller transitions into USBOPERATIONAL.

Host Controller Frame Number Register (HcFmNumber)

Register	Offset	R/W	Description	Reset Value
HCFMNUMBER	USBH_BA+0x03C	R	Host Controller Frame Number Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FN							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FN							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**FN**	**Frame Number** This 16-bit incrementing counter field is incremented coincident with the re-load of FR (HcFmRemaining[13:0]). The count rolls over from 'FFFFh' to '0h.'

Host Controller Periodic Start Register (HcPeriodicStart)

Register	Offset	R/W	Description	Reset Value
HCPERIODICSTART	USBH_BA+0x040	R/W	Host Controller Periodic Start Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		PS					
7	6	5	4	3	2	1	0
PS							

Bits	Description	
[31:14]	Reserved	Reserved.
[13:0]	PS	**Periodic Start** This field contains a value used by the List Processor to determine where in a frame the Periodic List processing must begin.

Host Controller Low-speed Threshold Register (HcLSThreshold)

Register	Offset	R/W	Description	Reset Value
HCLSTHRESHOLD	USBH_BA+0x044	R/W	Host Controller Low-speed Threshold Register	0x0000_0628

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				LST			
7	6	5	4	3	2	1	0
LST							

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**LST**	**Low-speed Threshold** This field contains a value which is compared to the FR (HcFmRemaining[13:0]) field prior to initiating a Low-speed transaction. The transaction is started only if FR (HcFmRemaining[13:0]) >= this field. The value is calculated by Host Controller Driver with the consideration of transmission and setup overhead.

Host Controller Root Hub Descriptor A Register (HcRhDescriptorA)

Register	Offset	R/W	Description	Reset Value
HCRHDESCRIPTORA	USBH_BA+0x048	R/W	Host Controller Root Hub Descriptor A Register	0x0000_0902

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved			NOCP	OCPM	Reserved		PSM
7	6	5	4	3	2	1	0
NDP							

Bits	Description	
[31:13]	**Reserved**	Reserved.
[12]	**NOCP**	**No over Current Protection** This bit describes how the over current status for the Root Hub ports reported. 0 = Over current status is reported. 1 = Over current status is not reported.
[11]	**OCPM**	**over Current Protection Mode** This bit describes how the over current status for the Root Hub ports reported. This bit is only valid when NOCP (HcRhDescriptorA[12]) is cleared. 0 = Global Over current. 1 = Individual Over current.
[10:9]	**Reserved**	Reserved.
[8]	**PSM**	**Power Switching Mode** This bit is used to specify how the power switching of the Root Hub ports is controlled. 0 = Global Switching. 1 = Individual Switching.
[7:0]	**NDP**	**Number Downstream Ports** USB host control supports two downstream ports and only one port is available in this series of chip.

Host Controller Root Hub Descriptor B Register (HcRhDescriptorB)

Register	Offset	R/W	Description	Reset Value
HCRHDESCRIPTORB	USBH_BA+0x04C	R/W	Host Controller Root Hub Descriptor B Register	0x0000_0000

31	30	29	28	27	26	25	24
PPCM							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PPCM							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:16]	**PPCM**	**Port Power Control Mask** Global power switching. This field is only valid if PowerSwitchingMode is set (individual port switching). When set, the port only responds to individual port power switching commands (Set/ClearPortPower). When cleared, the port only responds to global power switching commands (Set/ClearGlobalPower). 0 = Port power controlled by global power switching. 1 = Port power controlled by port power switching. **Note:** PPCM[15:2] and PPCM[0] are reserved.
[15:0]	**Reserved**	Reserved.

Host Controller Root Hub Status Register (HcRhStatus)

Register	Offset	R/W	Description	Reset Value
HCRHSTATUS	USBH_BA+0x050	R/W	Host Controller Root Hub Status Register	0x0000_0000

31	30	29	28	27	26	25	24
CRWE	Reserved						
23	22	21	20	19	18	17	16
Reserved						OCIC	LPSC
15	14	13	12	11	10	9	8
DRWE	Reserved						
7	6	5	4	3	2	1	0
Reserved						OCI	LPS

Bits	Description	
[31]	**CRWE**	**Clear Remote Wake-up Enable Bit** This bit is use to clear DRWE (HcRhStatus[15]). This bit always read as zero. Write Operation: 0 = No effect. 1 = Clear DRWE (HcRhStatus[15]).
[31:18]	**Reserved**	Reserved.
[17]	**OCIC**	**over Current Indicator Change** This bit is set by hardware when a change has occurred in OCI (HcRhStatus[1]). Write 1 to clear this bit to zero. 0 = OCI (HcRhStatus[1]) didn't change. 1 = OCI (HcRhStatus[1]) change.
[16]	**LPSC**	**Set Global Power** In global power mode (PSM (HcRhDescriptorA[8]) = 0), this bit is written to one to enable power to all ports. This bit always read as zero. Write Operation: 0 = No effect. 1 = Set global power.
[15]	**DRWE**	**Device Remote Wakeup Enable Bit** This bit controls if port's Connect Status Change as a remote wake-up event. Write Operation: 0 = No effect. 1 = Connect Status Change as a remote wake-up event Enabled. Read Operation: 0 = Connect Status Change as a remote wake-up event Disabled. 1 = Connect Status Change as a remote wake-up event Enabled.

[14:2]	**Reserved**	Reserved.
[1]	**OCI**	**over Current Indicator** This bit reflects the state of the over current status pin. This field is only valid if NOCP (HcRhDesA[12]) and OCPM (HcRhDesA[11]) are cleared. 0 = No over current condition. 1 = Over current condition.
[0]	**LPS**	**Clear Global Power** In global power mode (PSM (HcRhDescriptorA[8]) = 0), this bit is written to one to clear all ports' power. This bit always read as zero. Write Operation: 0 = No effect. 1 = Clear global power.

Host Controller Root Hub Port Status (HcRhPrt [1])

Register	Offset	R/W	Description	Reset Value
HCRHPORTSTATUS1	USBH_BA+0x054	R/W	Host Controller Root Hub Port Status [1]	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved			PRSC	OCIC	PSSC	PESC	CSC
15	14	13	12	11	10	9	8
Reserved						LSDA	PPS
7	6	5	4	3	2	1	0
Reserved			PRS	POCI	PSS	PES	CCS

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20]	**PRSC**	**Port Reset Status Change** This bit indicates that the port reset signal has completed. Write 1 to clear this bit to zero. 0 = Port reset is not complete. 1 = Port reset is complete.
[19]	**OCIC**	**Port over Current Indicator Change** This bit is set when POCI (HcRhPortStatus1[3]) changes. Write 1 to clear this bit to zero. 0 = POCI (HcRhPortStatus1[3]) didn't change. 1 = POCI (HcRhPortStatus1[3]) changes.
[18]	**PSSC**	**Port Suspend Status Change** This bit indicates the completion of the selective resume sequence for the port. Write 1 to clear this bit to zero. 0 = Port resume is not completed. 1 = Port resume completed.
[17]	**PESC**	**Port Enable Status Change** This bit indicates that the port has been disabled (PES (HcRhPortStatus1[1]) cleared) due to a hardware event. Write 1 to clear this bit to zero. 0 = PES (HcRhPortStatus1[1]) didn't change. 1 = PES (HcRhPortStatus1[1]) changed.

[16]	**CSC**	**Connect Status Change** This bit indicates connect or disconnect event has been detected (CCS (HcRhPortStatus1[0]) changed). Write 1 to clear this bit to zero. 0 = No connect/disconnect event (CCS (HcRhPortStatus1[0]) didn't change). 1 = Hardware detection of connect/disconnect event (CCS (HcRhPortStatus1[0]) changed).
[15:10]	**Reserved**	Reserved.
[9]	**LSDA**	**Low Speed Device Attached (Read) or Clear Port Power (Write)** This bit defines the speed (and bud idle) of the attached device. It is only valid when CCS (HcRhPortStatus1[0]) is set. This bit is also used to clear port power. Write Operation: 0 = No effect. 1 = Clear PPS (HcRhPortStatus1[8]). Read Operation: 0 = Full Speed device. 1 = Low-speed device.
[8]	**PPS**	**Port Power Status** This bit reflects the power state of the port regardless of the power switching mode. Write Operation: 0 = No effect. 1 = Port Power Enabled. Read Operation: 0 = Port power is Diabled. 1 = Port power is Enabled.
[7:5]	**Reserved**	Reserved.
[4]	**PRS**	**Port Reset Status** This bit reflects the reset state of the port. Write Operation: 0 = No effect. 1 = Set port reset. Read Operation 0 = Port reset signal is not active. 1 = Port reset signal is active.
[3]	**POCI**	**Port over Current Indicator (Read) or Clear Port Suspend (Write)** This bit reflects the state of the over current status pin dedicated to this port. This field is only valid if NOCP (HcRhDescriptorA[12]) is cleared and OCPM (HcRhDescriptorA[11]) is set. This bit is also used to initiate the selective result sequence for the port. Write Operation: 0 = No effect. 1 = Clear port suspend. Read Operation: 0 = No over current condition. 1 = Over current condition.

[2]	**PSS**	**Port Suspend Status** This bit indicates the port is suspended Write Operation: 0 = No effect. 1 = Set port suspend. Read Operation: 0 = Port is not suspended. 1 = Port is selectively suspended.
[1]	**PES**	**Port Enable Status** Write Operation: 0 = No effect. 1 = Set port enable. Read Operation: 0 = Port Disabled. 1 = Port Enabled.
[0]	**CCS**	**CurrentConnectStatus (Read) or ClearPortEnable Bit (Write)** Write Operation: 0 = No effect. 1 = Clear port enable. Read Operation: 0 = No device connected. 1 = Device connected.

Host Controller PHY Control Regsiter (HcPhyControl)

Register	Offset	R/W	Description	Reset Value
HCPHYCONTROL	USBH_BA+0x200	R/W	Host Controller PHY Control Regsiter	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				STBYEN	Reserved		
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27]	**STBYEN**	**USB Transceiver Standby Enable Bit** This bit controls if USB transceiver could enter the standby mode to reduce power consumption. 0 = The USB transceiver would never enter the standby mode. 1 = The USB transceiver will enter standby mode while port is in power off state (port power is inactive).
[26:0]	**Reserved**	Reserved.

Host Controller Miscellaneous Control Register (HcMiscControl)

Register	Offset	R/W	Description	Reset Value
HCMISCCONTROL	USBH_BA+0x204	R/W	Host Controller Miscellaneous Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							DPRT1
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved				OCAL	Reserved	ABORT	Reserved

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**DPRT1**	**Disable Port 1** This bit controls if the connection between USB host controller and transceiver of port 1 is disabled. If the connection is disabled, the USB host controller will not recognize any event of USB bus. Set this bit high, the transceiver of port 1 will also be forced into the standby mode no matter what USB host controller operation is. 0 = The connection between USB host controller and transceiver of port 1 Enabled. 1 = The connection between USB host controller and transceiver of port 1 Disabled and the transceiver of port 1 will also be forced into the standby mode.
[15:4]	**Reserved**	Reserved.
[3]	**OCAL**	**over Current Active Low** This bit controls the polarity of over current flag from external power IC. 0 = Over current flag is high active. 1 = Over current flag is low active.
[2]	**Reserved**	Reserved.
[1]	**ABORT**	**AHB Bus ERROR Response** This bit indicates there is an ERROR response received in AHB bus. 0 = No ERROR response received. 1 = ERROR response received.
[0]	**Reserved**	Reserved.

6.19 USB On-The-Go (OTG)

6.19.1 Overview

The OTG controller interfaces to USB PHY and USB controllers which consist of a USB 1.1 host controller and a USB 2.0 FS device controller. The OTG controller supports HNP and SRP protocols defined in the “On-The-Go and Embedded Host Supplement to the USB 2.0 Revision 1.3 Specification”.

USB frame, including USB host, USB device, and OTG controller, can be configured as Host-only, Device-only, ID-dependent or OTG Device mode defined in USBROLE (SYS_USBPHY[1:0]). In Host-only mode, USB frame acts as USB host. USB frame can support both full-speed and low-speed transfer. In Device-only mode, USB frame acts as USB device. USB frame only supports full-speed transfer. In ID-dependent mode, USB frame can be USB Host or USB device depends on USB_ID pin state. In OTG device mode, the role of USB frame depends on the definition of OTG specification. USB frame only supports full-speed transfer when OTG device acts as a peripheral.

6.19.2 Features

- Built in USB PHY
- Configurable to operate as:
  - Host-only
  - Device-only
  - ID-dependent: The role of USB frame is only dependent on USB_ID pin value--as USB Host (USB_ID pin is low) or USB Device (USB_ID pin is high). Not support HNP or SRP protocol.
  - OTG device: dependent on USB_ID pin status to be A-device (USB_ID pin is low) or B-device (USB_ID pin is high). Support HNP and SRP protocols.

6.19.3 Block Diagram

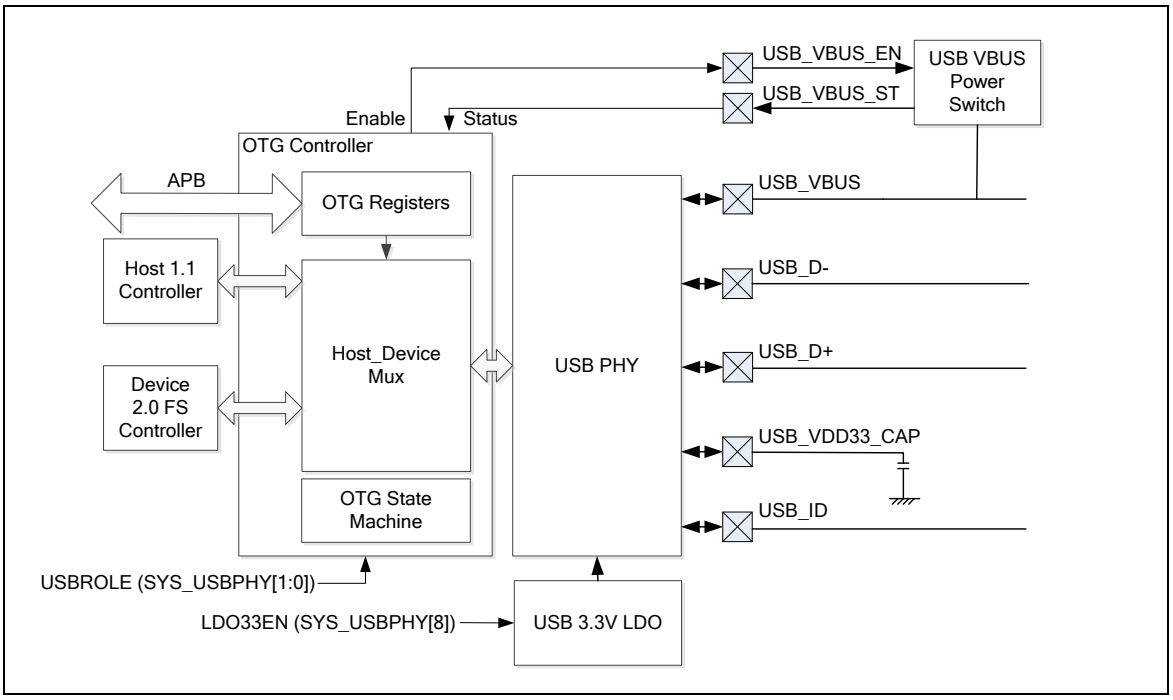


Figure 6.19-1 USB OTG Block Diagram

6.19.4 Basic Configuration

The OTG peripheral clock can be enabled by OTGCKEN (CLK_APBCLK0[26]). The role of USB frame is determined by USBROLE (SYS_USBPHY[1:0]). The internal USB 3.3V LDO can be enabled by LDO33EN (SYS_USBPHY[8]). These two configurations are write-protection bits. Before writing to these bits, user must disable the register protection function. Refer to the description of SYS_REGLCTL register for details. USB_VBUS_EN and USB_VBUS_ST pin functions are configured in SYS_GPA_MFPL or SYS_GPC_MFPL registers.

6.19.5 Functional Description

The role of USB frame depends on the setting of USBROLE (SYS_USBPHY[1:0]) and USB_ID pin status. The USBROLE configuration has precedence over USB_ID pin status. User can configure the OTG controller to USB Host mode, USB Device mode, ID dependent mode or OTG Device mode. In USB Host mode, the host controller will interact with USB PHY directly. In USB Device mode, the device controller will interact with USB PHY directly. In these cases, the OTG controller is used simply as a multiplexer. In ID dependent mode, USB_ID pin status will decide USB frame to act as USB host or USB device. If the USB_ID pin is FALSE state (low level), USB frame will act as USB host. If the USB_ID pin is TRUE state (high level), USB frame will act as USB device. In OTG Device mode, the OTG controller will handle OTG HNP and SRP protocols. If the USB_ID pin is FALSE state (low level), the OTG controller will act as an OTG A-device. If the USB_ID pin is TRUE state (high level), the OTG controller will act as an OTG B-device.

6.19.5.1 The Role of USB Frame

USB Device Mode

When USBROLE (SYS_USBPHY[1:0]) is set to 0, USB frame acts as USB device. USB host function is not available.

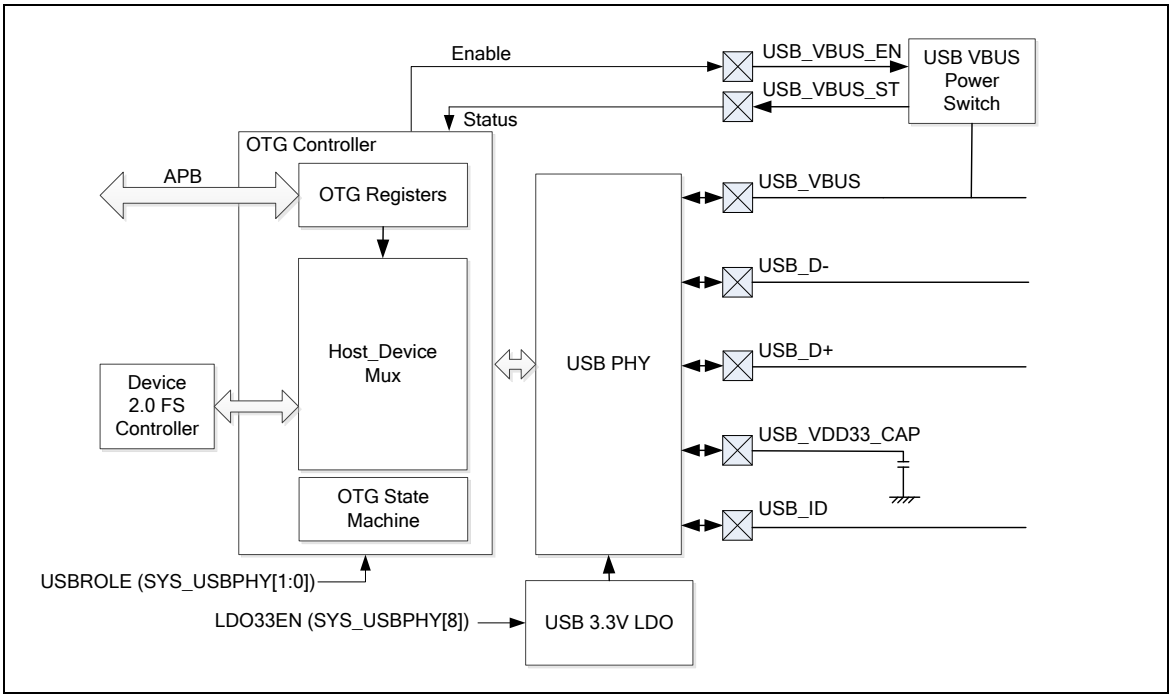


Figure 6.19-2 USB Device Mode

USB Host Mode

When USBROLE (SYS_USBPHY[1:0]) is set to 1, USB frame acts as USB host. USB device function is not available.

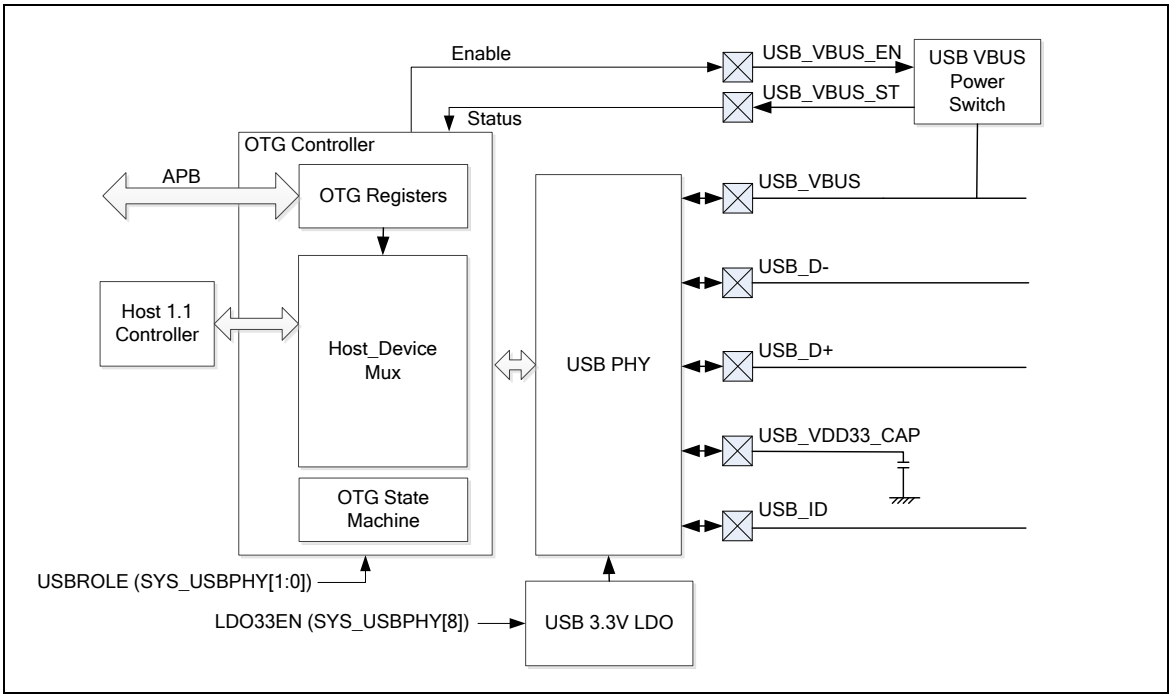


Figure 6.19-3 USB Host Mode

ID Dependent Mode

When USBROLE (SYS_USBPHY[1:0]) is set to 2, the role of USB frame depends on USB_ID pin status. The ID detection function can be enabled by set IDDETEN (OTG_PHYCTL[1]) to 1. The USB_ID pin status reflects on IDSTS (OTG_STATUS[1]). When USB frame acts as USB host (USB_ID pin is low level), the block diagram is the same as USB Host mode. When USB frame acts as USB device (USB_ID pin is high level), the block diagram is the same as USB Device mode.

OTG Device Mode

When USBROLE (SYS_USBPHY[1:0]) is set to 3, the role of USB frame depends on USB_ID pin status. The ID detection function can be enabled by set IDDETEN (OTG_PHYCTL[1]) to 1. The USB_ID pin status reflects on IDSTS (OTG_STATUS[1]). When USB_ID pin status is low level, the OTG controller acts as OTG A-device. When USB_ID pin status is high level, the OTG controller acts as OTG B-device. Please refer to OTG specification to get detail behavior of A-device and B-device.

6.19.5.2 Session Request Protocol (SRP)

When the USB frame is configured as OTG Device mode, OTG controller supports SRP to conserve power. Refer to OTG specification for details of SRP.

A-Device Session Request Protocol

1. A-device turns off USB bus power to conserve power. B-device recognizes such condition by checking VBUS status.
2. B-device requests A-device to supply USB bus power by Data line pulsing when B-device wants to connect to A-device.
3. A-device recognizes USB bus power request through checking SRPDETIF (OTG_INTSTS[13]).
4. A-device starts to drive VBUS by setting BUSREG (OTG_CTL[1]) to 1 once SRPDETIF (OTG_INTSTS[13]) is set to 1 by hardware. If VBUS reaches valid level in specific time interval and B-device is connected, A-device will become USB host, HOSTIF (OTG_INTSTS[7]) will be set to 1. If VBUS cannot reach valid level in specific time interval, it means overcurrent condition occurs. Then VBUS error bit, VBEIF (OTG_IS[1]), will be set to 1.

B-device Session Request Protocol

1. A-device turns off USB bus power to conserve power. B-device recognizes such condition by checking VBUSVLD (OTG_STATUS[5]).
2. B-device can request A-device to supply USB bus power by setting BUSREQ(OTG_CTL[1]) to 1.
3. B-device will generate data line pulsing as defined in OTG specification.
4. A-device will start to drive VBUS after detecting data line pulsing and B-device can recognize such condition by checking VBUSVLD (OTG_STATUS[5]). If A-device drives VBUS to valid level in specific time interval, B-device becomes USB peripheral, PDEVIF (OTG_INTSTS[6]) will be set to 1. If A-device does not drive VBUS to valid level in specific time interval, SRP failure flag, SRPFIF (OTG_INTSTS[2]), will be set to 1 and B-device will go to idle state defined in OTG specification.

6.19.5.3 Host Negotiation Protocol (HNP)

When the USB frame is configured as OTG Device mode, the host function can be transferred between two directly connected OTG device without changing the cable connection. Refer to OTG specification for details of HNP.

A-device Host Negotiation Protocol

1. A-Host defined in OTG specification sends SetFeature b_hnp_enable command to enable B-device HNP capability. B-device responses ACK to indicate B-device supports HNP. User needs to set HNPREQEN (OTG_CTL[2]) to 1 to enable HNP protocol.
2. A-Host goes to a_suspend state by setting BUSREQ (OTG_CTL[1]) to 0 and put USB bus into J-state (USB_D+ high and USB_D- low) when A-Host has finished all desired operations.
3. A-Host becomes A-Peripheral if A-Host detects B-peripheral dis-connected by checking USB_D+ and USB_D- low in specific time interval. If A-Host cannot detect B-Peripheral dis-connected in specific time interval, A-Host will back to idle state.

B-device Host Negotiation Protocol

1. After B-Peripheral receives SetFeature b_hnp_enable command successfully, user enables B-peripheral HNP function by setting HNPREQEN (OTG_CTL[2]) to 1.
2. User sets BUSREG (OTG_CTL[1]) to 1 after detecting USB bus in J-state(USB_D+ high and USB_D- low). Then USB_D+ pull high resistor will be removed to cause USB disconnect state (USB_D+ low and USB_D- low).
3. If B-device detects A-device is connected (USB_D+ high) in specific time interval, B-device will become B-Host. If B-device cannot detect A-device is connected (USB_D+ high) in specific time interval, HNP failure flag, HNPFIF (OTG_INTSTS[3]), will be set to 1.

6.19.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
OTG Base Address: **OTG_BA = 0x4004_D000**				
OTG_CTL	OTG_BA+0x00	R/W	OTG Control Register	0x0000_0000
OTG_PHYCTL	OTG_BA+0x04	R/W	OTG PHY Control Register	0x0000_0000
OTG_INTEN	OTG_BA+0x08	R/W	OTG Interrupt Enable Register	0x0000_0000
OTG_INTSTS	OTG_BA+0x0C	R/W	OTG Interrupt Status Register	0x0000_0000
OTG_STATUS	OTG_BA+0x10	R	OTG Status Register	0x0000_0006

6.19.7 Register Description

OTG Control Register (OTG_CTL)

Register	Offset	R/W	Description	Reset Value
OTG_CTL	OTG_BA+0x00	R/W	OTG Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**WKEN**	**OTGEN**	**Reserved**	**HNPREQEN**	**BUSREQ**	**VBUSDROP**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**WKEN**	**OTG ID Pin Wake-up Enable Bit** 0 = OTG ID pin status change wake-up function Disabled. 1 = OTG ID pin status change wake-up function Enabled.
[4]	**OTGEN**	**OTG Function Enable Bit** User needs to set this bit to enable OTG function while USB frame configured as OTG device. When USB frame not configured as OTG device, this bit is must be low. 0= OTG function Disabled. 1 = OTG function Enabled.
[3]	**Reserved**	Reserved.
[2]	**HNPREQEN**	**OTG HNP Request Enable Bit** When USB frame as A-device, set this bit when A-device allows to process HNP protocol—A-device changes role from Host to Peripheral. This bit will be cleared when OTG state changes from a_suspend to a_peripheral or goes back to a_idle state. When USB frame as B-device, set this bit after the OTG A-device successfully sends a SetFeature (b_hnp_enable) command to the OTG B-device to start role change—B-device changes role from Peripheral to Host. This bit will be cleared when OTG state changes from b_peripheral to b_wait_acon or goes back to b_idle state. 0 = HNP request Disabled. 1 = HNP request Enabled (A-device can change role from Host to Peripheral or B-device can change role from Peripheral to Host). **Note:** Refer to OTG specification to get a_suspend, a_peripheral, a_idle and b_idle state.
[1]	**BUSREQ**	**OTG Bus Request** If OTG A-device wants to do data transfers via USB bus, setting this bit will drive VBUS high to detect USB device connection. If user won't use the bus any more, clearing this bit will drop VBUS to save power. This bit will be cleared when A-device goes to A_wait_vfall state. This bit will be also cleared if VBUSDROP (OTG_CTL[0]) bit is set or IDSTS (OTG_STATUS[1]) changed. If user of an OTG-B Device wants to request VBUS, setting this bit will run SRP protocol. This bit will be cleared if SRP failure (OTG A-device does not provide VBUS after B-

Bits	Description	
		device issues ARP in specified interval, defined in OTG specification). This bit will be also cleared if VBUSDROP (OTG_CTL[0]) bit is set IDSTS (OTG_STATUS[1]) changed. 0 = Not launch VBUS in OTG A-device or not request SRP in OTG B-device. 1 = Launch VBUS in OTG A-device or request SRP in OTG B-device.
[0]	**VBUSDROP**	**Drop VBUS Control** If user application running on this OTG A-device wants to conserve power, set this bit to drop VBUS. BUSREQ (OTG_CTL[1]) will be also cleared no matter A-device or B-device. 0 = Not drop the VBUS. 1 = Drop the VBUS.

OTG PHY Control Register (OTG_PHYCTL)

Register	Offset	R/W	Description	Reset Value
OTG_PHYCTL	OTG_BA+0x04	R/W	OTG PHY Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		VBSTSPOL	VBENPOL	Reserved		IDDETEN	OTGPHYEN

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**VBSTSPOL**	**Off-chip USB VBUS Power Switch Status Polarity** The polarity of off-chip USB VBUS power switch valid signal depends on the selected component. A USB_VBUS_ST pin is used to monitor the valid signal of the off-chip USB VBUS power switch. Set this bit as following according to the polarity of off-chip USB VBUS power switch. 0 = The polarity of off-chip USB VBUS power switch valid status is high. 1 = The polarity of off-chip USB VBUS power switch valid status is low.
[4]	**VBENPOL**	**Off-chip USB VBUS Power Switch Enable Polarity** The OTG controller will enable off-chip USB VBUS power switch to provide VBUS power when need. A USB_VBUS_EN pin is used to control the off-chip USB VBUS power switch. The polarity of enabling off-chip USB VBUS power switch (high active or low active) depends on the selected component. Set this bit as following according to the polarity of off-chip USB VBUS power switch. 0 = The off-chip USB VBUS power switch enable is active high. 1 = The off-chip USB VBUS power switch enable is active low.
[3:2]	**Reserved**	Reserved.
[1]	**IDDETEN**	**ID Detection Enable Bit** 0 = Detect ID pin status Disabled. 1 = Detect ID pin status Enabled.
[0]	**OTGPHYEN**	**OTG PHY Enable Bit** When USB frame is configured as OTG-device, user needs to set this bit before using OTG function. If device is not configured as OTG-device, this bit is “don’t care”. 0 = OTG PHY Disabled. 1 = OTG PHY Enabled.

OTG Interrupt Enable Register (OTG_INTEN)

Register	Offset	R/W	Description	Reset Value
OTG_INTEN	OTG_BA+0x08	R/W	OTG Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		SRPDETIEN	Reserved	SECHGIEN	VBCHGIEN	AVLDCHGIEN	BVLDCHGIEN
7	6	5	4	3	2	1	0
HOSTIEN	PDEVIEN	IDCHGIEN	GOIDLEIEN	HNPFIEN	SRPFIEN	VBEIEN	ROLECHGIEN

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**SRPDETIEN**	**SRP Detected Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[12]	**Reserved**	Reserved.
[11]	**SECHGIEN**	**SESSEND Status Changed Interrupt Enable Bit** If this bit is set to 1 and SESSEND (OTG_STATUS[2]) status is changed from high to low or from low to high, a interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[10]	**VBCHGIEN**	**VBUSVLD Status Changed Interrupt Enable Bit** If this bit is set to 1 and VBUSVLD (OTG_STATUS[5]) status is changed from high to low or from low to high, a interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[9]	**AVLDCHGIEN**	**A-device Session Valid Status Changed Interrupt Enable Bit** If this bit is set to 1 and AVLD (OTG_STATUS[4]) status is changed from high to low or from low to high, a interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[8]	**BVLDCHGIEN**	**B-device Session Valid Status Changed Interrupt Enable Bit** If this bit is set to 1 and BVLD (OTG_STATUS[3]) status is changed from high to low or from low to high, a interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[7]	**HOSTIEN**	**Act As Host Interrupt Enable Bit** If this bit is set to 1 and the device is changed as a host, a interrupt will be asserted.

Bits	Description	
		0 = This device as a host interrupt Disabled. 1 = This device as a host interrupt Enabled.
[6]	**PDEVIEN**	**Act As Peripheral Interrupt Enable Bit** If this bit is set to 1 and the device is changed as a peripheral, a interrupt will be asserted. 0 = This device as a peripheral interrupt Disabled. 1 = This device as a peripheral interrupt Enabled.
[5]	**IDCHGIEN**	**IDSTS Changed Interrupt Enable Bit** If this bit is set to 1 and IDSTS (OTG_STATUS[1]) status is changed from high to low or from low to high, a interrupt will be asserted. 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[4]	**GOIDLEIEN**	**OTG Device Goes to IDLE State Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** Going to idle state means going to a_idle or b_idle state. Please refer to A-device state diagram and B-device state diagram in OTG spec.
[3]	**HNPFIEN**	**HNP Fail Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[2]	**SRPFIEN**	**SRP Fail Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[1]	**VBEIEN**	**VBUS Error Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled. **Note:** VBUS error means going to a_vbus_err state. Please refer to A-device state diagram in OTG spec.
[0]	**ROLECHGIEN**	**Role (Host or Peripheral) Changed Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled.

OTG Interrupt Status Register (OTG_INTSTS)

Register	Offset	R/W	Description	Reset Value
OTG_INTSTS	OTG_BA+0x0C	R/W	OTG Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		SRPDETIF	Reserved	SECHGIF	VBCHGIF	AVLDCHGIF	BVLDCHGIF
7	6	5	4	3	2	1	0
HOSTIF	PDEVIF	IDCHGIF	GOIDLEIF	HNPFIF	SRPFIF	VBEIF	ROLECHGIF

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**SRPDETIF**	**SRP Detected Interrupt Status** 0 = SRP not detected. 1 = SRP detected. **Note:** Write 1 to clear this status.
[12]		**Note:**
[11]	**SECHGIF**	**SESSEND State Change Interrupt Status** 0 = SESSEND (OTG_STATUS[2]) not toggled. 1 = SESSEND (OTG_STATUS[2]) from high to low or from low to high. **Note:** Write 1 to clear this flag.
[10]	**VBCHGIF**	**VBUSVLD State Change Interrupt Status** 0 = VBUSVLD (OTG_STATUS[5]) not toggled. 1 = VBUSVLD (OTG_STATUS[5]) from high to low or from low to high. **Note:** Write 1 to clear this status.
[9]	**AVLDCHGIF**	**A-device Session Valid State Change Interrupt Status** 0 = AVLD (OTG_STATUS[4]) not toggled. 1 = AVLD (OTG_STATUS[4]) from high to low or low to high. **Note:** Write 1 to clear this status.
[8]	**BVLDCHGIF**	**B-device Session Valid State Change Interrupt Status** 0 = BVLD (OTG_STATUS[3]) is not toggled. 1 = BVLD (OTG_STATUS[3]) from high to low or low to high. **Note:** Write 1 to clear this status.
[7]	**HOSTIF**	**Act As Host Interrupt Status** 0= This device does not act as a host. 1 = This device acts as a host. **Note:** Write 1 to clear this flag.

Bits	Description	
[6]	**PDEVIF**	**Act As Peripheral Interrupt Status** 0= This device does not act as a peripheral. 1 = This device acts as a peripheral. **Note:** Write 1 to clear this flag.
[5]	**IDCHGIF**	**ID State Change Interrupt Status** 0 = IDSTS (OTG_STATUS[1]) not toggled. 1 = IDSTS (OTG_STATUS[1]) from high to low or from low to high. **Note:** Write 1 to clear this flag.
[4]	**GOIDLEIF**	**OTG Device Goes to IDLE Interrupt Status** Flag is set if the OTG device transfers from non-idle state to idle state. The OTG device will be neither a host nor a peripheral. 0 = OTG device does not go back to idle state (a_idle or b_idle). 1 = OTG device goes back to idle state(a_idle or b_idle). **Note 1:** Going to idle state means going to a_idle or b_idle state. Please refer to OTG specification. **Note 2:** Write 1 to clear this flag.
[3]	**HNPFIF**	**HNP Fail Interrupt Status** When A-device has granted B-device to be host and USB bus is in SE0 (both USB_D+ and USB_D- low) state, this bit will be set when A-device does not connect after specified interval expires. 0 = A-device connects to B-device before specified interval expires. 1 = A-device does not connect to B-device before specified interval expires. **Note:** Write 1 to clear this flag.
[2]	**SRPFIF**	**SRP Fail Interrupt Status** After initiating SRP, an OTG B-device will wait for the OTG A-device to drive VBUS high at least TB_SRP_FAIL minimum, defined in OTG specification. This flag is set when the OTG B-device does not get VBUS high after this interval. 0 = OTG B-device gets VBUS high before this interval. 1 = OTG B-device does not get VBUS high before this interval. **Note:** Write 1 to clear this flag.
[1]	**VBEIF**	**VBUS Error Interrupt Status** This bit will be set when voltage on VBUS cannot reach a minimum valid threshold 4.4V within a maximum time of 100ms after OTG A-device starting to drive VBUS high. 0 = OTG A-device drives VBUS over threshold voltage before this interval expires. 1 = OTG A-device cannot drive VBUS over threshold voltage before this interval expires. **Note:** Write 1 to clear this flag and recover from the VBUS error state.
[0]	**ROLECHGIF**	**OTG Role Change Interrupt Status** This flag is set when the role of an OTG device changed from a host to a peripheral, or changed from a peripheral to a host while USB_ID pin status does not change. 0 = OTG device role not changed. 1 = OTG device role changed. **Note:** Write 1 to clear this flag.

OTG Functional Status Register (OTG_STATUS)

Register	Offset	R/W	Description	Reset Value
OTG_STATUS	OTG_BA+0x10	R	OTG Status Register	0x0000_0006

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		VBUSVLD	AVLD	BVLD	SESSEND	IDSTS	OVERCUR

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**VBUSVLD**	**VBUS Valid Status** When VBUS is larger than 4.7V, this bit will be set to 1. 0 = VBUS is not valid. 1 = VBUS is valid.
[4]	**AVLD**	**A-device Session Valid Status** 0 = A-device session is not valid. 1 = A-device session is valid.
[3]	**BVLD**	**B-device Session Valid Status** 0 = B-device session is not valid. 1 = B-device session is valid.
[2]	**SESSEND**	**Session End Status** When VBUS voltage is lower than 0.4V, this bit will be set to 1. Session end means no meaningful power on VBUS. 0 = Session is not end. 1 = Session is end.
[1]	**IDSTS**	**USB_ID Pin State of Mini-b/Micro-plug** 0 = Mini-A/Micro-A plug is attached. 1 = Mini-B/Micro-B plug is attached.
[0]	**OVERCUR**	**over Current Condition** The voltage on VBUS cannot reach a minimum VBUS valid threshold, 4.4V minimum, within a maximum time of 100ms after OTG A-device drives VBUS high. 0 = OTG A-device drives VBUS successfully. 1 = OTG A-device cannot drives VBUS high in this interval.

6.20 Controller Area Network (CAN)

6.20.1 Overview

The C_CAN consists of the CAN Core, Message RAM, Message Handler, Control Registers and Module Interface (Refer to Figure 6.20-1 CAN Peripheral Block Diagram) The CAN Core performs communication according to the CAN protocol version 2.0 part A and B. The bit rate can be programmed to values up to 1MBit/s. For the connection to the physical layer, additional transceiver hardware is required.

For communication on a CAN network, individual Message Objects are configured. The Message Objects and Identifier Masks for acceptance filtering of received messages are stored in the Message RAM. All functions concerning the handling of messages are implemented in the Message Handler. These functions include acceptance filtering, the transfer of messages between the CAN Core and the Message RAM, and the handling of transmission requests as well as the generation of the module interrupt.

The register set of the C_CAN can be accessed directly by the software through the module interface. These registers are used to control/configure the CAN Core and the Message Handler and to access the Message RAM.

6.20.2 Features

- Supports CAN protocol version 2.0 part A and B
- Bit rates up to 1 MBit/s
- 32 Message Objects
- Each Message Object has its own identifier mask
- Programmable FIFO mode (concatenation of Message Objects)
- Maskable interrupt
- Disabled Automatic Re-transmission mode for Time Triggered CAN applications
- Programmable loop-back mode for self-test operation
- 16-bit module interfaces to the AMBA APB bus
- Supports wake-up function

6.20.3 Basic Configuration

The basic configurations of CAN are as follows.

- CAN pins are configured on SYS_GPA_MFPL, SYS_GPA_MFPH and SYS_GPC_MFPL registers.
- Enable CAN clock (CAN_EN (APBCLK0[24])).
- Reset CAN controller (CAN_RST (IPRSTC2[24])).

6.20.4 Block Diagram

The C_CAN interfaces with the AMBA APB bus. The Figure 6.20-1 shows the block diagram of the C_CAN.

- **CAN Core**

CAN Protocol Controller and Rx/Tx Shift Register for serial/parallel conversion of messages.

- **Message RAM**

Stores Message Objects and Identifier Masks

- **Registers**

All registers used to control and to configure the C_CAN.

- **Message Handler**

State Machine that controls the data transfer between the Rx/Tx Shift Register of the CAN Core and the Message RAM as well as the generation of interrupts as programmed in the Control and Configuration Registers.

- **Module Interface**

C_CAN interfaces to the AMBA APB 16-bit bus from ARM.

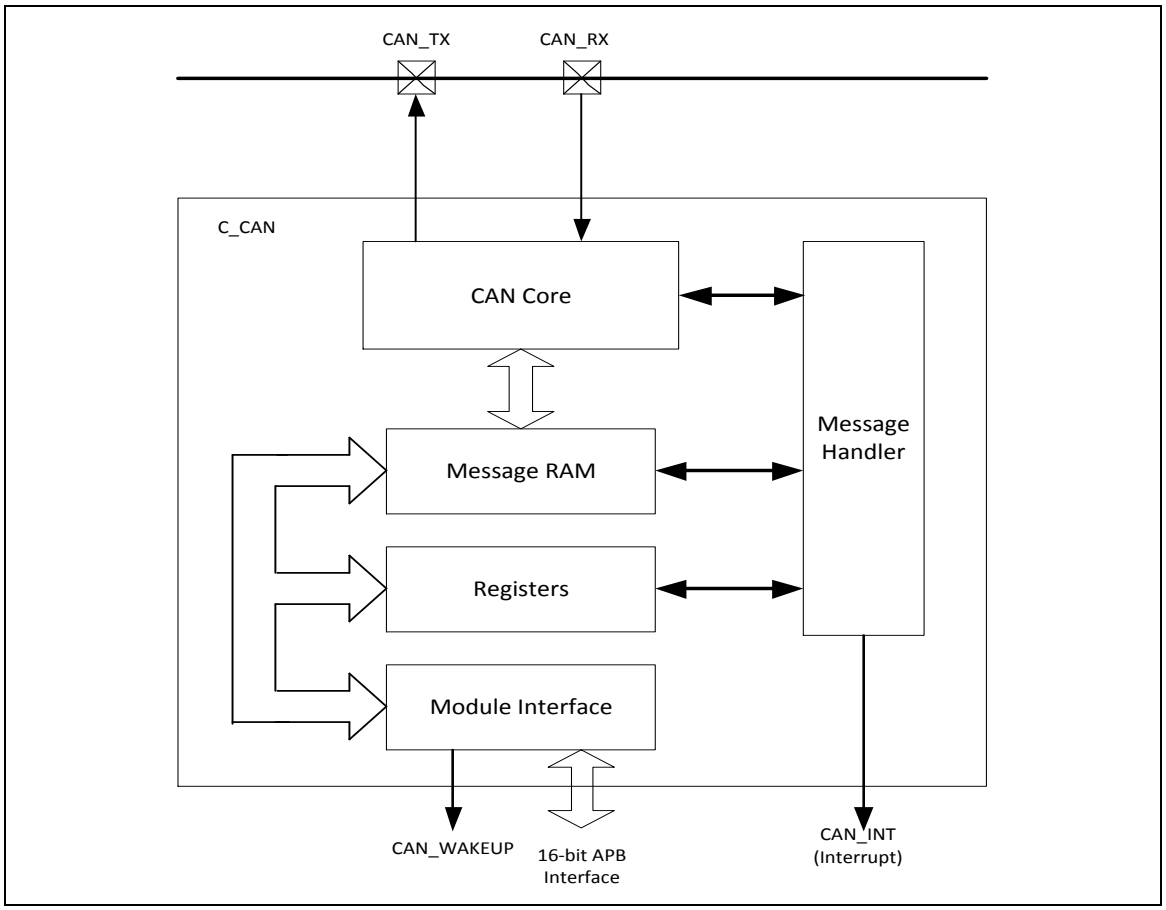


Figure 6.20-1 CAN Peripheral Block Diagram

6.20.5 Functional Description

6.20.5.1 Software Initialization

The software initialization is started by setting the Init bit (CAN_CON[0]), either by a software or a hardware reset, or by going to bus-off state.

While the Init bit is set, all messages transfer to and from the CAN bus are stopped and the status of the CAN_TX output pin is recessive (HIGH). The Error Management Logic (EML) counters are unchanged. Setting the Init bit does not change any configuration register.

To initialize the CAN Controller, software has to set up the Bit Timing Register and each Message Object. If a Message Object is not required, the corresponding MsgVal bit (CAN_IFn_ARB2[15])

should be cleared. Otherwise, the entire Message Object has to be initialized.

Access to the Bit Timing Register and to the Baud Rate Prescaler Extension Register for configuring bit timing is enabled when both the Init and CCE (CAN_CON[6]) bits are set.

Resetting the Init bit (by software only) finishes the software initialization. Later, the Bit Stream Processor (BSP) (see Section 6.5.7.15: Configuring the Bit Timing) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= Bus Idle) before it can take part in bus activities and start the message transfer.

The initialization of the Message Objects is independent of Init and can be done on the fly, but the Message Objects should all be configured to particular identifiers or set to not valid before the BSP starts the message transfer.

To change the configuration of a Message Object during normal operation, the software has to start by resetting the corresponding MsgVal bit. When the configuration is completed, MsgVal bit is set again.

6.20.5.2 CAN Message Transfer

Once the C_CAN is initialized and Init bit (CAN_CON[0]) is reset to zero, the C_CAN Core synchronizes itself to the CAN bus and starts the message transfer.

Received messages are stored in their appropriate Message Objects if they pass the Message Handler's acceptance filtering. The whole message including all arbitration bits, DLC (CAN_IFn_MCON[3:0]) and eight data bytes (CAN_IFn_DAT_A1/2; CAN_IFn_DAT_B1/2) are stored in the Message Object. If the Identifier Mask is used, the arbitration bits which are masked to "don't care" may be overwritten in the Message Object.

Software can read or write each message any time through the Interface Registers and the Message Handler guarantees data consistency in case of concurrent accesses.

Messages to be transmitted are updated by the application software. If a permanent Message Object (arbitration and control bits are set during configuration) exists for the message, only the data bytes are updated and the TxRqst bit (CAN_IFn_MCON[8]) with NewDat bit (CAN_IFn_MCON[15]) are set to start the transmission. If several transmit messages are assigned to the same Message Object (when the number of Message Objects is not sufficient), the whole Message Object has to be configured before the transmission of this message is requested.

The transmission of any number of Message Objects may be requested at the same time. Message objects are transmitted subsequently according to their internal priority. Messages may be updated or set to not valid any time, even when their requested transmission is still pending. The old data will be discarded when a message is updated before its pending transmission has started.

Depending on the configuration of the Message Object, the transmission of a message may be requested autonomously by the reception of a remote frame with a matching identifier.

6.20.5.3 Disabled Automatic Retransmission

In accordance with the CAN Specification (see ISO11898, 6.3.3 Recovery Management), the C_CAN provides means for automatic retransmission of frames that have lost arbitration or have been disturbed by errors during transmission. The frame transmission service will not be confirmed to the user before the transmission is successfully completed. This means that, by default, automatic retransmission is enabled. It can be disabled to enable the C_CAN to work within a Time Triggered CAN (TTCAN, see ISO11898-1) environment.

The Disabled Automatic Retransmission mode is enabled by setting the Disable Automatic Retransmission (DAR bit (CAN_CON[5])) to one. In this operation mode, the programmer has to consider the different behavior of bits TxRqst (CAN_IFn_MCON[8]) and NewDat (CAN_IFn_MCON[15]) of the Message Buffers:

- When a transmission starts, bit TxRqst of the respective Message Buffer is cleared, while bit

NewDat remains set.

- When the transmission completed successfully, bit NewDat is cleared.
- When a transmission fails (lost arbitration or error), bit NewDat remains set.
- To restart the transmission, the software should set the bit TxRqst again.

6.20.6 Test Mode

Test Mode is entered by setting the Test bit (CAN_CON[7]). In Test Mode, bits Tx1 (CAN_TEST[6]), Tx0 (CAN_TEST[5]), LBack (CAN_TEST[4]), Silent (CAN_TEST[3]) and Basic (CAN_TEST[2]) are writeable. Bit Rx (CAN_TEST[7]) monitors the state of the CAN_RX pin and therefore is only readable. All Test Register functions are disabled when the Test bit is cleared.

6.20.6.1 Silent Mode

The CAN Core can be set in Silent Mode by programming the Silent bit (CAN_TEST[3]) to one. In Silent Mode, the C_CAN is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the CAN Core is required to send a dominant bit (ACK bit, Error Frames), the bit is rerouted internally so that the CAN Core monitors this dominant bit, although the CAN bus may remain in recessive state. The Silent Mode can be used to analysis the traffic on a CAN bus without affecting it by the transmission of dominant bits. The Figure 6.20-2 shows the connection of signals CAN_TX and CAN_RX to the CAN Core in Silent Mode.

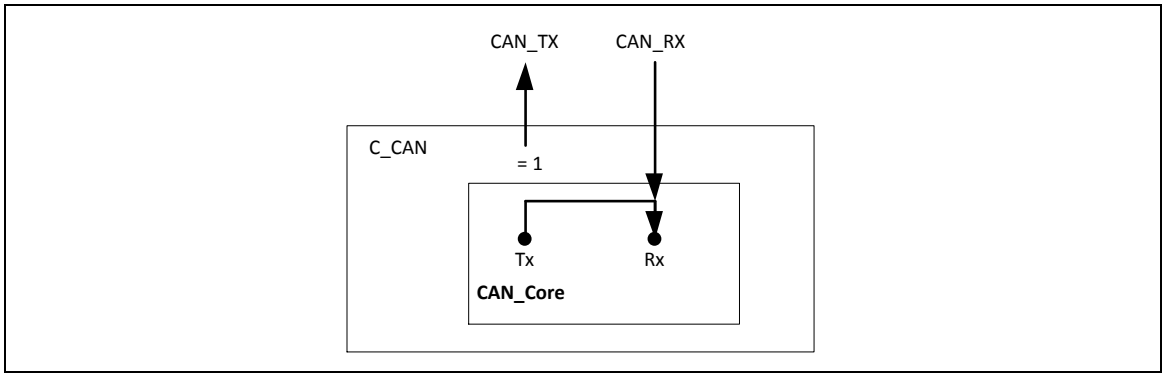


Figure 6.20-2 CAN Core in Silent Mode

6.20.6.2 Loop Back Mode

The CAN Core can be set in Loop Back Mode by programming the Test Register bit LBack (CAN_TEST[4]) to one. In Loop Back Mode, the CAN Core treats its own transmitted messages as received messages and stores them in a Receive Buffer (if they pass acceptance filtering). The Figure 6.20-3 shows the connection of signals, CAN_TX and CAN_RX, to the CAN Core in Loop Back Mode.

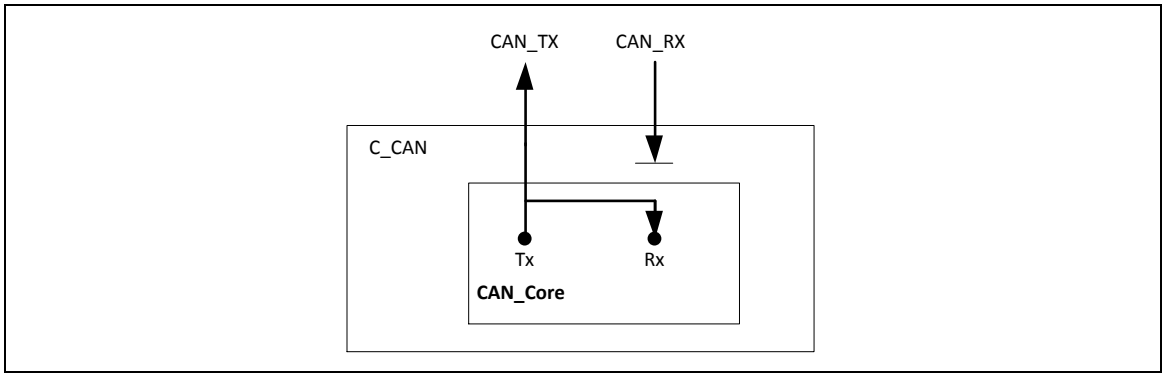


Figure 6.20-3 CAN Core in Loop Back Mode

This mode is provided for self-test functions. To be independent from external stimulation, the CAN Core ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/ remote frame) in Loop Back Mode. In this mode, the CAN Core performs an internal feedback from its Tx output to its Rx input. The actual value of the CAN_RX input pin is disregarded by the CAN Core. The transmitted messages can be monitored on the CAN_TX pin.

6.20.6.3 Loop Back Combined with Silent Mode

It is also possible to combine Loop Back Mode and Silent Mode by programming bits LBack (CAN_TEST[4]) and Silent (CAN_TEST[3]) to one at the same time. This mode can be used for a “Hot Selftest”, which means that C_CAN can be tested without affecting a running CAN system connected to the CAN_TX and CAN_RX pins. In this mode, the CAN_RX pin is disconnected from the CAN Core and the CAN_TX pin is held recessive. The Figure 6.20-4 shows the connection of signals CAN_TX and CAN_RX to the CAN Core in case of the combination of Loop Back Mode with Silent Mode.

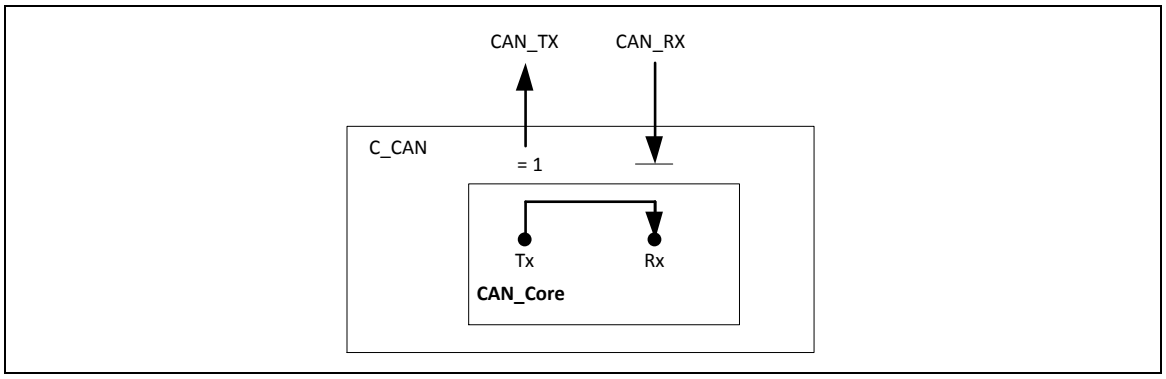


Figure 6.20-4 CAN Core in Loop Back Mode Combined with Silent Mode

6.20.6.4 Basic Mode

The CAN Core can be set in Basic Mode by programming the Basic bit (CAN_TEST[2]) to one. In this mode, the C_CAN runs without the Message RAM.

The IF1 Registers are used as Transmit Buffer. The transmission of the contents of the IF1 Registers is requested by writing the Busy bit (CAN_IFn_CREQ[15]) of the IF1 Command Request Register to one. The IF1 Registers are locked while the Busy bit is set. The Busy bit indicates that the transmission is pending.

As soon the CAN bus is idle, the IF1 Registers are loaded into the shift register of the CAN Core and the transmission is started. When the transmission has been completed, the Busy bit is reset and the locked IF1 Registers are released.

A pending transmission can be aborted at any time by resetting the Busy bit in the IF1 Command Request Register while the IF1 Registers are locked. If the software has reset the Busy bit, a possible retransmission in case of lost arbitration or in case of an error is disabled.

The IF2 Registers are used as a Receive Buffer. After the reception of a message the contents of the shift register is stored into the IF2 Registers, without any acceptance filtering.

Additionally, the actual contents of the shift register can be monitored during the message transfer. Each time a read Message Object is initiated by writing the Busy bit of the IF2 Command Request Register to one, the contents of the shift register are stored into the IF2 Registers.

In Basic Mode, the evaluation of all Message Object related control and status bits and the control bits of the IFn Command Mask Registers are turned off. The message number of the Command request registers is not evaluated. The NewDat (CAN_IFn_MCON[15]) and MsgLst (CAN_IFn_MCON[14]) bits retain their function, DLC3-0 indicates the received DLC (CAN_IFn_MCON[3:0]), and the other control bits are read as '0'.

6.20.6.5 Software Control of CAN_TX Pin

Four output functions are available for the CAN transmit pin, CAN_TX. In addition to its default function (serial data output), the CAN transmit pin can drive the CAN Sample Point signal to monitor CAN_Core's bit timing and it can drive constant dominant or recessive values. The latter two functions, combined with the readable CAN receive pin CAN_RX, can be used to check the physical layer of the CAN bus.

The output mode for the CAN_TX pin is selected by programming the Tx1 (CAN_TEST[6]) and Tx0 (CAN_TEST[5]) bits.

The three test functions of the CAN_TX pin interfere with all CAN protocol functions. CAN_TX must be left in its default function when CAN message transfer or any of the test modes (Loop Back Mode, Silent Mode or Basic Mode) are selected.

6.20.7 CAN Communications

6.20.7.1 Managing Message Objects

The configuration of the Message Objects in the Message RAM (with the exception of the bits MsgVal, NewDat, IntPnd and TxRqst) will not be affected by resetting the chip. All the Message Objects must be initialized by the application software or they must be "not valid" (MsgVal bit = '0') and the bit timing must be configured before the application software clears the Init bit (CAN_CON[0]).

The configuration of a Message Object is done by programming Mask, Arbitration, Control and Data fields of one of the two interface registers to the desired values. By writing to the corresponding IFn Command Request Register, the IFn Message Buffer Registers are loaded into the addressed Message Object in the Message RAM.

When the Init bit is cleared, the CAN Protocol Controller state machine of the CAN_Core and the state machine of the Message Handler control the internal data flow of the C_CAN. Received messages that pass the acceptance filtering are stored into the Message RAM, messages with pending transmission request are loaded into the CAN_Core's Shift Register and are transmitted through the CAN bus.

The application software reads received messages and updates messages to be transmitted through the IFn Interface Registers. Depending on the configuration, the application software is interrupted on certain CAN message and CAN error events.

6.20.7.2 Message Handler State Machine

The Message Handler controls the data transfer between the Rx/Tx Shift Register of the CAN Core,

the Message RAM and the IFn Registers.

The Message Handler FSM controls the following functions:

- Data Transfer from IFn Registers to the Message RAM
- Data Transfer from Message RAM to the IFn Registers
- Data Transfer from Shift Register to the Message RAM
- Data Transfer from Message RAM to Shift Register
- Data Transfer from Shift Register to the Acceptance Filtering unit
- Scanning of Message RAM for a matching Message Object
- Handling of TxRqst flags
- Handling of interrupts.

6.20.7.3 Data Transfer from/to Message RAM

When the application software initiates a data transfer between the IFn Registers and Message RAM, the Message Handler sets the Busy bit (CAN_IFn_CREQ[15]) to '1'. After the transfer has completed, the Busy bit is again cleared (see the Figure 6.20-5).

The respective Command Mask Register specifies whether a complete Message Object or only parts of it will be transferred. Due to the structure of the Message RAM, it is not possible to write single bits/bytes of one Message Object. It is always necessary to write a complete Message Object into the Message RAM. Therefore, the data transfer from the IFn Registers to the Message RAM requires a read-modify-write cycle. First, those parts of the Message Object that are not to be changed are read from the Message RAM and then the complete contents of the Message Buffer Registers are written into the Message Object.

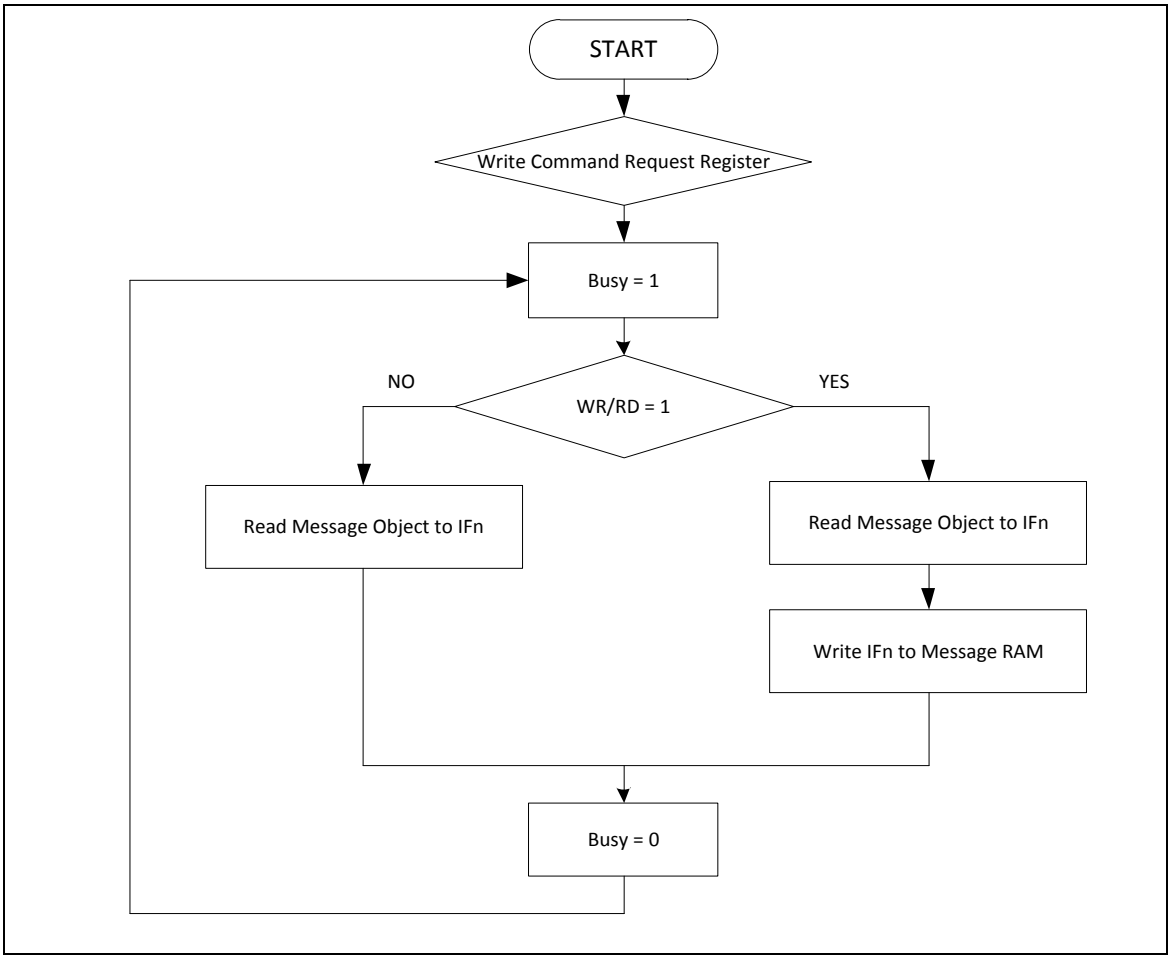


Figure 6.20-5 Data transfer between IFn Registers and Message

After a partial write of a Message Object, the Message Buffer Registers that are not selected in the Command Mask Register will set the actual contents of the selected Message Object.

After a partial read of a Message Object, the Message Buffer Registers that are not selected in the Command Mask Register will be left unchanged.

6.20.7.4 Message Transmission

If the shift register of the CAN Core cell is ready for loading and if there is no data transfer between the IFn Registers and Message RAM, the MsgVal bit (CAN_IFn_ARB2[15]) and TxRqst bits (CAN_TXREQ1/2) are evaluated. The valid Message Object with the highest priority pending transmission request is loaded into the shift register by the Message Handler and the transmission is started. The NewDat (CAN_IFn_MCON[15]) bit of the Message Object is reset.

After a successful transmission and also if no new data was written to the Message Object (NewDat = '0') since the start of the transmission, the TxRqst bit of the Message Control register (CAN_IFn_MCON[8]) will be reset. If TxIE bit (CAN_IFn_MCON[11]) is set, IntPnd bit (CAN_IFn_MCON[13]) of the Interrupt Identifier register will be set after a successful transmission. If the C_CAN has lost the arbitration or if an error occurred during the transmission, the message will be retransmitted as soon as the CAN bus is free again. Meanwhile, if the transmission of a message with higher priority has been requested, the messages will be transmitted in the order of their priority.

6.20.7.5 Acceptance Filtering of Received Messages

When the arbitration and control field (Identifier + IDE + RTR + DLC) of an incoming message is completely shifted into the Rx/Tx Shift Register of the CAN Core, the Message Handler FSM starts the scanning of the Message RAM for a matching valid Message Object.

To scan the Message RAM for a matching Message Object, the Acceptance Filtering unit is loaded with the arbitration bits from the CAN Core shift register. The arbitration and mask fields (including MsgVal (CAN_IFn_ARB2[15]), UMask (CAN_IFn_MCON[12]), NewDat (CAN_IFn_MCON[15]) and EoB (CAN_IFn_MCON[7])) of Message Object 1 are then loaded into the Acceptance Filtering unit and compared with the arbitration field from the shift register. This is repeated with each following Message Object until a matching Message Object is found or until the end of the Message RAM is reached.

If a match occurs, the scan is stopped and the Message Handler FSM proceeds depending on the type of frame (Data Frame or Remote Frame) received.

Reception of Data Frame

The Message Handler FSM stores the message from the CAN Core shift register into the respective Message Object in the Message RAM. Not only the data bytes, but all arbitration bits and the Data Length Code are stored into the corresponding Message Object. This is done to keep the data bytes connected with the identifier even if arbitration mask registers are used.

The NewDat bit (CAN_IFn_MCON[15]) is set to indicate that new data (not yet seen by the software) has been received. The application software should reset NewDat bit when the Message Object has been read. If at the time of reception, the NewDat bit was already set, MsgLst (CAN_IFn_MCON[14]) is set to indicate that the previous data (supposedly not seen by the software) is lost. If the RxIE bit (CAN_IFn_MCON[10]) is set, the IntPnd bit (CAN_IFn_MCON[13]) is set, causing the Interrupt Register to point to this Message Object.

The TxRqst bit (CAN_IFn_MCON[8]) of this Message Object is reset to prevent the transmission of a Remote Frame, while the requested Data Frame has just been received.

Reception of Remote Frame

When a Remote Frame is received, three different configurations of the matching Message Object have to be considered:

1) Dir (CAN_IFn_ARB2[13]) = ‘1’ (direction = transmit), RmtEn (CAN_IFn_MCON[9]) = ‘1’ and UMask (CAN_IFn_MCON[12]) = ‘1’ or ’0’

At the reception of a matching Remote Frame, the TxRqst bit of this Message Object is set. The rest of the Message Object remains unchanged.

2) Dir = ‘1’ (direction = transmit), RmtEn = ‘0’ and UMask = ’0’

At the reception of a matching Remote Frame, the TxRqst bit of this Message Object remains unchanged; the Remote Frame is ignored.

3) Dir = ‘1’ (direction = transmit), RmtEn = ‘0’ and UMask = ’1’

At the reception of a matching Remote Frame, the TxRqst bit of this Message Object is reset. The arbitration and control field (Identifier + IDE + RTR + DLC) from the shift register is stored in the Message Object of the Message RAM and the NewDat bit (CAN_IFn_MCON[15]) of this Message Object is set. The data field of the Message Object remains unchanged; the Remote Frame is treated similar to a received Data Frame.

6.20.7.6 Receive/Transmit Priority

The receive/transmit priority for the Message Objects is attached to the message number. Message Object 1 has the highest priority, while Message Object 32 has the lowest priority. If more than one transmission request is pending, they are serviced due to the priority of the corresponding Message Object

6.20.7.7 Configuring a Transmit Object

The Table 6-35 shows how a Transmit Object should be initialized.

Ms	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	1	0	0	0	appl.	0	appl.	0

Table 6-35 Initialization of a Transmit Object

Note: appl. = application software.

The Arbitration Register values (ID28-0 (CAN_IFn_ARB1/2) and Xtd bit (CAN_IFn_ARB2[14])) are provided by the application. They define the identifier and type of the outgoing message. If an 11-bit Identifier (“Standard Frame”) is used, it is programmed to ID28 - ID18. The ID17 - ID0 can then be disregarded.

If the TxIE bit (CAN_IFn_MCON[11]) is set, the IntPnd bit (CAN_IFn_MCON[13]) will be set after a successful transmission of the Message Object.

If the RmtEn bit (CAN_IFn_MCON[9]) is set, a matching received Remote Frame will cause the TxRqst bit (CAN_IFn_MCON[8]) to be set; the Remote Frame will autonomously be answered by a Data Frame.

The Data Register values (DLC3-0 (CAN_IFn_MCON[3:0]), Data(0)-(7)) are provided by the application, TxRqst and RmtEn may not be set before the data is valid.

The Mask Registers (Msk28-0, UMask, MXtd and MDir bits) may be used (UMask (CAN_IFn_MCON[12]) = ’1’) to allow groups of Remote Frames with similar identifiers to set the TxRqst bit. The Dir bit (CAN_IFn_ARB2[13]) should not be masked.

6.20.7.8 Updating a Transmit Object

The software may update the data bytes of a Transmit Object any time through the IFn Interface registers, neither MsgVal bit (CAN_IFn_ARB2[15]) nor TxRqst (CAN_IFn_MCON[8]) have to be reset before the update.

Even if only a part of the data bytes are to be updated, all four bytes of the corresponding IFn Data A Register or IFn Data B Register have to be valid before the contents of that register are transferred to the Message Object. Either the application software has to write all four bytes into the IFn Data Register or the Message Object is transferred to the IFn Data Register before the software writes the new data bytes.

When only the (eight) data bytes are updated, first 0x0087 is written to the Command Mask Register and then the number of the Message Object is written to the Command Request Register, concurrently updating the data bytes and setting TxRqst.

To prevent the reset of TxRqst at the end of a transmission that may already be in progress while the data is updated, NewDat (CAN_IFn_MCON[15]) has to be set together with TxRqst.

When NewDat is set together with TxRqst, NewDat will be reset as soon as the new transmission has

started.

6.20.7.9 Configuring a Receive Object

The Table 6-36 shows how a Receive Object should be initialized.

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	appl.	appl.	1	0	0	0	appl.	0	0	0	0

Table 6-36 Initialization of a Receive Object

The Arbitration Registers values (ID28-0 (CAN_IFn_ARB1/2) and Xtd bit (CAN_IFn_ARB2[14])) are provided by the application. They define the identifier and type of accepted received messages. If an 11-bit Identifier (“Standard Frame”) is used, it is programmed to ID28 - ID18. Then ID17 - ID0 can be disregarded. When a Data Frame with an 11-bit Identifier is received, ID17 - ID0 will be set to ‘0’.

If the RxIE bit (CAN_IFn_MCON[10]) is set, the IntPnd bit (CAN_IFn_MCON[13]) will be set when a received Data Frame is accepted and stored in the Message Object.

The Data Length Code (DLC3-0 (CAN_IFn_MCON[3:0])) is provided by the application. When the Message Handler stores a Data Frame in the Message Object, it will store the received Data Length Code and eight data bytes. If the Data Length Code is less than 8, the remaining bytes of the Message Object will be overwritten by unspecified values.

The Mask Registers (Msk28-0, UMask, MXtd and MDir bits) may be used (UMask (CAN_IFn_MCON[12]) = ’1’) to allow groups of Data Frames with similar identifiers to be accepted. The Dir bit (CAN_IFn_ARB2[13]) should not be masked in typical applications.

6.20.7.10 Handling Received Messages

The application software may read a received message any time through the IFn Interface registers. The data consistency is guaranteed by the Message Handler state machine.

Typically, the software will write first 0x007F to the Command Mask Register and then the number of the Message Object to the Command Request Register. This combination will transfer the whole received message from the Message RAM into the Message Buffer Register. Additionally, the bits NewDat (CAN_IFn_MCON[15]) and IntPnd (CAN_IFn_MCON[13]) are cleared in the Message RAM (not in the Message Buffer).

If the Message Object uses masks for acceptance filtering, the arbitration bits show which of the matching messages have been received.

The actual value of NewDat shows whether a new message has been received since the last time this Message Object was read. The actual value of MsgLst (CAN_IFn_MCON[14]) shows whether more than one message has been received since the last time this Message Object was read. MsgLst will not be automatically reset.

By means of a Remote Frame, the software may request another CAN node to provide new data for a receive object. Setting the TxRqst bit (CAN_IFn_MCON[8]) of a receive object will cause the transmission of a Remote Frame with the receive object’s identifier. This Remote Frame triggers the other CAN node to start the transmission of the matching Data Frame. If the matching Data Frame is received before the Remote Frame could be transmitted, the TxRqst bit is automatically reset.

6.20.7.11 Configuring a FIFO Buffer

With the exception of the EoB bit (CAN_IFn_MCON[7]), the configuration of Receive Objects belonging to a FIFO Buffer is the same as the configuration of a (single) Receive Object, see Section 6.5.7.9: Configuring a Receive Object.

To concatenate two or more Message Objects into a FIFO Buffer, the identifiers and masks (if used) of these Message Objects have to be programmed to matching values. Due to the implicit priority of the Message Objects, the Message Object with the lowest number will be the first Message Object of the FIFO Buffer. The EoB bit of all Message Objects of a FIFO Buffer except the last have to be programmed to zero. The EoB bit of the last Message Object of a FIFO Buffer is set to one, configuring it as the End of the Block.

6.20.7.12 Receiving Messages with FIFO Buffers

Received messages with identifiers matching to a FIFO Buffer are stored into a Message Object of this FIFO Buffer starting with the Message Object with the lowest message number.

When a message is stored into a Message Object of a FIFO Buffer, the NewDat bit (CAN_IFn_MCON[15]) of this Message Object is set. By setting NewDat while EoB (CAN_IFn_MCON[7]) is zero, the Message Object is locked for further write access by the Message Handler until the application software has written the NewDat bit back to zero.

Messages are stored into a FIFO Buffer until the last Message Object of this FIFO Buffer is reached. If none of the preceding Message Objects is released by writing NewDat to zero, all further messages for this FIFO Buffer will be written into the last Message Object of the FIFO Buffer and therefore overwrite the previous messages.

6.20.7.13 Reading from a FIFO Buffer

When the application software transfers the contents of a Message Object to the IFn Message Buffer register by writing its number to the IFn Command Request Register, the corresponding Command Mask Register should be programmed in such a way that bits NewDat (CAN_IFn_MCON[15]) and IntPnd (CAN_IFn_MCON[13]) are reset to zero (TxRqst/NewDat (CAN_IFn_CMASK[2]) = ‘1’ and ClrIntPnd (CAN_IFn_CMASK[3]) = ‘1’). The values of these bits in the Message Control Register always reflect the status before resetting the bits.

To assure the correct function of a FIFO Buffer, the application software should read the Message Objects starting at the FIFO Object with the lowest message number.

The Figure 6.20-6 shows how a set of Message Objects which are concatenated to a FIFO Buffer can be handled by the application software.

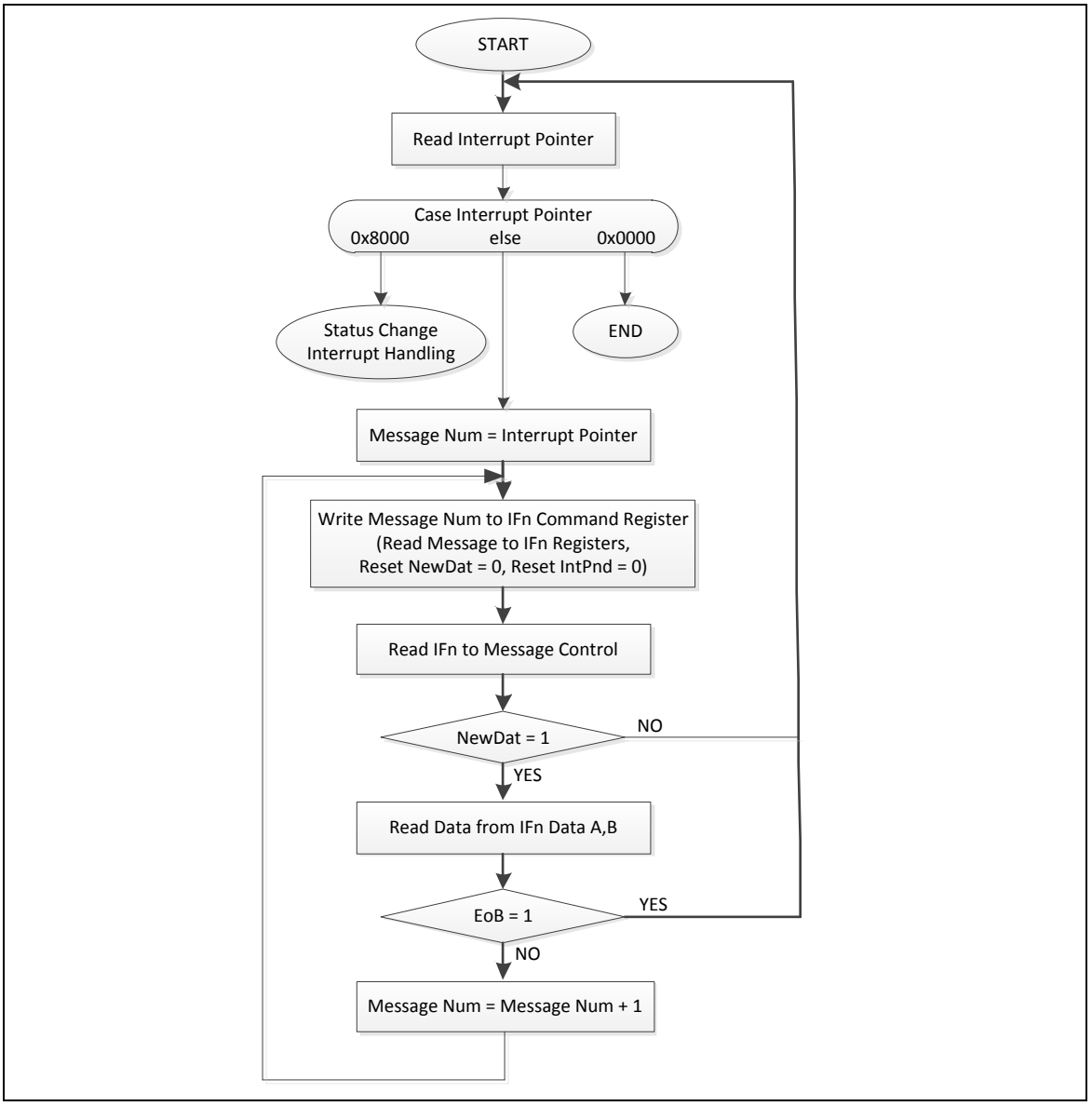


Figure 6.20-6 Application Software Handling of a FIFO Buffer

6.20.7.14 Handling Interrupts

If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the application software has cleared it.

The Status Interrupt has the highest priority. Among the message interrupts, interrupt priority of the Message Object decreases with increasing message number.

A message interrupt is cleared by clearing the IntPnd bit (CAN_IFn_MCON[13]) of the Message Object. The Status Interrupt is cleared by reading the Status Register.

The interrupt identifier, IntId, in the Interrupt Register, indicates the cause of the interrupt. When no interrupt is pending, the register will hold the value zero. If the value of the Interrupt Register is different from zero, then there is an interrupt pending and, if IE (CAN_CON[1]) is set, the CAN_INT interrupt signal is active. The interrupt remains active until the Interrupt Register is back to value zero (the cause of the interrupt is reset) or until IE is reset.

The value 0x8000 indicates that an interrupt is pending because the CAN Core has updated (not necessarily changed) the Status Register (Error Interrupt or Status Interrupt). This interrupt has the highest priority. The application software can update (reset) the status bits RxOk (CAN_STATUS[4]), TxOk (CAN_STATUS[3]) and LEC (CAN_STATUS[2:0]), but a write access of the software to the Status Register can never generate or reset an interrupt.

All other values indicate that the source of the interrupt is one of the Message Objects. IntId points to the pending message interrupt with the highest interrupt priority.

The application software controls whether a change of the Status Register may cause an interrupt (bits EIE (CAN_CON[3]) and SIE (CAN_CON[2])) and whether the interrupt line becomes active when the Interrupt Register is different from zero (bit IE in the CAN Control Register). The Interrupt Register will be updated even when IE is reset.

The application software has two possibilities to follow the source of a message interrupt. First, it can follow the IntId in the Interrupt Register and second it can poll the Interrupt Pending Register.

An interrupt service routine that is reading the message that is the source of the interrupt may read the message and reset the Message Object's IntPnd at the same time (bit ClrIntPnd (CAN_IFn_CMASK[3])). When IntPnd is cleared, the Interrupt Register will point to the next Message Object with a pending interrupt.

6.20.7.15 Configuring the Bit Timing

Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly.

In many cases, the CAN bit synchronization will amend a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. However, in the case of arbitration, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive.

The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and interaction of the CAN nodes on the CAN bus.

6.20.7.16 Bit Time and Bit Rate

CAN supports bit rates in the range of lower than 1 Kbit/s up to 1000 Kbit/s. Each member of the CAN network has its own clock generator, usually a quartz oscillator. The timing parameter of the bit time (i.e. the reciprocal of the bit rate) can be configured individually for each CAN node, creating a common bit rate even though the oscillator periods of the CAN nodes (f_{osc}) may be different.

The frequencies of these oscillators are not absolutely stable, small variations are caused by changes in temperature or voltage and by deteriorating components. As long as the variations remain inside a specific oscillator tolerance range (d_f), the CAN nodes are able to compensate for the different bit rates by re-synchronizing to the bit stream.

According to the CAN specification, the bit time is divided into four segments (see theFigure 6.20-7). The Synchronization Segment, the Propagation Time Segment, the Phase Buffer Segment 1 and the Phase Buffer Segment 2. Each segment consists of a specific, programmable number of time quanta (see Table 6-37). The length of the time quantum (t_q), which is the basic time unit of the bit time, is defined by the CAN controller's APB clock f_{APB} and the BRP bit (CAN_BTIME[5:0]) : t_q = BRP / f_{APB}.

The Synchronization Segment, Sync_Seg, is that part of the bit time where edges of the CAN bus level are expected to occur. The distance between an edge that occurs outside of Sync_Seg, and the Sync_Seg is called the phase error of that edge. The Propagation Time Segment, Prop_Seg, is intended to compensate for the physical delay time within the CAN network. The Phase Buffer Segments Phase_Seg1 and Phase_Seg2 surround the Sample Point. The (Re-)Synchronization Jump Width (SJW) defines how far a re-synchronization may move the Sample Point inside the limits defined by the Phase Buffer Segments to compensate for edge phase errors.

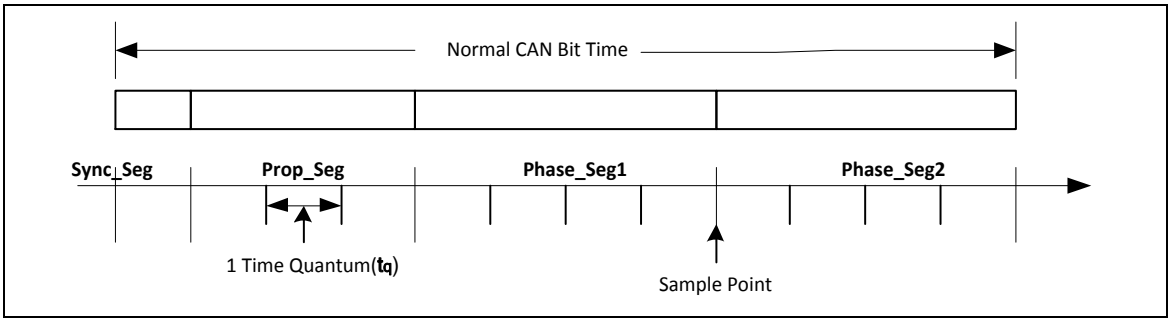


Figure 6.20-7 Bit Timing

Parameter	Range	Remark
BRP	[1..32]	Defines the length of the time quantum t_q
Sync_Seg	1 t_q	Fixed length, synchronization of bus input to APB clock
Prop_Seg	[1..8] t_q	Compensates for the physical delay time
Phase_Seg1	[1..8] t_q	Which may be lengthened temporarily by synchronization
Phase_Seg2	[1..8] t_q	Which may be shortened temporarily by synchronization
SJW	[1..4] t_q	Which may not be longer than either Phase Buffer Segment
This table describes the minimum programmable ranges required by the CAN protocol		

Table 6-37 CAN Bit Time Parameters

A given bit rate may be met by different bit time configurations, but for the proper function of the CAN network the physical delay time and the oscillator's tolerance range have to be considered.

6.20.7.17 Propagation Time Segment

This part of the bit time is used to compensate physical delay time within the network. These delay time consist of the signal propagation time on the bus and the internal delay time of the CAN nodes.

Any CAN node synchronized to the bit stream on the CAN bus will be out of phase with the transmitter of that bit stream, caused by the signal propagation time between the two nodes. The CAN protocol's non-destructive bitwise arbitration and the dominant acknowledge bit provided by receivers of CAN messages requires that a CAN node transmitting a bit stream must also be able to receive dominant bits transmitted by other CAN nodes that are synchronized to that bit stream. The example in the Figure 6.20-8 shows the phase shift and propagation time between two CAN nodes.

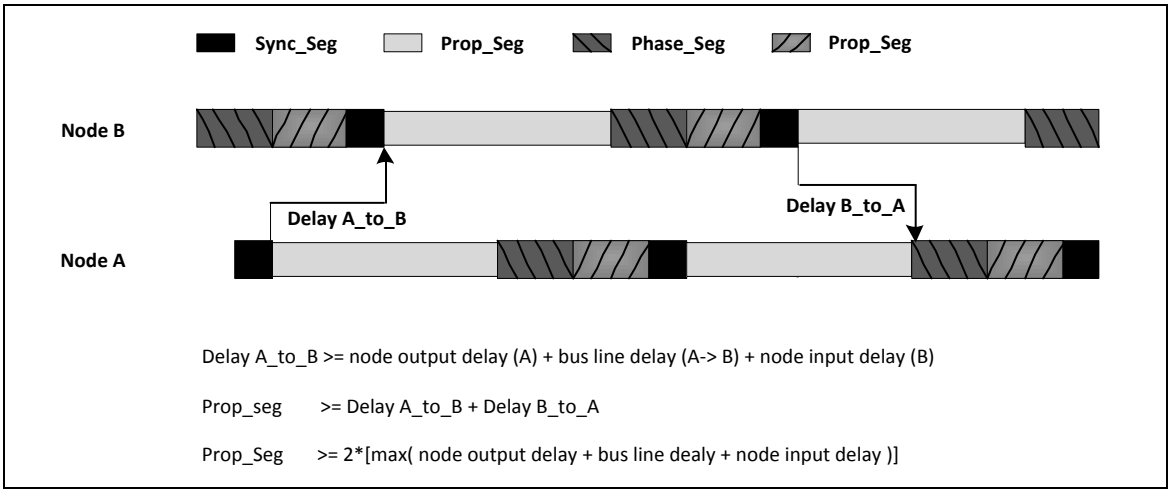


Figure 6.20-8 Propagation Time Segment

In this example, both nodes A and B are transmitters, performing an arbitration for the CAN bus. Node A has sent its Start of Frame bit less than one bit time earlier than node B, therefore node B has synchronized itself to the received edge from recessive to dominant. Since node B has received this edge delay (A_to_B) after it has been transmitted, B's bit timing segments are shifted with respect to A. Node B sends an identifier with higher priority and so it will win the arbitration at a specific identifier bit when it transmits a dominant bit while node A transmits a recessive bit. The dominant bit transmitted by node B will arrive at node A after the delay (B_to_A).

Due to oscillator tolerances, the actual position of node A's Sample Point can be anywhere inside the nominal range of node A's Phase Buffer Segments, so the bit transmitted by node B must arrive at node A before the start of Phase_Seg1. This condition defines the length of Prop_Seg.

If the edge from recessive to dominant transmitted by node B arrives at node A after the start of Phase_Seg1, it can happen that node A samples a recessive bit instead of a dominant bit, resulting in a bit error and the destruction of the current frame by an error flag.

The error occurs only when two nodes arbitrate for the CAN bus that have oscillators of opposite ends of the tolerance range and that are separated by a long bus line. This is an example of a minor error in the bit timing configuration (Prop_Seg is too short) that causes sporadic bus errors.

Some CAN implementations provide an optional 3 Sample Mode but the C_CAN does not. In this mode, the CAN bus input signal passes a digital low-pass filter, using three samples and a majority logic to determine the valid bit value. This results in an additional input delay of 1 t_q, requiring a longer Prop_Seg.

6.20.7.18 Phase Buffer Segments and Synchronization

The Phase Buffer Segments (Phase_Seg1 and Phase_Seg2) and the Synchronization Jump Width (SJW) are used to compensate for the oscillator tolerance. The Phase Buffer Segments may be lengthened or shortened by synchronization.

Synchronizations occur on edges from recessive to dominant, their purpose is to control the distance between edges and Sample Points.

Edges are detected by sampling the actual bus level in each time quantum and comparing it with the bus level at the previous Sample Point. A synchronization may be done only if a recessive bit was sampled at the previous Sample Point and if the bus level at the actual time quantum is dominant.

An edge is synchronous if it occurs inside of Sync_Seg, otherwise the distance between edge and the end of Sync_Seg is the edge phase error, measured in time quanta. If the edge occurs before

Sync_Seg, the phase error is negative, else it is positive.

Two types of synchronization exist, Hard Synchronization and Re-synchronization.

A Hard Synchronization is done once at the start of a frame and inside a frame only when Re-synchronizations occur.

- Hard Synchronization

After a hard synchronization, the bit time is restarted with the end of Sync_Seg, regardless of the edge phase error. Thus hard synchronization forces the edge, which has caused the hard synchronization to lie within the synchronization segment of the restarted bit time.

- Bit Re-synchronization

Re-synchronization leads to a shortening or lengthening of the bit time such that the position of the sample point is shifted with regard to the edge.

When the phase error of the edge which causes Re-synchronization is positive, Phase_Seg1 is lengthened. If the magnitude of the phase error is less than SJW, Phase_Seg1 is lengthened by the magnitude of the phase error, else it is lengthened by SJW.

When the phase error of the edge, which causes Re-synchronization is negative, Phase_Seg2 is shortened. If the magnitude of the phase error is less than SJW, Phase_Seg2 is shortened by the magnitude of the phase error, else it is shortened by SJW.

When the magnitude of the phase error of the edge is less than or equal to the programmed value of SJW, the results of Hard Synchronization and Re-synchronization are the same. If the magnitude of the phase error is larger than SJW, the Re-synchronization cannot compensate the phase error completely, an error (phase error - SJW) remains.

Only one synchronization may be done between two Sample Points. The Synchronizations maintain a minimum distance between edges and Sample Points, giving the bus level time to stabilize and filtering out spikes that are shorter than (Prop_Seg + Phase_Seg1).

Apart from noise spikes, most synchronizations are caused by arbitration. All nodes synchronize “hard” on the edge transmitted by the “leading” transceiver that started transmitting first, but due to propagation delay time, they cannot become ideally synchronized. The “leading” transmitter does not necessarily win the arbitration, therefore the receivers have to synchronize themselves to different transmitters that subsequently “take the lead” and that are differently synchronized to the previously “leading” transmitter. The same happens at the acknowledge field, where the transmitter and some of the receivers will have to synchronize to that receiver that “takes the lead” in the transmission of the dominant acknowledge bit.

Synchronizations after the end of the arbitration will be caused by oscillator tolerance, when the differences in the oscillator’s clock periods of transmitter and receivers sum up during the time between synchronizations (at most ten bits). These summarized differences may not be longer than the SJW, limiting the oscillator’s tolerance range.

The examples in the Figure 6.20-9 shows how the Phase Buffer Segments are used to compensate for phase errors. There are three drawings of each two consecutive bit timings. The upper drawing shows the synchronization on a “late” edge, the lower drawing shows the synchronization on an “early” edge, and the middle drawing is the reference without synchronization.

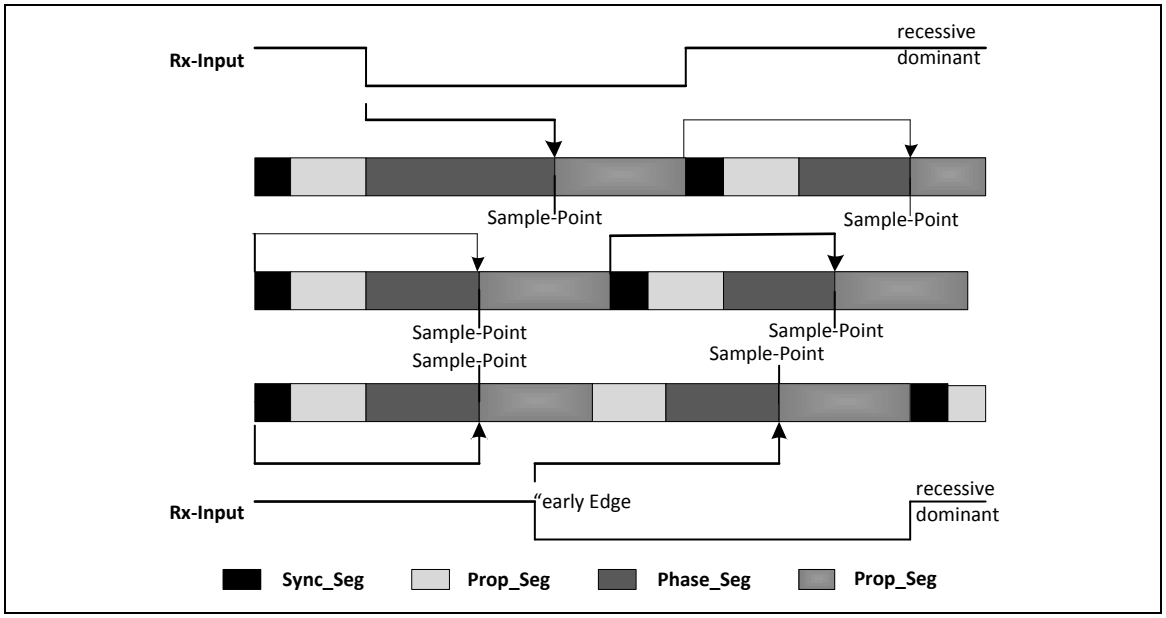


Figure 6.20-9 Synchronization on "late" and "early" Edges

In the first example an edge from recessive to dominant occurs at the end of Prop_Seg. The edge is "late" since it occurs after the Sync_Seg. Reacting to the "late" edge, Phase_Seg1 is lengthened so that the distance from the edge to the Sample Point is the same as it would have been from the Sync_Seg to the Sample Point if no edge had occurred. The phase error of this "late" edge is less than SJW, so it is fully compensated and the edge from dominant to recessive at the end of the bit, which is one nominal bit time long, occurs in the Sync_Seg.

In the second example an edge from recessive to dominant occurs during Phase_Seg2. The edge is "early" since it occurs before a Sync_Seg. Reacting to the "early" edge, Phase_Seg2 is shortened and Sync_Seg is omitted, so that the distance from the edge to the Sample Point is the same as it would have been from an Sync_Seg to the Sample Point if no edge had occurred. As in the previous example, the magnitude of this "early" edge's phase error is less than SJW, so it is fully compensated.

The Phase Buffer Segments are lengthened or shortened temporarily only; at the next bit time, the segments return to their nominal programmed values.

In these examples, the bit timing is seen from the point of view of the CAN implementation's state machine, where the bit time starts and ends at the Sample Points. The state machine omits Sync_Seg when synchronising on an "early" edge because it cannot subsequently redefine that time quantum of Phase_Seg2 where the edge occurs to be the Sync_Seg.

The examples in the Figure 6.20-10 shows how short dominant noise spikes are filtered by synchronisations. In both examples the spike starts at the end of Prop_Seg and has the length of (Prop_Seg + Phase_Seg1).

In the first example, the Synchronization Jump Width is greater than or equal to the phase error of the spike's edge from recessive to dominant. Therefore the Sample Point is shifted after the end of the spike; a recessive bus level is sampled.

In the second example, SJW is shorter than the phase error, so the Sample Point cannot be shifted far enough; the dominant spike is sampled as actual bus level.

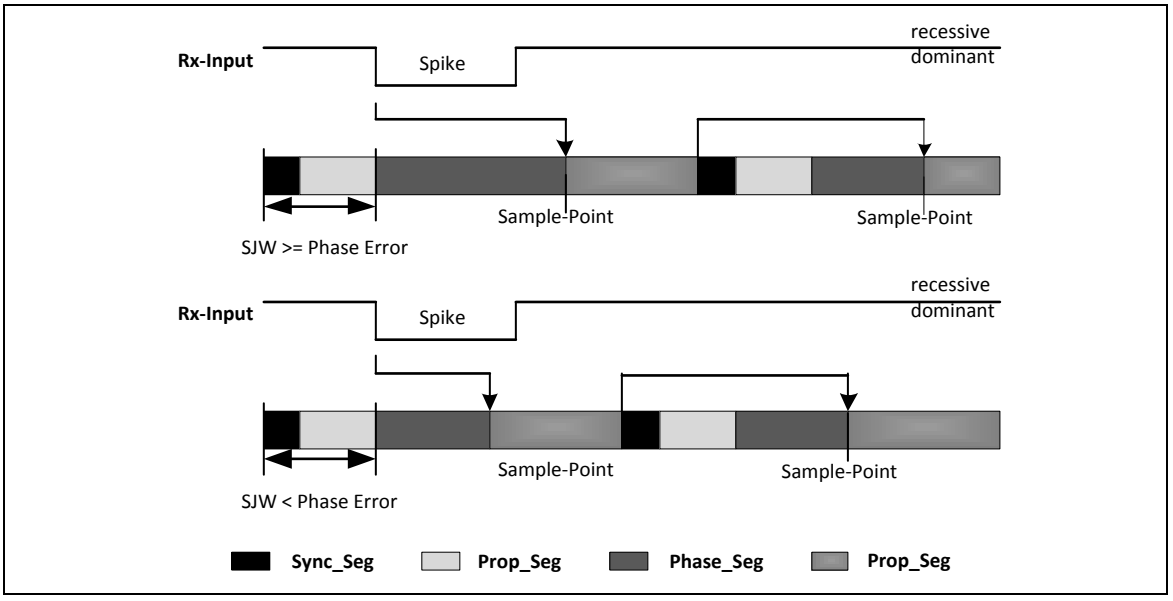


Figure 6.20-10 Filtering of Short Dominant Spikes

6.20.7.19 Oscillator Tolerance Range

The oscillator tolerance range was increased when the CAN protocol was developed from version 1.1 to version 1.2 (version 1.0 was never implemented in silicon). The option to synchronize on edges from dominant to recessive became obsolete, only edges from recessive to dominant are considered for synchronization. The protocol update to version 2.0 (A and B) had no influence on the oscillator tolerance.

The tolerance range df for an oscillator frequency f_{osc} around the nominal frequency f_{nom} is:

$$(1 - d_f) \cdot f_{nom} <= f_{osc} <= (1 + d_f) \cdot f_{nom}$$

It depends on the proportions of Phase_Seg1, Phase_Seg2, SJW and the bit time. The maximum tolerance df is the defined by two conditions (both shall be met):

$$\text{I: } d_f <= \frac{\min(\text{Phase\_Seg1}, \text{Phase\_Seg2})}{2 * (13 * \text{bit\_time} - \text{Phase\_Seg2})}$$

$$\text{II: } d_f <= \frac{\text{SJW}}{20 * \text{bit\_time}}$$

Note: These conditions base on the APB cock = f_{osc}.

It has to be considered that SJW may not be larger than the smaller of the Phase Buffer Segments and that the Propagation Time Segment limits that part of the bit time that may be used for the Phase Buffer Segments.

The combination Prop_Seg = 1 and Phase_Seg1 = Phase_Seg2 = SJW = 4 allows the largest possible oscillator tolerance of 1.58%. This combination with a Propagation Time Segment of only

10% of the bit time is not suitable for short bit times; it can be used for bit rates of up to 125 Kbit/s (bit time = 8us) with a bus length of 40 m.

6.20.7.20 Configuring the CAN Protocol Controller

In most CAN implementations and also in the C_CAN, the bit timing configuration is programmed in two register bytes. The sum of Prop_Seg and Phase_Seg1 (as TSEG1 (CAN_BTIME[11:8])) is combined with Phase_Seg2 (as TSEG2 (CAN_BTIME[14:12])) in one byte, SJW (CAN_BTIME[7:6]) and BRP (CAN_BTIME[5:0]) are combined in the other byte.

In these bit timing registers, the four components TSEG1, TSEG2, SJW and BRP have to be programmed to a numerical value that is one less than its functional value. Therefore, instead of values in the range of [1..n], values in the range of [0..n-1] are programmed. That way, e.g. SJW (functional range of [1..4]) is represented by only two bits.

Therefore the length of the bit time is (programmed values) [TSEG1 + TSEG2 + 3] t_q or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] t_q.

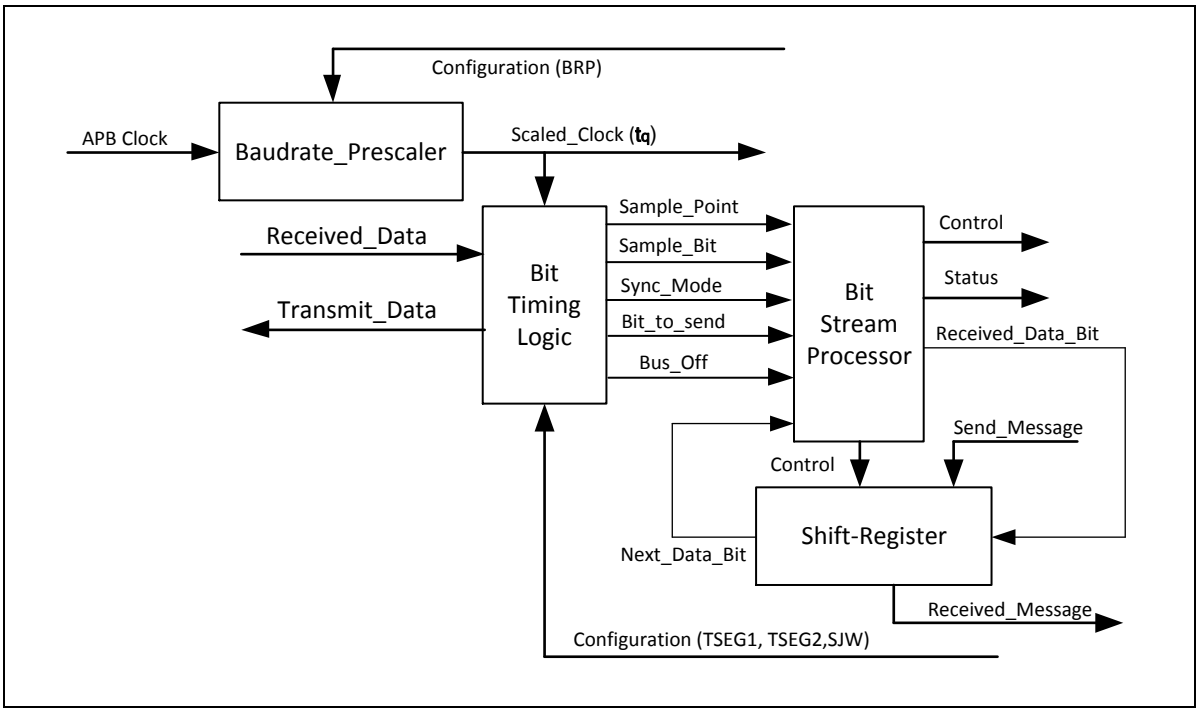


Figure 6.20-11 Structure of the CAN Core's CAN Protocol Controller

The data in the bit timing registers is the configuration input of the CAN protocol controller. The Baud Rate Prescaler (configured by BRP) defines the length of the time quantum, the basic time unit of the bit time; the Bit Timing Logic (configured by TSEG1, TSEG2 and SJW) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the Sample Point, and occasional synchronizations are controlled by the BTL (Bit Timing Logic) state machine, which is evaluated once each time quantum. The rest of the CAN protocol controller, the BSP (Bit Stream Processor) state machine is evaluated once each bit time, at the Sample Point.

The Shift Register sends the messages serially and parallelizes received messages. It's loading and shifting is controlled by the BSP.

The BSP translates messages into frames and vice versa. It generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. It is evaluated at the Sample Point and processes the sampled bus input bit. The time that is needed to calculate the next bit to be sent after the Sample point (e.g. data bit, CRC (Cyclic Redundancy Check) bit, stuff bit, error flag or idle) is called the Information Processing Time (IPT).

The IPT is application specific but may not be longer than 2 t_q; the IPT for the C_CAN is 0 t_q. Its length is the lower limit of the programmed length of Phase_Seg2. In case of a synchronization, Phase_Seg2 may be shortened to a value less than IPT, which does not affect bus timing.

6.20.7.21 Calculating Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a desired bit rate or bit time. The resulting bit time (1/bit rate) must be an integer multiple of the APB clock period.

The bit time may consist of 4 to 25 time quanta, the length of the time quantum t_q is defined by the Baud Rate Prescaler with t_q = (Baud Rate Prescaler)/f_{apb_clk}. Several combinations may lead to the desired bit time, allowing iterations of the following steps.

First part of the bit time to be defined is the Prop_Seg. Its length depends on the delay time measured in the APB clock. A maximum bus length as well as a maximum node delay has to be defined for expandible CAN bus systems. The resulting time for Prop_Seg is converted into time quanta (rounded up to the nearest integer multiple of t_q).

The Sync_Seg is 1 t_q long (fixed), leaving (bit time – Prop_Seg – 1) t_q for the two Phase Buffer Segments. If the number of remaining t_q is even, the Phase Buffer Segments have the same length, Phase_Seg2 = Phase_Seg1, else Phase_Seg2 = Phase_Seg1 + 1.

The minimum nominal length of Phase_Seg2 has to be regarded as well. Phase_Seg2 may not be shorter than the IPT of the CAN controller, which, depending on the actual implementation, is in the range of [0..2] t_q.

The length of the Synchronization Jump Width is set to its maximum value, which is the minimum of 4 and Phase_Seg1.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formulas given in Section “Oscillator Tolerance Range”.

If more than one configuration is possible, that configuration allowing the highest oscillator tolerance range should be chosen.

CAN nodes with different system clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay time, is done once for the whole network.

The oscillator tolerance range of the CAN systems is limited by that node with the lowest tolerance range.

The calculation may shows that bus length or bit rate have to be decreased or that the stability of the oscillator frequency has to be increased in order to find a protocol compliant configuration of the CAN bit timing. The resulting configuration is written into the Bit Timing Register: (Phase_Seg2-1) & (Phase_Seg1+Prop_Seg-1) & (SynchronisationJumpWidth-1) & (Prescaler-1)

Example for Bit Timing at High Baud Rate

In this example, the frequency of APB_CLK is 10 MHz, BRP (CAN_BTIME[5:0]) is 0, and the bit rate is 1 MBit/s.

t_q	100	ns	= t_{APB_CLK}
delay of bus driver	50	ns	
delay of receiver circuit	30	ns	
delay of bus line (40m)	220	ns	
t_{Prop}	600	ns	= 6 • t_q
t_{SJW}	100	ns	= 1 • t_q
t_{TSeg1}	700	ns	= t_{Prop} + t_{SJW}
t_{TSeg2}	200	ns	= Information Processing Time + 1 • t_q
$t_{Sync-Seg}$	100	ns	= 1 • t_q
bit time	1000	ns	= $t_{Sync-Seg}$ + t_{TSeg1} + t_{TSeg2}
tolerance for APB_CLK	0.39	%	$= \frac{Min(PB1, PB2)}{2 \times 13 \times (bit\ \ time - PB2))}$

$$= \frac{0.1\mu s}{2x(13x(1\mu s - 0.2\mu s))}$$

In this example, the concatenated bit time parameters are $(2\text{-}1)_3$ & $(7\text{-}1)_4$ & $(1\text{-}1)_2$ & $(1\text{-}1)_6$, and the Bit Timing Register is programmed to 0x1600.

Example for Bit Timing at Low Baud Rate

In this example, the frequency of APB_CLK is 2 MHz, BRP (CAN_BTIME[5:0]) is 1, and the bit rate is 100 Kbit/s.

t_q 1 us = 2 • t_{APB_CLK}

delay of bus driver	200	ns	
delay of receiver circuit	80	ns	
delay of bus line (40m)	220	ns	
t_{Prop}	1	us	= 1 • t_q
t_{SJW}	4	us	= 4 • t_q
t_{TSeg1}	5	us	= t_{Prop} + t_{SJW}
t_{TSeg2}	4	us	= Information Processing Time + 3 • t_q
$t_{Sync-Seg}$	1	us	= 1 • t_q
bit time	10	us	= $t_{Sync-Seg}$ + t_{TSeg1} + t_{TSeg2}
tolerance for APB_CLK	1.58	%	$= \frac{Min(PB1, PB2)}{2 \times 13 \times (bit\ \ time - PB2))}$

$$= \frac{4us}{2 \times (13 \times (10us - 4us))}$$

In this example, the concatenated bit time parameters are $(4-1)_3$ & $(5-1)_4$ & $(4-1)_2$ & $(2-1)_6$, and the Bit Timing Register is programmed to 0x34C1.

6.20.8 Register Map

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

Register	Offset	R/W	Description	Reset Value
CAN Base Address: **CAN_BA = 0x400A_0000**				
CAN_CON	CAN_BA+0x00	R/W	Control Register	0x0000_0001
CAN_STATUS	CAN_BA+0x04	R/W	Status Register	0x0000_0000
CAN_ERR	CAN_BA+0x08	R	Error Counter Register	0x0000_0000
CAN_BTIME	CAN_BA+0x0C	R/W	Bit Timing Register	0x0000_2301
CAN_IIDR	CAN_BA+0x10	R	Interrupt Identifier Register	0x0000_0000
CAN_TEST	CAN_BA+0x14	R/W	Test Register (Register Map Note 1)	0x0000_0080
CAN_BRPE	CAN_BA+0x18	R/W	Baud Rate Prescaler Extension Register	0x0000_0000
CAN_IFn_CREQ **n = 1,2**	CAN_BA+0x20 + (0x60 *(n-1))	R/W	IFn (Register Map Note 2) Command Request Registers	0x0000_0001
CAN_IFn_CMASK **n=1,2**	CAN_BA+0x24 + (0x60 *(n-1))	R/W	IFn Command Mask Registers	0x0000_0000
CAN_IFn_MASK1 **n=1,2**	CAN_BA+0x28 + (0x60 *(n-1))	R/W	IFn Mask 1 Registers	0x0000_FFFF
CAN_IFn_MASK2 **n=1,2**	CAN_BA+0x2C + (0x60 *(n-1))	R/W	IFn Mask 2 Registers	0x0000_FFFF
CAN_IFn_ARB1 **n=1,2**	CAN_BA+0x30 + (0x60 *(n-1))	R/W	IFn Arbitration 1 Registers	0x0000_0000
CAN_IFn_ARB2 **n=1,2**	CAN_BA+0x34 + (0x60 *(n-1))	R/W	IFn Arbitration 2 Registers	0x0000_0000
CAN_IFn_MCON **n=1,2**	CAN_BA+0x38 + (0x60 *(n-1))	R/W	IFn Message Control Registers	0x0000_0000
CAN_IFn_DAT_A1 **n=1,2**	CAN_BA+0x3C + (0x60 *(n-1))	R/W	IFn Data A1 Registers (Register Map Note 3)	0x0000_0000
CAN_IFn_DAT_A2 **n=1,2**	CAN_BA+0x40 + (0x60 *(n-1))	R/W	IFn Data A2 Registers (Register Map Note 3)	0x0000_0000
CAN_IFn_DAT_B1 **n=1,2**	CAN_BA+0x44 + (0x60 *(n-1))	R/W	IFn Data B1 Registers (Register Map Note 3)	0x0000_0000
CAN_IFn_DAT_B2 **n=1,2**	CAN_BA+0x48 + (0x60 *(n-1))	R/W	IFn Data B2 Registers (Register Map Note 3)	0x0000_0000
CAN_TXREQ1	CAN_BA+0x100	R	Transmission Request Register 1	0x0000_0000
CAN_TXREQ2	CAN_BA+0x104	R	Transmission Request Register 2	0x0000_0000
CAN_NDAT1	CAN_BA+0x120	R	New Data Register 1	0x0000_0000

CAN_NDAT2	CAN_BA+0x124	R	New Data Register 2	0x0000_0000
CAN_IPND1	CAN_BA+0x140	R	Interrupt Pending Register 1	0x0000_0000
CAN_IPND2	CAN_BA+0x144	R	Interrupt Pending Register 2	0x0000_0000
CAN_MVLD1	CAN_BA+0x160	R	Message Valid Register 1	0x0000_0000
CAN_MVLD2	CAN_BA+0x164	R	Message Valid Register 2	0x0000_0000
CAN_WU_EN	CAN_BA+0x168	R/W	Wake-up Enable Control Register	0x0000_0000
CAN_WU_STATUS	CAN_BA+0x16C	R/W	Wake-up Status Register	0x0000_0000

Note: 1. 0x00 & 0br0000000, where r signifies the actual value of the CAN_RX

2. IFn: The two sets of Message Interface Registers – IF1 and IF2, have identical function

3. An/Bn: The two sets of data registers – A1, A2 and B1, B2.

4. CAN_BA, where x = 0 or 1.

CAN Interface Reset State

After the hardware reset, the C_CAN registers hold the reset values which are given in the register description in 6.20.8

Register Map.

Additionally the bus-off state is reset and the output CAN_TX is set to recessive (HIGH). The value 0x0001 (Init = ‘1’) in the CAN Control Register enables the software initialization. The C_CAN does not influence the CAN bus until the application software resets the Init bit (CAN_CON[0]) to ‘0’.

The data stored in the Message RAM is not affected by a hardware reset. After powered on, the contents of the Message RAM are undefined.

CAN Register Map for Each Bit Function

Addr Offset	Register Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00h	CAN_CON	Reserved								Test	CCE	DAR	Res	EIE	SIE	IE	Init
04h	CAN_STATUS	Reserved								BOff	EWarn	EPass	RxOk	TxOk	LEC		
08h	CAN_ERR	RP	REC6-0							TEC7-0							
0Ch	CAN_BTIME	Res	TSeg2			TSeg1				SJW		BRP					
10h	CAN_IIDR	IntId15-8								IntId7-0							
14h	CAN_TEST	Reserved								Rx	Tx1	Tx0	LBack	Silent	Basic	Reserved	
18h	CAN_BRPE	Reserved												BRPE			
20h	CAN_IF1_CREQ	Busy	Reserved									Message Number					
24h	CAN_IF1_CMASK	Reserved								WR/RD	Mask	Arb	Control	ClrIntPnd	TxRqst/	Data A	Data B
28h	CAN_IF1_MASK1	Msk15-0															
2Ch	CAN_IF1_MASK2	MXtd	MDir	Res	Msk28-16												
30h	CAN_IF1_ARB1	ID15-0															
34h	CAN_IF1_ARB2	MsgVal	Xtd	Dir	ID28-16												

Addr Offset	Register Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
38h	CAN_IF1_MCON	NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst	EoB	Reserved			DLC3-0			
3Ch	CAN_IF1_DAT_A1	Data(1)								Data(0)							
40h	CAN_IF1_DAT_A2	Data(3)								Data(2)							
44h	CAN_IF1_DAT_B1	Data(5)								Data(4)							
48h	CAN_IF1_DAT_B2	Data(7)								Data(6)							
80h	CAN_IF2_CREQ	Busy	Reserved									Message Number					
84h	CAN_IF2_CMASK	Reserved								WR/RD	Mask	Arb	Control	ClrIntPnd	TxRqst/	Data A	Data B
88h	CAN_IF2_MASK1	Msk15-0															
8Ch	CAN_IF2_MASK2	MXtd	MDir	Res.	Msk28-16												
90h	CAN_IF2_ARB1	ID15-0															
94h	CAN_IF2_ARB2	MsgVal	Xtd	Dir	ID28-16												
98h	CAN_IF2_MCON	NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst	EoB	Reserved			DLC3-0			
9Ch	CAN_IF2_DAT_A1	Data(1)								Data(0)							

Addr Offset	Register Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A0h	CAN_IF2_DAT_A2	Data(3)								Data(2)							
A4h	CAN_IF2_DAT_B1	Data(5)								Data(4)							
A8h	CAN_IF2_DAT_B2	Data(7)								Data(6)							
100h	CAN_TXREQ1	TxRqst16-1															
104h	CAN_TXREQ2	TxRqst32-17															
120h	CAN_NDAT1	NewDat16-1															
124h	CAN_NDAT2	NewDat32-17															
140h	CAN_IPND1	IntPnd16-1															
144h	CAN_IPND2	IntPnd32-17															
160h	CAN_MVLD1	MsgVal16-1															
164h	CAN_MVLD2	MsgVal32-17															
168h	CAN_WU_EN	Reserved															WAKUP_EN
16Ch	CAN_WU_STATUS	Reserved															WAKUP_STS
170h	CAN_RAM_CEN	Reserved															RAM_CEN
Others	Reserved	Reserved															

Table 6-38 CAN Register Map for Each Bit Function

Note: Reserved bits are read as 0’ except for IFn Mask 2 Register where they are read as ’1’.

Res. = Reserved

6.20.9 Register Description

The C_CAN allocates an address space of 256 bytes. The registers are organized as 16-bit registers.

The two sets of interface registers (IF1 and IF2) control the software access to the Message RAM. They buffer the data to be transferred to and from the RAM, avoiding conflicts between software accesses and message reception/transmission.

CAN Control Register (CAN_CON)

Register	Offset	R/W	Description	Reset Value
CAN_CON	CAN_BA+0x00	R/W	Control Register	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Test	CCE	DAR	Reserved	EIE	SIE	IE	Init

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**Test**	**Test Mode Enable Bit** 0 = Normal Operation. 1 = Test Mode.
[6]	**CCE**	**Configuration Change Enable Bit** 0 = No write access to the Bit Timing Register. 1 = Write access to the Bit Timing Register (CAN_BTIME) allowed. (while Init bit (CAN_CON[0]) = 1).
[5]	**DAR**	**Automatic Re-transmission Disable Bit** 0 = Automatic Retransmission of disturbed messages Enabled. 1 = Automatic Retransmission Disabled.
[4]	**Reserved**	Reserved.
[3]	**EIE**	**Error Interrupt Enable Bit** 0 = Disabled - No Error Status Interrupt will be generated. 1 = Enabled - A change in the bits BOff (CAN_STATUS[7]) or EWarn (CAN_STATUS[6]) in the Status Register will generate an interrupt.
[2]	**SIE**	**Status Change Interrupt Enable Bit** 0 = Disabled - No Status Change Interrupt will be generated. 1 = Enabled - An interrupt will be generated when a message transfer is successfully completed or a CAN bus error is detected.
[1]	**IE**	**Module Interrupt Enable Bit** 0 = Module Interrupt Disabled. 1 = Module Interrupt Enabled.
[0]	**Init**	**Init Initialization** 0 = Normal Operation. 1 = Initialization is started.

Note: The bus-off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or resetting the Init bit (CAN_CON[0]). If the device goes in the bus-off state, it will set Init of its own accord, stopping all bus activities. Once Init has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operations. At the end of the bus-off recovery sequence, the Error Management Counters will be reset.

During the waiting time after resetting Init, each time a sequence of 11 recessive bits has been monitored, a Bit0Error code is written to the Status Register, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the proceeding of the bus-off recovery sequence.

CAN Status Register (CAN_STATUS)

Register	Offset	R/W	Description	Reset Value
CAN_STATUS	CAN_BA+0x04	R/W	Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
BOff	EWarn	EPass	RxOK	TxOK	LEC		

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**BOff**	**Bus-off Status (Read Only)** 0 = The CAN module is not in bus-off state. 1 = The CAN module is in bus-off state.
[6]	**EWarn**	**Error Warning Status (Read Only)** 0 = Both error counters are below the error warning limit of 96. 1 = At least one of the error counters in the EML has reached the error warning limit of 96.
[5]	**EPass**	**Error Passive (Read Only)** 0 = The CAN Core is error active. 1 = The CAN Core is in the error passive state as defined in the CAN Specification.
[4]	**RxOK**	**Received a Message Successfully** 0 = No message has been successfully received since this bit was last reset by the CPU. This bit is never reset by the CAN Core. 1 = A message has been successfully received since this bit was last reset by the CPU (independent of the result of acceptance filtering).
[3]	**TxOK**	**Transmitted a Message Successfully** 0 = Since this bit was reset by the CPU, no message has been successfully transmitted. This bit is never reset by the CAN Core. 1 = Since this bit was last reset by the CPU, a message has been successfully (error free and acknowledged by at least one other node) transmitted.
[2:0]	**LEC**	**Last Error Code (Type of the Last Error to Occur on the CAN Bus)** The LEC field holds a code, which indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. The unused code '7' may be written by the CPU to check for updates. The Table 6-39 describes the error code.

Error Code	Meanings
0	No Error
1	Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
2	Form Error: A fixed format part of a received frame has the wrong format.
3	AckError: The message this CAN Core transmitted was not acknowledged by another node.
4	Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant.
5	Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), though the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored Bus value was recessive. During bus-off recovery, this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceedings of the bus-off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).
6	CRCError: The CRC check sum was incorrect in the message received, the CRC received for an incoming message does not match with the calculated CRC for the received data.
7	Unused: When the LEC shows the value '7', no CAN bus event was detected since the CPU wrote this value to the LEC.

Table 6-39 Last Error Code

Status Interrupts

A Status Interrupt is generated by bits BOff (CAN_STATUS[7]) and EWarn (CAN_STATUS[6]) (Error Interrupt) or by RxOk (CAN_STATUS[4]), TxOk (CAN_STATUS[3]) and LEC (CAN_STATUS[2:0]) (Status Change Interrupt) assumed that the corresponding enable bits in the CAN Control Register are set. A change of bit EPass (CAN_STATUS[5]) or a write to RxOk, TxOk or LEC will never generate a Status Interrupt.

Reading the Status Register will clear the Status Interrupt value (8000h) in the Interrupt Register, if it is pending.

CAN Error Counter Register (CAN_ERR)

Register	Offset	R/W	Description	Reset Value
CAN_ERR	CAN_BA+0x08	R	Error Counter Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RP	**REC**						
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TEC							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**RP**	**Receive Error Passive** 0 = The Receive Error Counter is below the error passive level. 1 = The Receive Error Counter has reached the error passive level as defined in the CAN Specification.
[14:8]	**REC**	**Receive Error Counter** Actual state of the Receive Error Counter. Values between 0 and 127.
[7:0]	**TEC**	**Transmit Error Counter** Actual state of the Transmit Error Counter. Values between 0 and 255.

Bit Timing Register (CAN_BTIME)

Register	Offset	R/W	Description	Reset Value
CAN_BTIME	CAN_BA+0x0C	R/W	Bit Timing Register	0x0000_2301

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved	**TSeg2**			**TSeg1**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SJW		**BRP**					

Bits	Description	
[31:15]	**Reserved**	Reserved.
[14:12]	**TSeg2**	**Time Segment After Sample Point** 0x0-0x7: Valid values for TSeg2 are [0…7]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
[11:8]	**TSeg1**	**Time Segment Before the Sample Point Minus Sync_Seg** 0x01-0x0F: valid values for TSeg1 are [1…15]. The actual interpretation by the hardware of this value is such that one more than the value programmed is used.
[7:6]	**SJW**	**(Re)Synchronization Jump Width** 0x0-0x3: Valid programmed values are [0…3]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
[5:0]	**BRP**	**Baud Rate Prescaler** 0x01-0x3F: The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are [0…63]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Note: With a module clock APB_CLK of 8 MHz, the reset value of 0x2301 configures the C_CAN for a bit rate of 500 Kbit/s. The registers are only writable if bits CCE (CAN_CON[6]) and Init (CAN_CON[0]) are set.

Interrupt Identify Register (CAN_IIDR)

Register	Offset	R/W	Description	Reset Value
CAN_IIDR	CAN_BA+0x10	R	Interrupt Identifier Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
IntId							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IntId							

Bits	Description	
[15:0]	**IntId**	**Interrupt Identifier (Indicates the Source of the Interrupt)** If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the application software has cleared it. If IntId is different from 0x0000 and IE (CAN_CON[1]) is set, the IRQ interrupt signal to the EIC is active. The interrupt remains active until IntId is back to value 0x0000 (the cause of the interrupt is reset) or until IE is reset. The Status Interrupt has the highest priority. Among the message interrupts, the Message Object' s interrupt priority decreases with increasing message number. A message interrupt is cleared by clearing the Message Object's IntPnd bit (CAN_IFn_MCON[13]). The Status Interrupt is cleared by reading the Status Register.

IntId Value	Meanings
0x0000	No Interrupt is Pending
0x0001-0x0020	Number of Message Object which caused the interrupt.
0x0021-0x7FFF	Unused
0x8000	Status Interrupt
0x8001-0xFFFF	Unused

Table 6-40 Source of Interrupts

Test Register (CAN_TEST)

Register	Offset	R/W	Description	Reset Value
CAN_TEST	CAN_BA+0x14	R/W	Test Register (Register Map Note 1)	0x0000_0080

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Rx	Tx		LBack	Silent	Basic	Reserved	

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**Rx**	**Monitors the Actual Value of CAN_RX Pin (Read Only) *(1)** 0 = The CAN bus is dominant (CAN_RX = '0'). 1 = The CAN bus is recessive (CAN_RX = '1').
[6:5]	**Tx**	**Tx Control of CAN_TX Pin** 00 = Reset value, CAN_TX pin is controlled by the CAN Core. 01 = Sample Point can be monitored at CAN_TX pin. 10 = CAN_TX pin drives a dominant ('0') value. 11 = CAN_TX pin drives a recessive ('1') value.
[4]	**LBack**	**Loop Back Mode Enable Bit** 0 = Loop Back Mode is Disabled. 1 = Loop Back Mode is Enabled.
[3]	**Silent**	**Silent Mode** 0 = Normal operation. 1 = The module is in Silent Mode.
[2]	**Basic**	**Basic Mode** 0 = Basic Mode Disabled. 1= IF1 Registers used as Tx Buffer, IF2 Registers used as Rx Buffer.
[1:0]	**Reserved**	Reserved.

Reset value: 0000 0000 R000 0000 b (R:current value of RX pin)

Note: Write access to the Test Register is enabled by setting the Test bit (CAN_CON[7]). The different test functions may be combined, but Tx[1-0] "00" (CAN_TEST[6:5]) disturbs message transfer..

Baud Rate Prescaler Extension REGISTER (CAN_BRPE)

Register	Offset	R/W	Description	Reset Value
CAN_BRPE	CAN_BA+0x18	R/W	Baud Rate Prescaler Extension Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**BRPE**			

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3:0]	**BRPE**	**BRPE: Baud Rate Prescaler Extension** 0x00-0x0F: By programming BRPE, the Baud Rate Prescaler can be extended to values up to 1023. The actual interpretation by the hardware is that one more than the value programmed by BRPE (MSBs) and BTIME (LSBs) is used.

Message Interface Register Sets

There are two sets of Interface Registers, which are used to control the CPU access to the Message RAM. The Interface Registers avoid conflict between the CPU accesses to the Message RAM and CAN message reception and transmission by buffering the data to be transferred. A complete Message Object or parts of the Message Object may be transferred between the Message RAM and the IFn Message Buffer registers in one single transfer.

The function of the two interface register sets is identical except for the Basic test mode. They can be used the way one set of registers is used for data transfer to the Message RAM while the other set of registers is used for the data transfer from the Message RAM, allowing both processes to be interrupted by each other. The Table 6-41 provides an overview of the two Interface Register sets.

Each set of Interface Registers consists of Message Buffer Registers controlled by their own Command Registers. The Command Mask Register specifies the direction of the data transfer and which parts of a Message Object will be transferred. The Command Request Register is used to select a Message Object in the Message RAM as target or source for the transfer and to start the action specified in the Command Mask Register.

Address	IF1 Register Set	Address	IF2 Register Set
CAN_BA+0x20	IF1 Command Request	CAN_BA+0x80	IF2 Command Request
CAN_BA+0x24	IF1 Command Mask	CAN_BA+0x84	IF2 Command Mask
CAN_BA+0x28	IF1 Mask 1	CAN_BA+0x88	IF2 Mask 1
CAN_BA+0x2C	IF1 Mask 2	CAN_BA+0x8C	IF2 Mask 2
CAN_BA+0x30	IF1 Arbitration 1	CAN_BA+0x90	IF2 Arbitration 1
CAN_BA+0x34	IF1 Arbitration 2	CAN_BA+0x94	IF2 Arbitration 2
CAN_BA+0x38	IF1 Message Control	CAN_BA+0x98	IF2 Message Control
CAN_BA+0x3C	IF1 Data A 1	CAN_BA+0x9C	IF2 Data A 1
CAN_BA+0x40	IF1 Data A 2	CAN_BA+0xA0	IF2 Data A 2
CAN_BA+0x44	IF1 Data B 1	CAN_BA+0xA4	IF2 Data B 1
CAN_BA+0x48	IF1 Data B 2	CAN_BA+0xA8	IF2 Data B 2

Table 6-41 IF1 and IF2 Message Interface Register

IFn Command Request Register (CAN_IFn_CREQ)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_CREQ n = 1,2	CAN_BA+0x20 + (0x60 *(n-1))	R/W	IFn (Register Map Note 2) Command Request Registers	0x0000_0001

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Busy	**Reserved**						
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**Message Number**					

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**Busy**	**Busy Flag** 0 = Read/write action has finished. 1 = Writing to the IFn Command Request Register is in progress. This bit can only be read by the software.
[14:6]	**Reserved**	Reserved.
[5:0]	**Message Number**	**Message Number** 0x01-0x20: Valid Message Number, the Message Object in the Message RAM is selected for data transfer. 0x00: Not a valid Message Number, interpreted as 0x20. 0x21-0x3F: Not a valid Message Number, interpreted as 0x01-0x1F.

A message transfer is started as soon as the application software has written the message number to the Command Request Register. With this write operation, the Busy bit (CAN_IFn_CREQ[15]) is automatically set to notify the CPU that a transfer is in progress. After a waiting time of 3 to 6 APB_CLK periods, the transfer between the Interface Register and the Message RAM is completed. The Busy bit is cleared.

Note: When a Message Number that is not valid is written into the Command Request Register, the Message Number will be transformed into a valid value and that Message Object will be transferred.

IFn Command Mask Register (CAN_IFn_CMASK)

The control bits of the IFn Command Mask Register specify the transfer direction and select which of the IFn Message Buffer Registers are source or target of the data transfer.

Register	Offset	R/W	Description	Reset Value
CAN_IFn_CMASK n = 1,2	CAN_BA+0x24 + (0x60 *(n-1))	R/W	IFn Command Mask Registers	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WR/RD	**Mask**	**Arb**	**Control**	**ClrIntPnd**	**TxRqst/NewDat**	**DAT_A**	**DAT_B**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**WR/RD**	**Write / Read Mode** 0 = Read: Transfer data from the Message Object addressed by the Command Request Register into the selected Message Buffer Registers. 1 = Write: Transfer data from the selected Message Buffer Registers to the Message Object addressed by the Command Request Register.
[6]	**Mask**	**Access Mask Bits** Write Operation: 0 = Mask bits unchanged. 1 = Transfer Identifier Mask + MDir + MXtd to Message Object. Read Operation: 0 = Mask bits unchanged. 1 = Transfer Identifier Mask + MDir + MXtd to IFn Message Buffer Register.
[5]	**Arb**	**Access Arbitration Bits** Write Operation: 0 = Arbitration bits unchanged. 1 = Transfer Identifier + Dir (CAN_IFn_ARB2[13]) + Xtd (CAN_IFn_ARB2[14]) + MsgVal (CAN_IFn_ARB2[15]) to Message Object. Read Operation: 0 = Arbitration bits unchanged. 1 = Transfer Identifier + Dir + Xtd + MsgVal to IFn Message Buffer Register.
[4]	**Control**	**Control Access Control Bits** Write Operation: 0 = Control Bits unchanged. 1 = Transfer Control Bits to Message Object.

		Read Operation: 0 = Control Bits unchanged. 1 = Transfer Control Bits to IFn Message Buffer Register.
[3]	**ClrIntPnd**	**Clear Interrupt Pending Bit** Write Operation: When writing to a Message Object, this bit is ignored. Read Operation: 0 = IntPnd bit (CAN_IFn_MCON[13]) remains unchanged. 1 = Clear IntPnd bit in the Message Object.
[2]	**TxRqst/NewDat**	**Access Transmission Request Bit When Write Operation** 0 = TxRqst bit unchanged. 1 = Set TxRqst bit. **Note:** If a transmission is requested by programming bit TxRqst/NewDat in the IFn Command Mask Register, bit TxRqst in the IFn Message Control Register will be ignored. Access New Data Bit when Read Operation. 0 = NewDat bit remains unchanged. 1 = Clear NewDat bit in the Message Object. **Note:** A read access to a Message Object can be combined with the reset of the control bits IntPnd and NewDat. The values of these bits transferred to the IFn Message Control Register always reflect the status before resetting these bits.
[1]	**DAT_A**	**Access Data Bytes [3:0]** Write Operation: 0 = Data Bytes [3:0] unchanged. 1 = Transfer Data Bytes [3:0] to Message Object. Read Operation: 0 = Data Bytes [3:0] unchanged. 1 = Transfer Data Bytes [3:0] to IFn Message Buffer Register.
[0]	**DAT_B**	**Access Data Bytes [7:4]** Write Operation: 0 = Data Bytes [7:4] unchanged. 1 = Transfer Data Bytes [7:4] to Message Object. Read Operation: 0 = Data Bytes [7:4] unchanged. 1 = Transfer Data Bytes [7:4] to IFn Message Buffer Register.

IFn Mask 1 Register (CAN_IFn_MASK1)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_MASK1 **n = 1,2**	CAN_BA+0x28 + (0x60 *(n-1))	R/W	IFn Mask 1 Registers	0x0000_FFFF

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Msk[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Msk[7:0]							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**Msk[15:0]**	**Identifier Mask 15-0** 0 = The corresponding bit in the identifier of the message object cannot inhibit the match in the acceptance filtering. 1 = The corresponding identifier bit is used for acceptance filtering.

IFn Mask 2 Register (CAN_IFn_MASK2)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_MASK2 **n = 1,2**	CAN_BA+0x2C + (0x60 *(n-1))	R/W	IFn Mask 2 Registers	0x0000_FFFF

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
MXtd	**MDir**	**Reserved**	**Msk[28:24]**				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Msk[23:16]							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**MXtd**	**Mask Extended Identifier** 0 = The extended identifier bit (IDE) has no effect on the acceptance filtering. 1 = The extended identifier bit (IDE) is used for acceptance filtering. **Note:** When 11-bit (“standard”) Identifiers are used for a Message Object, the identifiers of received Data Frames are written into bits ID28 to ID18 (CAN_IFn_ARB2[12:2]). For acceptance filtering, only these bits together with mask bits Msk28 to Msk18 (CAN_IFn_MASK2[12:2]) are considered.
[14]	**MDir**	**Mask Message Direction** 0 = The message direction bit (Dir (CAN_IFn_ARB2[13])) has no effect on the acceptance filtering. 1 = The message direction bit (Dir) is used for acceptance filtering.
[13]	**Reserved**	Reserved.
[12:0]	**Msk[28:16]**	**Identifier Mask 28-16** 0 = The corresponding bit in the identifier of the message object cannot inhibit the match in the acceptance filtering. 1 = The corresponding identifier bit is used for acceptance filtering.

IFn Arbitration 1 Register (CAN_IFn_ARB1)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_ARB1 **n = 1,2**	CAN_BA+0x30 + (0x60 *(n-1))	R/W	IFn Arbitration 1 Registers	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ID[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[7:0]							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**ID[15:0]**	**Message Identifier 15-0** ID28 - ID0, 29-bit Identifier (“Extended Frame”). ID28 - ID18, 11-bit Identifier (“Standard Frame”)

IFn Arbitration 2 Register (CAN_IFn_ARB2)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_ARB2 **n = 1,2**	CAN_BA+0x34 + (0x60 *(n-1))	R/W	IFn Arbitration 2 Registers	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
MsgVal	**Xtd**	**Dir**	**ID[28:24]**				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[23:16]							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**MsgVal**	**Message Valid** 0 = The Message Object is ignored by the Message Handler. 1 = The Message Object is configured and should be considered by the Message Handler. **Note:** The application software must reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init (CAN_CON[0]). This bit must also be reset before the identifier Id28-0 (CAN_IFn_ARB1/2), the control bits Xtd (CAN_IFn_ARB2[14]), Dir (CAN_IFn_ARB2[13]), or the Data Length Code DLC3-0 (CAN_IFn_MCON[3:0]) are modified, or if the Messages Object is no longer required.
[14]	**Xtd**	**Extended Identifier** 0 = The 11-bit (“standard”) Identifier will be used for this Message Object. 1 = The 29-bit (“extended”) Identifier will be used for this Message Object.
[13]	**Dir**	**Message Direction** 0 = Direction is receive. On TxRqst, a Remote Frame with the identifier of this Message Object is transmitted. On reception of a Data Frame with matching identifier, that message is stored in this Message Object. 1 = Direction is transmit. On TxRqst, the respective Message Object is transmitted as a Data Frame. On reception of a Remote Frame with matching identifier, the TxRqst bit (CAN_IFn_CMASK[2]) of this Message Object is set (if RmtEn (CAN_IFn_MCON[9]) = one).
[12:0]	**ID[28:16]**	**Message Identifier 28-16** ID28 - ID0, 29-bit Identifier (“Extended Frame”). ID28 - ID18, 11-bit Identifier (“Standard Frame”)

IFn Message Control Register (CAN_IFn_MCON)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_MCON n = 1,2	CAN_BA+0x38 + (0x60 *(n-1))	R/W	IFn Message Control Registers	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
NewDat	MsgLst	IntPnd	UMask	TxIE	RxIE	RmtEn	TxRqst
7	6	5	4	3	2	1	0
EoB	Reserved			DLC			

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**NewDat**	**New Data** 0 = No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the application software. 1 = The Message Handler or the application software has written new data into the data portion of this Message Object.
[14]	**MsgLst**	Message Lost (only valid for Message Objects with direction = receive). 0 = No message lost since last time this bit was reset by the CPU. 1 = The Message Handler stored a new message into this object when NewDat was still set, the CPU has lost a message.
[13]	**IntPnd**	**Interrupt Pending** 0 = This message object is not the source of an interrupt. 1 = This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority.
[12]	**UMask**	**Use Acceptance Mask** 0 = Mask ignored. 1 = Use Mask (Msk28-0, MXtd, and MDir) for acceptance filtering. **Note:** If the UMask bit is set to one, the Message Object's mask bits have to be programmed during initialization of the Message Object before MsgVal bit (CAN_IFn_ARB2[15]) is set to one.
[11]	**TxIE**	**Transmit Interrupt Enable Bit** 0 = IntPnd (CAN_IFn_MCON[13]) will be left unchanged after the successful transmission of a frame. 1 = IntPnd will be set after a successful transmission of a frame.
[10]	**RxIE**	**Receive Interrupt Enable Bit** 0 = IntPnd (CAN_IFn_MCON[13]) will be left unchanged after a successful reception of a frame. 1 = IntPnd will be set after a successful reception of a frame.

[9]	**RmtEn**	**Remote Enable Bit** 0 = At the reception of a Remote Frame, TxRqst (CAN_IFn_MCON[8]) is left unchanged. 1 = At the reception of a Remote Frame, TxRqst is set.
[8]	**TxRqst**	**Transmit Request** 0 = This Message Object is not waiting for transmission. 1 = The transmission of this Message Object is requested and is not yet done.
[7]	**EoB**	**End of Buffer** 0 = Message Object belongs to a FIFO Buffer and is not the last Message Object of that FIFO Buffer. 1 = Single Message Object or last Message Object of a FIFO Buffer. **Note:** This bit is used to concatenate two or more Message Objects (up to 32) to build a FIFO Buffer. For single Message Objects (not belonging to a FIFO Buffer), this bit must always be set to one.
[6:4]	**Reserved**	Reserved.
[3:0]	**DLC**	**Data Length Code** 0-8: Data Frame has 0-8 data bytes. 9-15: Data Frame has 8 data bytes **Note:** The Data Length Code of a Message Object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the Message Handler stores a data frame, it will write the DLC to the value given by the received message. Data(0): 1st data byte of a CAN Data Frame Data(1): 2nd data byte of a CAN Data Frame Data(2): 3rd data byte of a CAN Data Frame Data(3): 4th data byte of a CAN Data Frame Data(4): 5th data byte of a CAN Data Frame Data(5): 6th data byte of a CAN Data Frame Data(6): 7th data byte of a CAN Data Frame Data(7): 8th data byte of a CAN Data Frame **Note:** The Data(0) byte is the first data byte shifted into the shift register of the CAN Core during a reception while the Data(7) byte is the last. When the Message Handler stores a Data Frame, it will write all the eight data bytes into a Message Object. If the Data Length Code is less than 8, the remaining bytes of the Message Object will be overwritten by unspecified values.

IFn Data A1 Register (CAN_IFn_DAT_A1)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_DAT_A1 **n = 1,2**	CAN_BA+0x3C + (0x60 *(n-1))	R/W	IFn Data A1 Registers (Register Map Note 3)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Data(1)							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data(0)							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**Data(1)**	**Data Byte 1** 2nd data byte of a CAN Data Frame
[7:0]	**Data(0)**	**Data Byte 0** 1st data byte of a CAN Data Frame

IFn Data A2 Register (CAN_IFn_DAT_A2)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_DAT_A2 **n = 1,2**	CAN_BA+0x40 + (0x60 *(n-1))	R/W	IFn Data A2 Registers (Register Map Note 3)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Data(3)							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data(2)							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**Data(3)**	**Data Byte 3** 4th data byte of CAN Data Frame
[7:0]	**Data(2)**	**Data Byte 2** 3rd data byte of CAN Data Frame

IFn Data B1 Register (CAN_IFn_DAT_B1)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_DAT_B1 **n = 1,2**	CAN_BA+0x44 + (0x60 *(n-1))	R/W	IFn Data B1 Registers (Register Map Note 3)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Data(5)							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data(4)							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**Data(5)**	**Data Byte 5** 6th data byte of CAN Data Frame
[7:0]	**Data(4)**	**Data Byte 4** 5th data byte of CAN Data Frame

IFn Data B2 Register (CAN_IFn_DAT_B2)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_DAT_B2 n = 1,2	CAN_BA+0x48 + (0x60 *(n-1))	R/W	IFn Data B2 Registers (Register Map Note 3)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Data(7)							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data(6)							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**Data(7)**	**Data Byte 7** 8th data byte of CAN Data Frame.
[7:0]	**Data(6)**	**Data Byte 6** 7th data byte of CAN Data Frame.

In a CAN Data Frame, Data [0] is the first, Data [7] is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

Message Object in the Message Memory

There are 32 Message Objects in the Message RAM. To avoid conflicts between application software access to the Message RAM and CAN message reception and transmission, the CPU cannot directly access the Message Objects, these accesses are handled through the IFn Interface Registers. The Table 6-42 provides an overview of the structures of a Message Object.

Message Object												
UMask	Msk [28:0]	MXtd	MDir	EoB	NewDat		MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
MsgVal	ID [28:0]	Xtd	Dir	DLC [3:0]	Data(0)	Data(1)	Data(2)	Data(3)	Data(4)	Data(5)	Data(6)	Data(7)

Table 6-42 Structure of a Message Object in the Message Memory

The Arbitration Registers ID28-0 (CAN_IFn_ARB1/2), Xtd (CAN_IFn_ARB2[14]) and Dir (CAN_IFn_ARB2[13]) are used to define the identifier and type of outgoing messages and are used (together with the mask registers Msk28-0 (CAN_IFn_MASK1/2), MXtd (CAN_IFn_MASK2[15]) and MDir (CAN_IFn_MASK2[14])) for acceptance filtering of incoming messages. A received message is stored in the valid Message Object with matching identifier and Direction = receive (Data Frame) or Direction = transmit (Remote Frame). Extended frames can be stored only in Message Objects with Xtd = one, standard frames in Message Objects with Xtd = zero. If a received message (Data Frame or Remote Frame) matches with more than one valid Message Object, it is stored into that with the lowest message number.

Message Handler Registers

All Message Handler registers are read only. Their contents (TxRqst (CAN_IFn_MCON[8]), NewDat (CAN_IFn_MCON[15]), IntPnd (CAN_IFn_MCON[13]) and MsgVal (CAN_IFn_ARB2[15]) bits of each Message Object and the Interrupt Identifier) are status information provided by the Message Handler FSM.

Transmission Request Register 1 (CAN_TXREQ1)

These registers hold the TxRqst bits of the 32 Message Objects. By reading the TxRqst bits, the software can check which Message Object in a Transmission Request is pending. The TxRqst bit of a specific Message Object can be set/reset by the application software through the IFn Message Interface Registers or by the Message Handler after reception of a Remote Frame or after a successful transmission.

Register	Offset	R/W	Description	Reset Value
CAN_TXREQ1	CAN_BA+0x100	R	Transmission Request Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TxRqst16-9							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TxRqst8-1							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**TxRqst16-1**	**Transmission Request Bits 16-1 (of All Message Objects)** 0 = This Message Object is not waiting for transmission. 1 = The transmission of this Message Object is requested and is not yet done. These bits are read only.

Transmission Request Register 2 (CAN_TXREQ2)

Register	Offset	R/W	Description	Reset Value
CAN_TXREQ2	CAN_BA+0x104	R	Transmission Request Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
TxRqst32-25							
7	6	5	4	3	2	1	0
TxRqst24-17							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**TxRqst32-17**	**Transmission Request Bits 32-17 (of All Message Objects)** 0 = This Message Object is not waiting for transmission. 1 = The transmission of this Message Object is requested and is not yet done. These bits are read only.

New Data Register 1 (CAN_NDAT1)

These registers hold the NewDat bits of the 32 Message Objects. By reading out the NewDat bits, the software can check for which Message Object the data portion was updated. The NewDat bit of a specific Message Object can be set/reset by the software through the IFn Message Interface Registers or by the Message Handler after reception of a Data Frame or after a successful transmission.

Register	Offset	R/W	Description	Reset Value
CAN_NDAT1	CAN_BA+0x120	R	New Data Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NewData16-9							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NewData8-1							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**NewData16-1**	**New Data Bits 16-1 (of All Message Objects)** 0 = No new data has been written into the data portion of this Message Object by the Message Handler since the last time this flag was cleared by the application software. 1 = The Message Handler or the application software has written new data into the data portion of this Message Object.

New Data Register 2 (CAN_NDAT2)

Register	Offset	R/W	Description	Reset Value
CAN_NDAT2	CAN_BA+0x124	R	New Data Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
NewData32-25							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NewData24-17							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**NewData32-17**	**New Data Bits 32-17 (of All Message Objects)** 0 = No new data has been written into the data portion of this Message Object by the Message Handler since the last time this flag was cleared by the application software. 1 = The Message Handler or the application software has written new data into the data portion of this Message Object.

Interrupt Pending Register 1 (CAN_IPND1)

These registers contain the IntPnd bits of the 32 Message Objects. By reading the IntPnd bits, the software can check for which Message Object an interrupt is pending. The IntPnd bit of a specific Message Object can be set/reset by the application software through the IFn Message Interface Registers or by the Message Handler after reception or after a successful transmission of a frame. This will also affect the value of IntId in the Interrupt Register.

Register	Offset	R/W	Description	Reset Value
CAN_IPND1	CAN_BA+0x140	R	Interrupt Pending Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
IntPnd16-9							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IntPnd8-1							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**IntPnd16-1**	**Interrupt Pending Bits 16-1 (of All Message Objects)** 0 = This message object is not the source of an interrupt. 1 = This message object is the source of an interrupt.

Interrupt Pending Register 2 (CAN_IPND2)

Register	Offset	R/W	Description	Reset Value
CAN_IPND2	CAN_BA+0x144	R	Interrupt Pending Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
IntPnd32-25							
7	6	5	4	3	2	1	0
IntPnd24-17							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**IntPnd32-17**	**Interrupt Pending Bits 32-17 (of All Message Objects)** 0 = This message object is not the source of an interrupt. 1 = This message object is the source of an interrupt.

Message Valid Register 1 (CAN_MVLD1)

These registers hold the MsgVal bits of the 32 Message Objects. By reading the MsgVal bits, the application software can check which Message Object is valid. The MsgVal bit of a specific Message Object can be set/reset by the application software via the IFn Message Interface Registers.

Register	Offset	R/W	Description	Reset Value
CAN_MVLD1	CAN_BA+0x160	R	Message Valid Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
MsgVal16- 9							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MsgVal8-1							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**MsgVal16-1**	**Message Valid Bits 16-1 (of All Message Objects) (Read Only)** 0 = This Message Object is ignored by the Message Handler. 1 = This Message Object is configured and should be considered by the Message Handler. Ex. CAN_MVLD1[0] means Message object No.1 is valid or not. If CAN_MVLD1[0] is set, message object No.1 is configured.

Message Valid Register 2 (CAN_MVLD2)

Register	Offset	R/W	Description	Reset Value
CAN_MVLD2	CAN_BA+0x164	R	Message Valid Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
MsgVal32-25							
7	6	5	4	3	2	1	0
MsgVal24-17							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**MsgVal32-17**	**Message Valid Bits 32-17 (of All Message Objects) (Read Only)** 0 = This Message Object is ignored by the Message Handler. 1 = This Message Object is configured and should be considered by the Message Handler. Ex.CAN_MVLD2[15] means Message object No.32 is valid or not. If CAN_MVLD2[15] is set, message object No.32 is configured.

Wake-up Enable Control Register (CAN_WU_EN)

Register	Offset	R/W	Description	Reset Value
CAN_WU_EN	CAN_BA+0x168	R/W	Wake-up Enable Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							WAKUP_EN

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**WAKUP_EN**	**Wake-up Enable Bit** 0 = The wake-up function Disabled. 1 = The wake-up function Enabled. **Note:** User can wake-up system when there is a falling edge in the CAN_Rx pin..

Wake-up Status Register (CAN_WU_STATUS)

Register	Offset	R/W	Description	Reset Value
CAN_WU_STATUS	CAN_BA+0x16C	R/W	Wake-up Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							WAKUP_STS

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**WAKUP_STS**	**Wake-up Status** 0 = No wake-up event occurred. 1 = Wake-up event occurred. **Note:** This bit can be cleared by writing '0'.

6.21 Touch Key (TK)

6.21.1 Overview

The capacitive touch-key sensing controller supports several programmable sensitivity levels for different applications to detect the finger touched or near the electrode covered by dielectric. It supports up to 16 touch-keys with single-scan or programmable periodic key-scans, and system can be waked-up by any key for low power applications.

6.21.2 Features

- Supports up to 16 touch-keys.
- Supports flexible reference channel setting, at least 1 reference channel needed.
- Programmable sensitivity levels for each channel.
- Programmable scanning speed for different applications.
- Supports any touch-key wake-up for low-power applications.
- Supports single key-scan and programmable periodic key-scan.
- Programmable interrupt options for key-scan complete with or without threshold control.

6.21.3 Block Diagram

The block diagram of Touch Key controller is depicted as Figure 6.21-1:

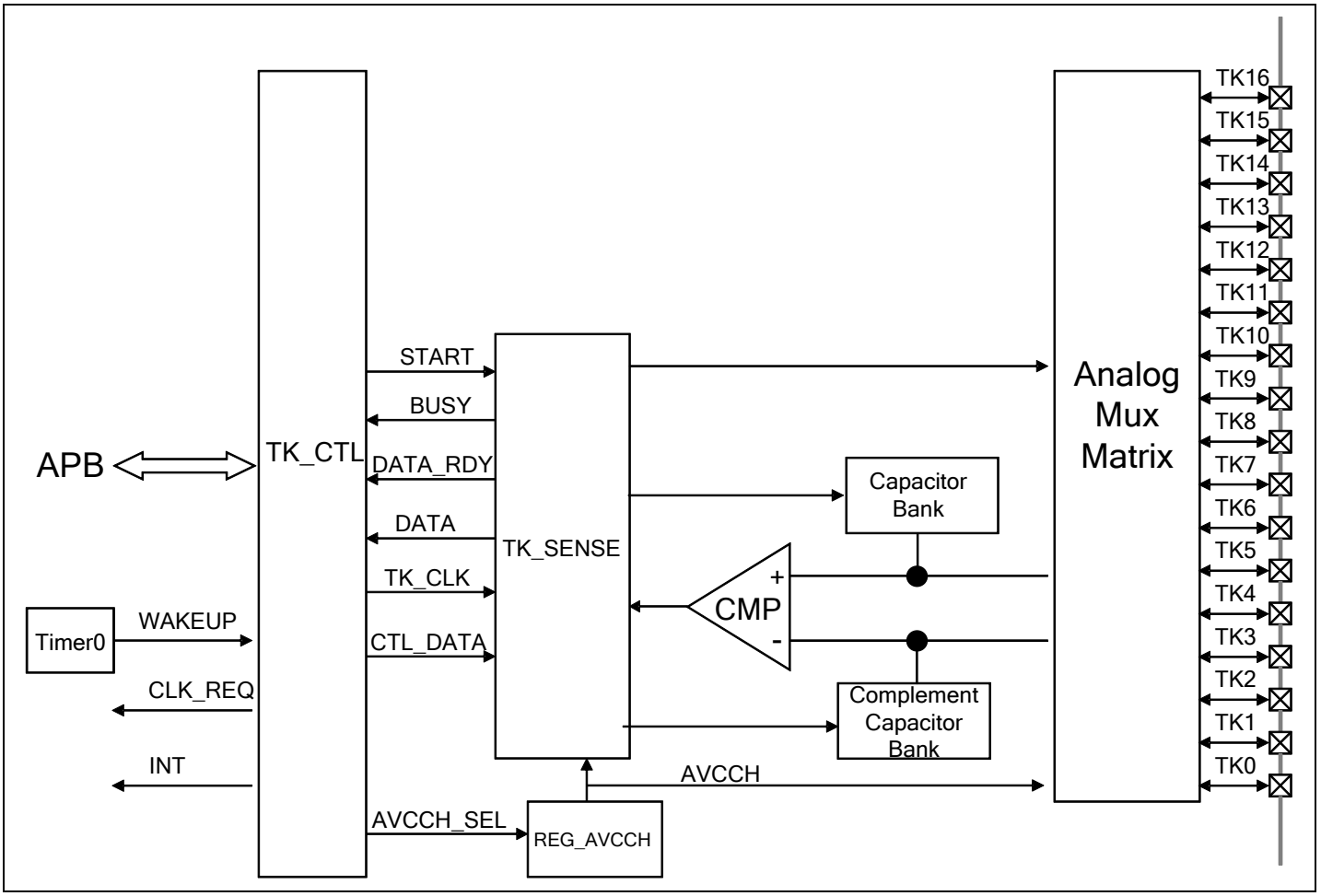


Figure 6.21-1 Touch Key block diagram

6.21.4 Basic Configuration

The Touch Key function pins are configured in SYS_GPB_MFPL[31:20], SYS_GPB_MFPH[31:28], SYS_GPB_MFPH[19:12], SYS_GPC_MFPH[3:0], SYS_GPD_MFPL[27:0], SYS_GPD_MFPH[7:0] and SYS_GPE_MFPL[15:12] Multiple Function Register.

The Touch Key clock are enabled in TKCKEN (CLK_APBCLK1[25]).

6.21.5 Functional Description

When a finger touches on a key-pad, capacitance of the key-pad sensed by Touch Key controller will be bigger than it is not touched. The capacitance is measured by Touch Key controller analog front-end circuit, user can read the sensed capacitance value to distinguish if a finger-touch event occurs or not.

There is a high/low sensing threshold control for each channel which supports keys scanning automatically, until any threshold setting condition is met. Thus processor can keep its normal run or Power-down mode to save power-consumption without interrupt.

The sensitivity level of Touch Key controller is programmable. The key scanning time is programmable for higher sensitivity or scanning speed.

Any key wake-up function and proximity detection are also supported.

In this chapter the letter x within register name denotes channel 0 to 16.

6.21.5.1 Touch Key Scan Mode

Single Scan Mode

In this mode, user must set TKEN (TK_CTL[31]) and SCAN (TK_CTL[24]) to high and other bits of TK_CTL, TK_REFCTL, TK_CCBDAT0, TK_CCBDAT 1, TK_CCBDAT 2, TK_CCBDAT 3, TK_CCBDAT 4, TK_IDLESEL and TK_POLCTL must be set according to application requirements for sensitivity setting. Those channels enabled by corresponding bits of TKSENx (TK_CTL[16:0]) register will be scanned successively when scan initiate. Once channel scanning starts, BUSY (TK_STATUS[0]) will be set until the scanning is complete. The SCIF (TK_STATUS[1]) register will be set when channel scanning completed and the sensed data in TK_DATn registers is valid for reading, according to those channels enabled in TK_CTL, n denotes number 0 to 4. SCIF (TK_STATUS[1]) can generate interrupt by setting SCINTEN (TK_INTEN[1]) register to high.

Periodic Scan Mode

This mode supports automatic scanning periodically. User can set TKEN (TK_CTL[31]) and TMRTRGEN (TK_CTL[25]) to high. In addition to set other bits of TK_CTL, TK_REFCTL, TK_CCBDAT0, TK_CCBDAT 1, TK_CCBDAT 2, TK_CCBDAT 3, TK_CCBDAT 4, TK_IDLESEL and TK_POLCTL according to application requirements, user must set Timer0 properly for determining period between scanning. Besides, user can apply wake-up function to key scanning in a low power system. By setting high/low threshold control properly, periodic scanning keeps even system is in sleep state, and until the system waked-up by interrupt which means whether a

finger-touch is met the threshold or not. Key scanning stops until the interrupt flags reset.

6.21.5.2 Reference Channel for Touch Key Scanning

There must be one channel at least as reference channel which PCB layout has special concern. TK16 is assigned as default reference channel automatically if there is no reference channel assigned by user. Touch Key controller malfunctions without assigning physical reference channel(s) to pin(s) in the application.

6.21.5.3 Idle State and Polarity Control

Those keys-pads are not being sensed always keep in Idle-State, their output level can be pre-determined in TK_IDLESEL register. For another sensing configuration, their output level is separated to two states: Idle-state and Polarity-State if their polarity control is activated in TK_POLCTL register. Their output level is pre-determined in TK_POLSEL register.

6.21.5.4 Sensing Time

Sensing Time is the time needed for each key-pad sensing, it is composed of SENPTCTL (TK_REFCTL[29:28]) and SENTCTL (TK_REFCTL[25:24]). Shorter Sensing Time comes with poor sensitivity and less power-consumption, vice versa.

Sensing Time = SENTCTL x SENPTCTL

6.21.5.5 Sensitivity Configuration

Sensitivity can be adjusted by setting Sensing Time properly. Besides, select capacitor bank polarity source properly which pre-determined in CBPOLSEL (TK_POLCTL[5:4]) also affects sensitivity. User can choose AVCCH as capacitor bank polarity source to have more sensitivity choice by programming AVCCH to proper level, which is pre-determined in AVCCHSEL (TK_CTL[22:20]).

6.21.5.6 Scan Interrupt Type

Scan Complete without Threshold Control Interrupt

SCIF (TK_STATUS[1]) is always set when key-scan is complete. An interrupt is generated if SCINTEN (TK_INTEN[1]) is set.

Scan Complete with Threshold Control Interrupt

Different with SCIF (TK_STATUS[1]), TKIFx (TK_STATUS[24:8]) is set if and only if corresponding scan result met its threshold control requirement. Besides, set SCTHIEN (TK_INTEN[0]) rather than SCINTEN (TK_INTEN[1]) to generate interrupt when a potential key touch/release is detected. There are two modes selected by THIMOD (TK_INTEN[31]). The high/low threshold control for each channel can be pre-determined in TK_THm_n registers, m,n denotes two adjacent channels' number, for example, TKTH0_1.

Edge Trigger Mode

In this mode, TKIFx is set if only if either the corresponding TKDATx is the first time greater than HTHx since it become greater than LTHx, or the first time less than LTHx since it become less than

HTHx. HTHx means a potential key touch occurs and LTHx means a potential key release occurs. As shown in Figure 6.21-2.This mode is very useful to generate interrupts only if a potential key touch/release occurs, and save system power consumption.

Level Trigger Mode

In this alternative mode, TKIFx is set if TKDATx is greater than HTHx. TKDATx greater than HTHx means a potential key touch occurs. As shown in Figure 6.21-3.

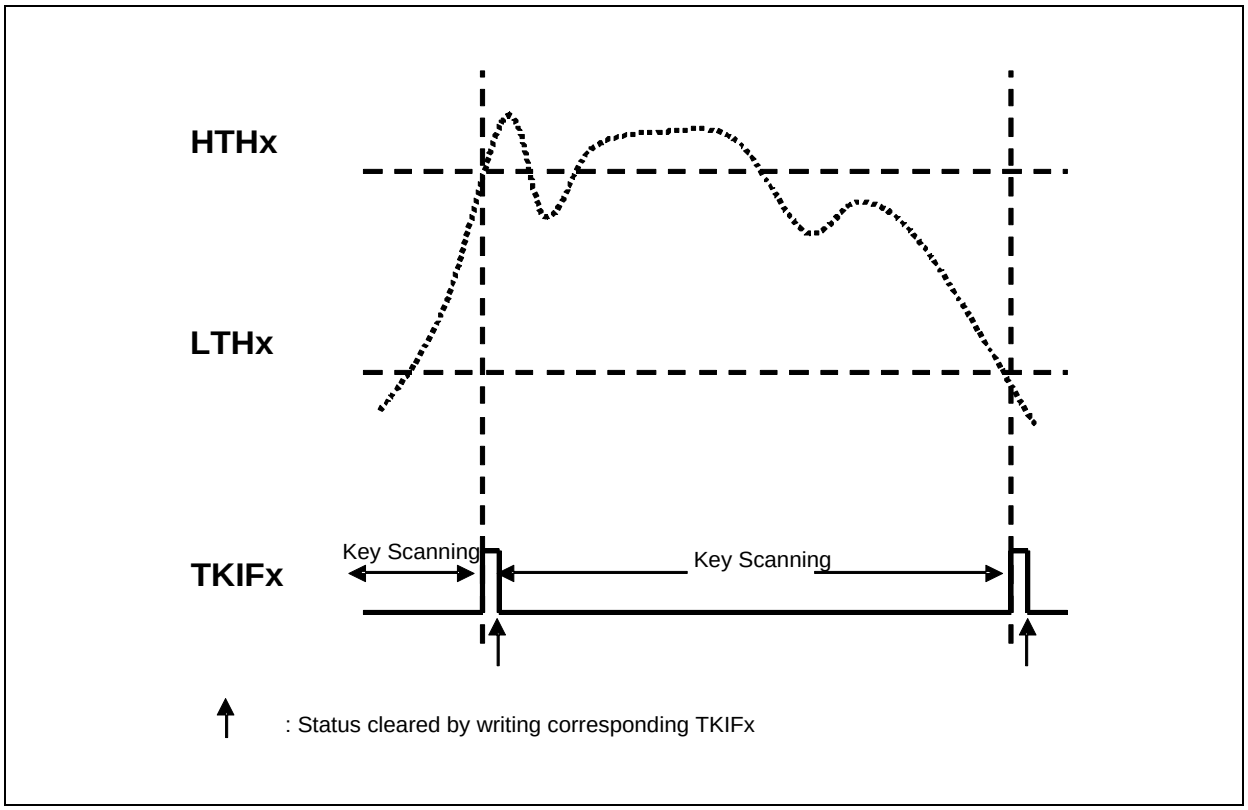


Figure 6.21-2 Touch Key Threshold Control in Edge Trigger Mode

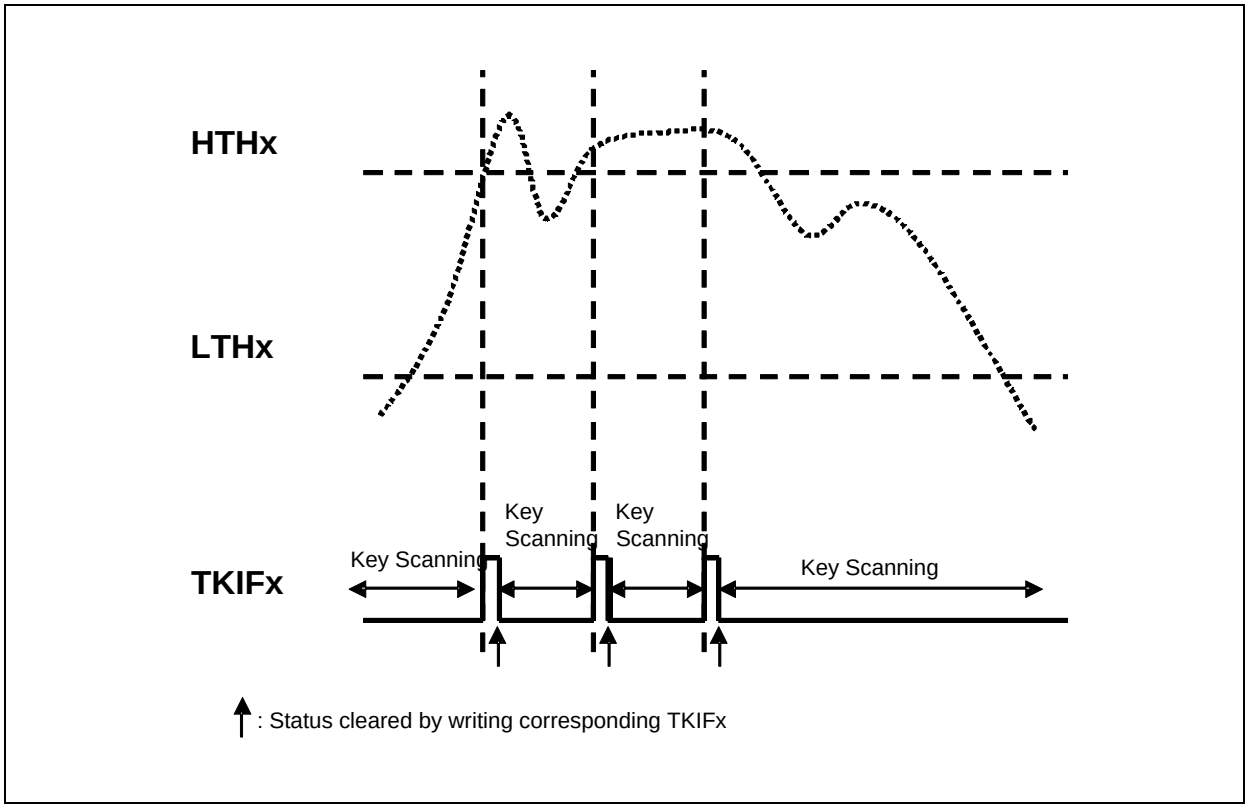


Figure 6.21-3 Touch Key Threshold Control in Level Trigger Mode

6.21.5.7 Low Power Consumption Solution

Non-stop key scanning in a low power system can be easily achieved. User can use Timer0 to wake-up Touch Key controller for key scanning periodically. Touch Key controller requests HIRC for key scanning only when waked-up, and keeps system in power-down state. Interrupts generated when a potential key scanning event occurs, otherwise, Touch Key controller will terminate key scanning without interrupt and makes itself in Power-down mode.

Wake-up by Key Touch/Release

To enable threshold control for interrupt generation, system keeps in low power until any potential key touch/release detected.

Wake-up by Any Key Touch

To save system more power consumption, user can use Any Key Wake-up function by setting SCANALL (TK_REFCTL[23]). All channels enabled but not assigned as reference channel is scanned, and scanning data is valid in TKDAT0 (TK_DAT0[7:0]) . The CCBDAT0 (TK_CCBDAT0[7:0]) may be different from normal and need to be calibrated individually. Proximity detection is also achievable by using this mode.

6.21.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
TK Base Address: **TK_BA = 0x400E_2000**				
TK_CTL	TK_BA+0x00	R/W	Touch Key Scan Control Register	0x0071_0000
TK_REFCTL	TK_BA+0x04	R/W	Touch Key Reference Control Register	0x2201_0000
TK_CCBDAT0	TK_BA+0x08	R/W	Touch Key Complement Capacitor Bank Data Register 0	0x0000_0000
TK_CCBDAT1	TK_BA+0x0C	R/W	Touch Key Complement Capacitor Bank Data Register 1	0x0000_0000
TK_CCBDAT2	TK_BA+0x10	R/W	Touch Key Complement Capacitor Bank Data Register 2	0x0000_0000
TK_CCBDAT3	TK_BA+0x14	R/W	Touch Key Complement Capacitor Bank Data Register 3	0x0000_0000
TK_CCBDAT4	TK_BA+0x18	R/W	Touch Key Complement Capacitor Bank Data Register 4	0x0000_0000
TK_IDLESEL	TK_BA+0x1C	R/W	Touch Key Idle State Control Register	0xFFFF_FFFF
TK_POLSEL	TK_BA+0x20	R/W	Touch Key Polarity Select Register	0xFFFF_FFFF
TK_POLCTL	TK_BA+0x24	R/W	Touch Key Polarity Control Register	0x0000_001F
TK_STATUS	TK_BA+0x28	R/W	Touch Key Status Register	0x0000_0000
TK_DAT0	TK_BA+0x2C	R	Touch Key Data Register 0	0x0000_0000
TK_DAT1	TK_BA+0x30	R	Touch Key Data Register 1	0x0000_0000
TK_DAT2	TK_BA+0x34	R	Touch Key Data Register 2	0x0000_0000
TK_DAT3	TK_BA+0x38	R	Touch Key Data Register 3	0x0000_0000
TK_DAT4	TK_BA+0x3C	R	Touch Key Data Register 4	0x0000_0000
TK_INTEN	TK_BA+0x40	R/W	Touch Key Interrupt Enable Register	0x0000_0000
TK_TH0_1	TK_BA+0x44	R/W	Touch Key TK0/TK1 Threshold Control Register	0xFF00_FF00
TK_TH2_3	TK_BA+0x48	R/W	Touch Key TK2/TK3 Threshold Control Register	0xFF00_FF00
TK_TH4_5	TK_BA+0x4C	R/W	Touch Key TK4/TK5 Threshold Control Register	0xFF00_FF00
TK_TH6_7	TK_BA+0x50	R/W	Touch Key TK6/TK7 Threshold Control Register	0xFF00_FF00
TK_TH8_9	TK_BA+0x54	R/W	Touch Key TK8/TK9 Threshold Control Register	0xFF00_FF00
TK_TH10_11	TK_BA+0x58	R/W	Touch Key TK10/TK11 Threshold Control Register	0xFF00_FF00
TK_TH12_13	TK_BA+0x5C	R/W	Touch Key TK12/TK13 Threshold Control Register	0xFF00_FF00
TK_TH14_15	TK_BA+0x60	R/W	Touch Key TK14/TK15 Threshold Control Register	0xFF00_FF00
TK_TH16	TK_BA+0x64	R/W	Touch Key TK16 Threshold Control Register	0x0000_FF00

6.21.7 Register Description

Touch Key Scan Control Register (TK_CTL)

Register	Offset	R/W	Description	Reset Value
TK_CTL	TK_BA+0x00	R/W	Touch Key Scan Control Register	0x0071_0000

31	30	29	28	27	26	25	24
TKEN	Reserved					TMRTRGEN	SCAN
23	22	21	20	19	18	17	16
Reserved	AVCCHSEL			Reserved			TKSEN16
15	14	13	12	11	10	9	8
TKSEN15	TKSEN14	TKSEN13	TKSEN12	TKSEN11	TKSEN10	TKSEN9	TKSEN8
7	6	5	4	3	2	1	0
TKSEN7	TKSEN6	TKSEN5	TKSEN4	TKSEN3	TKSEN2	TKSEN1	TKSEN0

Bits	Description	
[31]	**TKEN**	**Touch Key Scan Enable Bit** 0 = Disable Touch Key Function. 1 = Enable Touch Key Function.
[30:26]	**Reserved**	Reserved.
[25]	**TMRTRGEN**	**Timer Trigger Enable Bit** 0 = Disable timer to trigger key scan. 1 = Enable timer triggers key scan periodically. Key scan will be initiated by Timer0 periodically.
[24]	**SCAN**	**Scan** Write an “1’ to this bit will immediately initiate key scan on all channels which are enabled. This bit will be self-cleared after key scan started.
[23]	**Reserved**	Reserved.
[22:20]	**AVCCHSEL**	**AVCCH Voltage Select** 000 = 1/16 V_{DD}. 001 = 1/8 V_{DD}. 010 = 3/16 V_{DD}. 011 = 1/4 V_{DD}. 100 = 5/16 V_{DD}. 101 = 3/8 V_{DD}. 110 = 7/16 V_{DD}. 111 = 1/2 V_{DD}.
[19:17]	**Reserved**	Reserved.

Bits	Description	
[16]	**TKSEN16**	**TK16 Scan Enable Bit** This bit is ignored if TKREN16 (TK_REFCTL[16]) is “1”. 0 = TKDAT16 (TK_DAT4[7:0]) is invalid. 1 = TK16 is always enabled for key scan. TKDAT16 (TK_DAT4[7:0]) is valid.
[15]	**TKSEN15**	**TK15 Scan Enable Bit** This bit is ignored if TKREN15 (TK_REFCTL[15]) is “1”. 0 = TKDAT15 (TK_DAT3[31:24]) is invalid. 1 = TK15 is always enabled for key scan. TKDAT15 (TK_DAT3[31:24]) is valid.
[14]	**TKSEN14**	**TK14 Scan Enable Bit** This bit is ignored if TKREN14 (TK_REFCTL[14]) is “1”. 0 = TKDAT14 (TK_DAT3[23:16]) is invalid. 1 = TK14 is always enabled for key scan. TKDAT14 (TK_DAT3[23:16]) is valid.
[13]	**TKSEN13**	**TK13 Scan Enable Bit** This bit is ignored if TKREN13 (TK_REFCTL[13]) is “1”. 0 = TKDAT13 (TK_DAT3[15:8]) is invalid. 1 = TK13 is always enable for key scan. TKDAT13 (TK_DAT3[15:8]) is valid.
[12]	**TKSEN12**	**TK12 Scan Enable Bit** This bit is ignored if TKREN12 (TK_REFCTL[12]) is “1”. 0 = TKDAT12 (TK_DAT3[7:0]) is invalid. 1 = TK12 is always enable for Touch Key scan. TKDAT12 (TK_DAT3[7:0]) is valid.
[11]	**TKSEN11**	**TK11 Scan Enable Bit** This bit is ignored if TKREN11 (TK_REFCTL[11]) is “1”. 0 = TKDAT11 (TK_DAT2[31:24]) is invalid. 1 = TK11 is always enable for Touch Key scan. TKDAT11 (TK_DAT2[31:24]) is valid.
[10]	**TKSEN10**	**TK10 Scan Enable Bit** This bit is ignored if TKREN10 (TK_REFCTL[10]) is “1”. 0 = TKDAT10 (TK_DAT2[23:16]) is invalid. 1 = TK10 is always enable for Touch Key scan. TKDAT10 (TK_DAT2[23:16]) is valid.
[9]	**TKSEN9**	**TK9 Scan Enable Bit** This bit is ignored if TKREN9 (TK_REFCTL[9]) is “1”. 0 = TKDAT9 (TK_DAT2[15:8]) is invalid. 1 = TK9 is always enable for Touch Key scan. TKDAT9 (TK_DAT2[15:8]) is valid.
[8]	**TKSEN8**	**TK8 Scan Enable Bit** This bit is ignored if TKREN8 (TK_REFCTL[8]) is “1”. 0 = TKDAT8 (TK_DAT2[7:0]) is invalid. 1 = TK8 is always enable for Touch Key scan. TKDAT8 (TK_DAT2[7:0]) is valid.
[7]	**TKSEN7**	**TK7 Scan Enable Bit** This bit is ignored if TKREN7 (TK_REFCTL[7]) is “1”. 0 = TKDAT7 (TK_DAT1[31:24]) is invalid. 1 = TK7 is always enable for Touch Key scan. TKDAT7 (TK_DAT1[31:24]) is valid.

Bits	Description	
[6]	**TKSEN6**	**TK6 Scan Enable Bit** This bit is ignored if TKREN6 (TK_REFCTL[6]) is “1”. 0 = TKDAT6 (TK_DAT1[23:16]) is invalid. 1 = TK6 is always enable for Touch Key scan. TKDAT6 (TK_DAT1[23:16]) is valid.
[5]	**TKSEN5**	**TK5 Scan Enable Bit** This bit is ignored if TKREN5 (TK_REFCTL[5]) is “1”. 0 = TKDAT5 (TK_DAT1[15:8]) is invalid. 1 = TK5 is always enable for Touch Key scan. TKDAT5 (TK_DAT1[15:8]) is valid.
[4]	**TKSEN4**	**TK4 Scan Enable Bit** This bit is ignored if TKREN4 (TK_REFCTL[4]) is “1”. 0 = TKDAT4 (TK_DAT1[7:0]) is invalid. 1 = TK4 is always enable for Touch Key scan. TKDAT4 (TK_DAT1[7:0]) is valid.
[3]	**TKSEN3**	**TK3 Scan Enable Bit** 0 = TKDAT3 (TK_DAT0[31:24]) is invalid. 1 = TK3 is always enable for Touch Key scan. TKDAT3 (TK_DAT0[31:24]) is valid. This bit is ignored if TKREN3 (TK_REFCTL[3]) is “1”.
[2]	**TKSEN2**	**TK2 Scan Enable Bit** This bit is ignored if TKREN2 (TK_REFCTL[2]) is “1”. 0 = TKDAT2 (TK_DAT0[23:16]) is invalid. 1 = TK2 is always enable for Touch Key scan. TKDAT2 (TK_DAT0[23:16]) is valid.
[1]	**TKSEN1**	**TK1 Scan Enable Bit** This bit is ignored if TKREN1 (TK_REFCTL[1]) is “1”. 0 = TKDAT1 (TK_DAT0[15:8]) is invalid. 1 = TK1 is always enable for Touch Key scan. TKDAT1 (TK_DAT0[15:8]) is valid.
[0]	**TKSEN0**	**TK0 Scan Enable Bit** This bit is ignored if TKREN0 (TK_REFCTL[0]) is “1” except SCANALL (TK_REFCTL[23]) is “1”. 0 = TKDAT0 (TK_DAT0[7:0]) is invalid. 1 = TK0 is always enable for Touch Key scan. TKDAT0 (TK_DAT0[7:0]) is valid.

Touch Key Reference Control Register (TK_REFCTL)

Register	Offset	R/W	Description	Reset Value
TK_REFCTL	TK_BA+0x04	R/W	Touch Key Reference Control Register	0x2201_0000

31	30	29	28	27	26	25	24
Reserved		SENPTCTL		Reserved		SENTCTL	
23	22	21	20	19	18	17	16
SCANALL	Reserved						TKREN16
15	14	13	12	11	10	9	8
TKREN15	TKREN14	TKREN13	TKREN12	TKREN11	TKREN10	TKREN9	TKREN8
7	6	5	4	3	2	1	0
TKREN7	TKREN6	TKREN5	TKREN4	TKREN3	TKREN2	TKREN1	TKREN0

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29:28]	**SENPTCTL**	**Touch Key Sensing Pulse Width Time Control** 00 = 1us. 01 = 2us. 10 = 4us. 11 = 8us.
[27:26]	**Reserved**	Reserved.
[25:24]	**SENTCTL**	**Touch Key Sensing Time Control** 00 = 128 x SENPTCTL. 01 = 255 x SENPTCTL. 10 = 511 x SENPTCTL. 11 = 1023 x SENPTCTL.
[23]	**SCANALL**	**All Key Scan Enable Bit** This function is used for low power key scanning operation. TKDAT0 (TK_DAT0[7:0]) is the only one valid data when key scan is complete. 0 = Disable All Keys Scan function. 1 = Enable All Keys Scan function.
[22:17]	**Reserved**	Reserved.
[16]	**TKREN16**	**TK16 Reference Enable Bit** 0 = TK16 is not reference. 1 = TK16 is set as reference, and TKDAT16 (TK_DAT4[7:0]) is invalid. **Note:** This bit is forced to “1” automatically if none is set as reference.
[15]	**TKREN15**	**TK15 Reference Enable Bit** 0 = TK15 is not reference. 1 = TK15 is set as reference, and TKDAT15 (TK_DAT3[31:24]) is invalid.

Bits	Description	
[14]	**TKREN14**	**TK14 Reference Enable Bit** 0 = TK14 is not reference. 1 = TK14 is set as reference, and TKDAT14 (TK_DAT3[23:16]) is invalid.
[13]	**TKREN13**	**TK13 Reference Enable Bit** 0 = TK13 is not reference. 1 = TK13 is set as reference, and TKDAT13 (TK_DAT3[15:8]) is invalid.
[12]	**TKREN12**	**TK12 Reference Enable Bit** 0 = TK12 is not reference. 1 = TK12 is set as reference, and TKDAT12 (TK_DAT3[7:0]) is invalid.
[11]	**TKREN11**	**TK11 Reference Enable Bit** 0 = TK11 is not reference. 1 = TK11 is set as reference, and TKDAT11 (TK_DAT2[31:24]) is invalid.
[10]	**TKREN10**	**TK10 Reference Enable Bit** 0 = TK10 is not reference. 1 = TK10 is set as reference, and TKDAT10 (TK_DAT2[23:16]) is invalid.
[9]	**TKREN9**	**TK9 Reference Enable Bit** 0 = TK9 is not reference. 1 = TK9 is set as reference, and TKDAT9 (TK_DAT2[15:8]) is invalid.
[8]	**TKREN8**	**TK8 Reference Enable Bit** 0 = TK8 is not reference. 1 = TK8 is set as reference, and TKDAT8 (TK_DAT2[7:0]) is invalid.
[7]	**TKREN7**	**TK7 Reference Enable Bit** 0 = TK7 is not reference. 1 = TK7 is set as reference, and TKDAT7 (TK_DAT1[31:24]) is invalid.
[6]	**TKREN6**	**TK6 Reference Enable Bit** 0 = TK6 is not reference. 1 = TK6 is set as reference, and TKDAT6 (TK_DAT1[23:16]) is invalid.
[5]	**TKREN5**	**TK5 Reference Enable Bit** 0 = TK5 is not reference. 1 = TK5 is set as reference, and TKDAT5 (TK_DAT1[15:8]) is invalid.
[4]	**TKREN4**	**TK4 Reference Enable Bit** 0 = TK4 is not reference. 1 = TK4 is set as reference, and TKDAT4 (TK_DAT1[7:0]) is invalid.
[3]	**TKREN3**	**TK3 Reference Enable Bit** 0 = TK3 is not reference. 1 = TK3 is set as reference, and TKDAT3 (TK_DAT0[31:24]) is invalid.
[2]	**TKREN2**	**TK2 Reference Enable Bit** 0 = TK2 is not reference. 1 = TK2 is set as reference, and TKDAT2 (TK_DAT0[23:16]) is invalid.
[1]	**TKREN1**	**TK1 Reference Enable Bit** 0 = TK1 is not reference. 1 = TK1 is set as reference, and TKDAT1 (TK_DAT0[15:8]) is invalid.

Bits	Description	
[0]	**TKREN0**	**TK0 Reference Enable Bit** 0 = TK0 is not reference. 1 = TK0 is set as reference, and TKDAT0 (TK_DAT0[7:0]) is invalid except SCANALL (TK_REFCTL[23]) is “1”.

Touch Key Complement Capacitor Bank Data Register 0 (TK_CCBDAT0)

Register	Offset	R/W	Description	Reset Value
TK_CCBDAT0	TK_BA+0x08	R/W	Touch Key Complement Capacitor Bank Data Register 0	0x0000_0000

31	30	29	28	27	26	25	24
CCBDAT3							
23	22	21	20	19	18	17	16
CCBDAT2							
15	14	13	12	11	10	9	8
CCBDAT1							
7	6	5	4	3	2	1	0
CCBDAT0							

Bits	Description	
[31:24]	**CCBDAT3**	**TK3 Complement CB Data** This is register is used for TK3 sensitivity adjustment.
[23:16]	**CCBDAT2**	**TK2 Complement CB Data** This is register is used for TK2 sensitivity adjustment.
[15:8]	**CCBDAT1**	**TK1 Complement CB Data** This is register is used for TK1 sensitivity adjustment.
[7:0]	**CCBDAT0**	**TK0 Complement CB Data** This is register is used for TK0 sensitivity adjustment.

Touch Key Complement Capacitor Bank Data Register 1 (TK_CCBDAT1)

Register	Offset	R/W	Description	Reset Value
TK_CCBDAT1	TK_BA+0x0C	R/W	Touch Key Complement Capacitor Bank Data Register 1	0x0000_0000

31	30	29	28	27	26	25	24
CCBDAT7							
23	22	21	20	19	18	17	16
CCBDAT6							
15	14	13	12	11	10	9	8
CCBDAT5							
7	6	5	4	3	2	1	0
CCBDAT4							

Bits	Description	
[31:24]	**CCBDAT7**	**TK7 Complement CB Data** This is register is used for TK7 sensitivity adjustment.
[23:16]	**CCBDAT6**	**TK6 Complement CB Data** This is register is used for TK6 sensitivity adjustment.
[15:8]	**CCBDAT5**	**TK5 Complement CB Data** This is register is used for TK5 sensitivity adjustment.
[7:0]	**CCBDAT4**	**TK4 Complement CB Data** This is register is used for TK4 sensitivity adjustment.

Touch Key Complement Capacitor Bank Data Register 2 (TK_CCBDAT2)

Register	Offset	R/W	Description	Reset Value
TK_CCBDAT2	TK_BA+0x10	R/W	Touch Key Complement Capacitor Bank Data Register 2	0x0000_0000

31	30	29	28	27	26	25	24
CCBDAT11							
23	22	21	20	19	18	17	16
CCBDAT10							
15	14	13	12	11	10	9	8
CCBDAT9							
7	6	5	4	3	2	1	0
CCBDAT8							

Bits	Description	
[31:24]	**CCBDAT11**	**TK11 Complement CB Data** This is register is used for TK11 sensitivity adjustment.
[23:16]	**CCBDAT10**	**TK10 Complement CB Data** This is register is used for TK10 sensitivity adjustment.
[15:8]	**CCBDAT9**	**TK9 Complement CB Data** This is register is used for TK9 sensitivity adjustment.
[7:0]	**CCBDAT8**	**TK8 Complement CB Data** This is register is used for TK8 sensitivity adjustment.

Touch Key Complement Capacitor Bank Data Register 3 (TK_CCBDAT3)

Register	Offset	R/W	Description	Reset Value
TK_CCBDAT3	TK_BA+0x14	R/W	Touch Key Complement Capacitor Bank Data Register 3	0x0000_0000

31	30	29	28	27	26	25	24
CCBDAT15							
23	22	21	20	19	18	17	16
CCBDAT14							
15	14	13	12	11	10	9	8
CCBDAT13							
7	6	5	4	3	2	1	0
CCBDAT12							

Bits	Description	
[31:24]	**CCBDAT15**	**TK15 Complement CB Data** This is register is used for TK15 sensitivity adjustment.
[23:16]	**CCBDAT14**	**TK14 Complement CB Data** This is register is used for TK14 sensitivity adjustment.
[15:8]	**CCBDAT13**	**TK13 Complement CB Data** This is register is used for TK13 sensitivity adjustment.
[7:0]	**CCBDAT12**	**TK12 Complement CB Data** This is register is used for TK12 sensitivity adjustment.

Touch Key Complement Capacitor Bank Data Register 4 (TK_CCBDAT4)

Register	Offset	R/W	Description	Reset Value
TK_CCBDAT4	TK_BA+0x18	R/W	Touch Key Complement Capacitor Bank Data Register 4	0x0000_0000

31	30	29	28	27	26	25	24
REFCBDAT							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
CCBDAT16							

Bits	Description	
[31:24]	**REFCBDAT**	**Reference CB Data**
[23:8]	**Reserved**	Reserved.
[7:0]	**CCBDAT16**	**TK16 Complement CB Data** This is register is used for TK16 sensitivity adjustment.

Touch Key Idle State Control Register (TK_IDLESEL)

Register	Offset	R/W	Description	Reset Value
TK_IDLESEL	TK_BA+0x1C	R/W	Touch Key Idle State Control Register	0xFFFF_FFFF

31	30	29	28	27	26	25	24
IDLS15		IDLS14		IDLS13		IDLS12	
23	22	21	20	19	18	17	16
IDLS11		IDLS10		IDLS9		IDLS8	
15	14	13	12	11	10	9	8
IDLS7		IDLS6		IDLS5		IDLS4	
7	6	5	4	3	2	1	0
IDLS3		IDLS2		IDLS1		IDLS0	

Bits	Description	
[31:0]	**IDLSn**	**TKn Idle State Control** This register is ignored if both TKSENn (TK_CTL[n]) and POLENn (TK_POLCTL[n+8]) are “0” or TKRENn (TK_REFCTL[n]) is “1”. 00 = TKn connected to Gnd. 01 = TKn connected to AVCCH. 10 = TKn connected to V_{DD}. 11 = TKn connected to V_{DD}. n = 0 to 15.

Touch Key Polarity Select Register (TK_POLSEL)

Register	Offset	R/W	Description	Reset Value
TK_POLSEL	TK_BA+0x20	R/W	Touch Key Polarity Select Register	0xFFFF_FFFF

31	30	29	28	27	26	25	24
POLSEL15		POLSEL14		POLSEL13		POLSEL12	
23	22	21	20	19	18	17	16
POLSEL11		POLSEL10		POLSEL9		POLSEL8	
15	14	13	12	11	10	9	8
POLSEL7		POLSEL6		POLSEL5		POLSEL4	
7	6	5	4	3	2	1	0
POLSEL3		POLSEL2		POLSEL1		POLSEL0	

Bits	Description	
[31:0]	**POLSELn**	**TKn Polarity Select** This register is ignored if POLENn (TK_POLCTL[n+8]) is “0”, or either TKSENn (TK_CTL[n]) or TKRENn (TK_REFCTL[n]) is “1”. 00 = TKn connected to Gnd. 01 = TKn connected to AVCCH. 10 = TKn connected to V_{DD}. 11 = TKn connected to V_{DD}.

Touch Key Polarity Control Register (TK_POLCTL)

Register	Offset	R/W	Description	Reset Value
TK_POLCTL	TK_BA+0x24	R/W	Touch Key Polarity Control Register	0x0000_001F

31	30	29	28	27	26	25	24
SPOTINIT	Reserved						POLEN16
23	22	21	20	19	18	17	16
POLEN15	POLEN14	POLEN13	POLEN12	POLEN11	POLEN10	POLEN9	POLEN8
15	14	13	12	11	10	9	8
POLEN7	POLEN6	POLEN5	POLEN4	POLEN3	POLEN2	POLEN1	POLEN0
7	6	5	4	3	2	1	0
Reserved		CBPOLSEL		POLSEL16		IDLS16	

Bits	Description	
[31]	**SPOTINIT**	**Touch Key Sensing Initial Potential Control** 0 = Key pad is connected to Gnd before sensing. 1 = Key pad is connected to AVCCH before sensing.
[30:25]	**Reserved**	Reserved.
[24]	**POLEN16**	**TK16 Polarity Function Enable Coontrol** 0 = TK16 polarity function Disabled. 1 = TK16 polarity function Enabled.
[23]	**POLEN15**	**TK15 Polarity Function Enable Coontrol** 0 = TK15 polarity function Disabled. 1 = TK15 polarity function Enabled.
[22]	**POLEN14**	**TK14 Polarity Function Enable Coontrol** 0 = TK14 polarity function Disabled. 1 = TK14 polarity function Enabled.
[21]	**POLEN13**	**TK13 Polarity Function Enable Coontrol** 0 = TK13 polarity function Disabled. 1 = TK13 polarity function Enabled.
[20]	**POLEN12**	**TK12 Polarity Function Enable Coontrol** 0 = TK12 polarity function Disabled. 1 = TK12 polarity function Enabled.
[19]	**POLEN11**	**TK11 Polarity Function Enable Coontrol** 0 = TK11 polarity function Disabled. 1 = TK11 polarity function Enabled.
[18]	**POLEN10**	**TK10 Polarity Function Enable Coontrol** 0 = TK10 polarity function Disabled. 1 = TK10 polarity function Enabled.

Bits	Description	
[17]	**POLEN9**	**TK9 Polarity Function Enable Coontrol** 0 = TK9 polarity function Disabled. 1 = TK9 polarity function Enabled.
[16]	**POLEN8**	**TK8 Polarity Function Enable Coontrol** 0 = TK8 polarity function Disabled. 1 = TK8 polarity function Enabled.
[15]	**POLEN7**	**TK7 Polarity Function Enable Coontrol** 0 = TK7 polarity function Disabled. 1 = TK7 polarity function Enabled.
[14]	**POLEN6**	**TK6 Polarity Function Enable Coontrol** 0 = TK6 polarity function Disabled. 1 = TK6 polarity function Enabled.
[13]	**POLEN5**	**TK5 Polarity Function Enable Coontrol** 0 = TK5 polarity function Disabled. 1 = TK5 polarity function Enabled.
[12]	**POLEN4**	**TK4 Polarity Function Enable Coontrol** 0 = TK4 polarity function Disabled. 1 = TK4 polarity function Enabled.
[11]	**POLEN3**	**TK3 Polarity Function Enable Coontrol** 0 = TK3 polarity function Disabled. 1 = TK3 polarity function Enabled.
[10]	**POLEN2**	**TK2 Polarity Function Enable Coontrol** 0 = TK2 polarity function Disabled. 1 = TK2 polarity function Enabled.
[9]	**POLEN1**	**TK1 Polarity Function Enable Coontrol** 0 = TK1 polarity function Disabled. 1 = TK1 polarity function Enabled.
[8]	**POLEN0**	**TK0 Polarity Function Enable Coontrol** 0 = TK0 polarity function Disabled. 1 = TK0 polarity function Enabled.
[7:6]	**Reserved**	Reserved.
[5:4]	**CBPOLSEL**	**Capacitor Bank Polarity Select** 00 = Gnd. 01 = AVCCH. 10 = V_{DD}. 11 = V_{DD}.
[3:2]	**POLSEL16**	**TK16 Polarity Control** This register is ignored if POLEN16 (TK_POLCTL[24]) is “0”, or either TKSEN16 (TK_CTL[16]) or TKREN16 (TK_REFCTL[16]) is “1”. 00 = TK16 connected to Gnd. 01 = TK16 connected to AVCCH. 10 = TK16 connected to V_{DD}. 11 = TK16 connected to V_{DD}.

Bits	Description	
[1:0]	**IDLS16**	**TK16 Idle State Control** This register is ignored if both TKSEN16 (TK_CTL[16]) and POLEN16 (TK_POLCTL[24]) are “0” or TKREN16 (TK_REFCTL[16]) is “1”. 00 = TK16 connected to Gnd. 01 = TK16 connected to AVCCH. 10 = TK16 connected to V_{DD}. 11 = TK16 connected to V_{DD}.

Touch Key Status Register (TK_STATUS)

Register	Offset	R/W	Description	Reset Value
TK_STATUS	TK_BA+0x28	R/W	Touch Key Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							TKIF16
23	22	21	20	19	18	17	16
TKIF15	TKIF14	TKIF13	TKIF12	TKIF11	TKIF10	TKIF9	TKIF8
15	14	13	12	11	10	9	8
TKIF7	TKIF6	TKIF5	TKIF4	TKIF3	TKIF2	TKIF1	TKIF0
7	6	5	4	3	2	1	0
Reserved						SCIF	BUSY

Bits	Description	
[31:25]	**Reserved**	Reserved.
[24]	**TKIF16**	**TK16 Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = No threshold control event with TK16. 1 = Threshold control event occurs with TK16.
[23]	**TKIF15**	**TK15 Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = No threshold control event with TK15. 1 = Threshold control event occurs with TK15.
[22]	**TKIF14**	**TK14 Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = No threshold control event with TK14. 1 = Threshold control event occurs with TK14.
[21]	**TKIF13**	**TK13 Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = No threshold control event with TK13. 1 = Threshold control event occurs with TK13.
[20]	**TKIF12**	**TK12 Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = No threshold control event with TK12. 1 = Threshold control event occurs with TK12.
[19]	**TKIF11**	**TK11 Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = No threshold control event with TK11. 1 = Threshold control event occurs with TK11.

Bits	Description	
[18]	**TKIF10**	**TK10 Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = No threshold control event with TK10. 1 = Threshold control event occurs with TK10.
[17]	**TKIF9**	**TK9 Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = No threshold control event with TK9. 1 = Threshold control event occurs with TK9.
[16]	**TKIF8**	**TK8 Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = No threshold control event with TK8. 1 = Threshold control event occurs with TK8.
[15]	**TKIF7**	**TK7 Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = No threshold control event with TK7. 1 = Threshold control event occurs with TK7.
[14]	**TKIF6**	**TK6 Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = No threshold control event with TK6. 1 = Threshold control event occurs with TK6.
[13]	**TKIF5**	**TK5 Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = No threshold control event with TK5. 1 = Threshold control event occurs with TK5.
[12]	**TKIF4**	**TK4 Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = No threshold control event with TK4. 1 = Threshold control event occurs with TK4.
[11]	**TKIF3**	**TK3 Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = No threshold control event with TK3. 1 = Threshold control event occurs with TK3.
[10]	**TKIF2**	**TK2 Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = No threshold control event with TK2. 1 = Threshold control event occurs with TK2.
[9]	**TKIF1**	**TK1 Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = No threshold control event with TK1. 1 = Threshold control event occurs with TK1.

Bits	Description	
[8]	**TKIF0**	**TK0 Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = No threshold control event with TK0. 1 = Threshold control event occurs with TK0.
[7:2]	**Reserved**	Reserved.
[1]	**SCIF**	**Touch Key Scan Complete Interrupt Flag** This bit will be cleared by writing a “1” to this bit. 0 = Key scan is proceeding and data is not ready for read. 1 = Key scan is complete and data is ready for read in TKDATx registers. **Note1:** The Touch Key interrupt asserts if SCINTEN bit of TK_INTEN register is set. **Note2:** The Touch Key interrupt also asserts if SCTHIEN bit of TK_INTEN register is set and any channel data value is greater/less than its threshold setting.
[0]	**BUSY**	**Touch Key Busy (Read Only)** 0 = Key scan is complete or stopped. 1 = Key scan is proceeding.

Touch Key Data Register 0 (TK_DAT0)

Register	Offset	R/W	Description	Reset Value
TK_DAT0	TK_BA+0x2C	R	Touch Key Data Register 0	0x0000_0000

31	30	29	28	27	26	25	24
TKDAT3							
23	22	21	20	19	18	17	16
TKDAT2							
15	14	13	12	11	10	9	8
TKDAT1							
7	6	5	4	3	2	1	0
TKDAT0							

Bits	Description	
[31:24]	**TKDAT3**	**TK3 Sensing Result Data (Read Only)** This data is invalid if TKSEN3 (TK_CTL[3]) is “0” or TKREN3 (TK_REFCTL[3]) is “1”.
[23:16]	**TKDAT2**	**TK2 Sensing Result Data (Read Only)** This data is invalid if TKSEN2 (TK_CTL[2]) is “0” or TKREN2 (TK_REFCTL[2]) is “1”.
[15:8]	**TKDAT1**	**TK1 Sensing Result Data (Read Only)** This data is invalid if TKSEN1 (TK_CTL[1]) is “0” or TKREN1 (TK_REFCTL[1]) is “1”.
[7:0]	**TKDAT0**	**TK0 Sensing Result Data (Read Only)** This data is invalid if TKSEN0 (TK_CTL[0]) is “0” or TKREN0 (TK_REFCTL[0]) is “1” except SCANALL (TK_REFCTL[23]) is “1”.

Touch Key Data Register 1 (TK_DAT1)

Register	Offset	R/W	Description	Reset Value
TK_DAT1	TK_BA+0x30	R	Touch Key Data Register 1	0x0000_0000

31	30	29	28	27	26	25	24
TKDAT7							
23	22	21	20	19	18	17	16
TKDAT6							
15	14	13	12	11	10	9	8
TKDAT5							
7	6	5	4	3	2	1	0
TKDAT4							

Bits	Description	
[31:24]	**TKDAT7**	**TK7 Sensing Result Data (Read Only)** This data is invalid if TKSEN7 (TK_CTL[7]) is “0” or TKREN7 (TK_REFCTL[7]) is “1”.
[23:16]	**TKDAT6**	**TK6 Sensing Result Data (Read Only)** This data is invalid if TKSEN6 (TK_CTL[6]) is “0” or TKREN6 (TK_REFCTL[6]) is “1”.
[15:8]	**TKDAT5**	**TK5 Sensing Result Data (Read Only)** This data is invalid if TKSEN5 (TK_CTL[5]) is “0” or TKREN5 (TK_REFCTL[5]) is “1”.
[7:0]	**TKDAT4**	**TK0 Sensing Result Data (Read Only)** This data is invalid if TKSEN4 (TK_CTL[4]) is “0” or TKREN4 (TK_REFCTL[4]) is “1”.

Touch Key Data Register 2 (TK_DAT2)

Register	Offset	R/W	Description	Reset Value
TK_DAT2	TK_BA+0x34	R	Touch Key Data Register 2	0x0000_0000

31	30	29	28	27	26	25	24
TKDAT11							
23	22	21	20	19	18	17	16
TKDAT10							
15	14	13	12	11	10	9	8
TKDAT9							
7	6	5	4	3	2	1	0
TKDAT8							

Bits	Description	
[31:24]	**TKDAT11**	**TK11 Sensing Result Data (Read Only)** This data is invalid if TKSEN11 (TK_CTL[11]) is “0” or TKREN11 (TK_REFCTL[11]) is “1”.
[23:16]	**TKDAT10**	**TK10 Sensing Result Data (Read Only)** This data is invalid if TKSEN10 (TK_CTL[10]) is “0” or TKREN10 (TK_REFCTL[10]) is “1”.
[15:8]	**TKDAT9**	**TK9 Sensing Result Data (Read Only)** This data is invalid if TKSEN9 (TK_CTL[9]) is “0” or TKREN9 (TK_REFCTL[9]) is “1”.
[7:0]	**TKDAT8**	**TK8 Sensing Result Data (Read Only)** This data is invalid if TKSEN8 (TK_CTL[8]) is “0” or TKREN8 (TK_REFCTL[8]) is “1”.

Touch Key Data Register 3 (TK_DAT3)

Register	Offset	R/W	Description	Reset Value
TK_DAT3	TK_BA+0x38	R	Touch Key Data Register 3	0x0000_0000

31	30	29	28	27	26	25	24
TKDAT15							
23	22	21	20	19	18	17	16
TKDAT14							
15	14	13	12	11	10	9	8
TKDAT13							
7	6	5	4	3	2	1	0
TKDAT12							

Bits	Description	
[31:24]	**TKDAT15**	**TK15 Sensing Result Data (Read Only)** This data is invalid if TKSEN15 (TK_CTL[15]) is “0” or TKREN15 (TK_REFCTL[15]) is “1”.
[23:16]	**TKDAT14**	**TK14 Sensing Result Data (Read Only)** This data is invalid if TKSEN14 (TK_CTL[14]) is “0” or TKREN14 (TK_REFCTL[14]) is “1”.
[15:8]	**TKDAT13**	**TK13 Sensing Result Data (Read Only)** This data is invalid if TKSEN13 (TK_CTL[13]) is “0” or TKREN13 (TK_REFCTL[13]) is “1”.
[7:0]	**TKDAT12**	**TK12 Sensing Result Data (Read Only)** This data is invalid if TKSEN12 (TK_CTL[12]) is “0” or TKREN12 (TK_REFCTL[12]) is “1”.

Touch Key Data Register 4 (TK_DAT4)

Register	Offset	R/W	Description	Reset Value
TK_DAT4	TK_BA+0x3C	R	Touch Key Data Register 4	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
TKDAT16							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**TKDAT16**	**TK16 Sensing Result Data (Read Only)** This data is invalid if TKSEN16 (TK_CTL[16]) is “0” or TKREN16 (TK_REFCTL[16]) is “1”.

Touch Key Interrupt Enable Register (TK_INTEN)

Register	Offset	R/W	Description	Reset Value
TK_INTEN	TK_BA+0x40	R/W	Touch Key Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
THIMOD	Reserved						
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						SCINTEN	SCTHIEN

Bits	Description	
[31]	**THIMOD**	**Touch Key Threshold Interrupt Mode Select** 0 = Edge trigger mode. 1 = Level trigger mode.
[30:2]	**Reserved**	Reserved.
[1]	**SCINTEN**	**Touch Key Scan Complete Interrupt Enable Bit** 0 = Key scan complete without threshold control interrupt is disable. 1 = Key scan complete without threshold control interrupt is enable.
[0]	**SCTHIEN**	**Touch Key Scan Complete with High/Low Threshold Control Interrupt Enable Bit** 0 = Key scan complete with threshold control interrupt is disable. 1 = Key scan complete with threshold control interrupt is enable.

Touch Key TK0/TK1 Threshold Control Register (TK_TH0_1)

Register	Offset	R/W	Description	Reset Value
TK_TH0_1	TK_BA+0x44	R/W	Touch Key TK0/TK1 Threshold Control Register	0xFF00_FF00

31	30	29	28	27	26	25	24
HTH1							
23	22	21	20	19	18	17	16
LTH1							
15	14	13	12	11	10	9	8
HTH0							
7	6	5	4	3	2	1	0
LTH0							

Bits	Description	
[31:24]	**HTH1**	**High Threshold of TK1** High level for TK1 threshold control.
[23:16]	**LTH1**	**Low Threshold of TK1** Low level for TK1 threshold control.
[15:8]	**HTH0**	**High Threshold of TK0** High level for TK0 threshold control.
[7:0]	**LTH0**	**Low Threshold of TK0** Low level for TK0 threshold control.

Touch Key TK2/TK3 Threshold Control Register (TK_TH2_3)

Register	Offset	R/W	Description	Reset Value
TK_TH2_3	TK_BA+0x48	R/W	Touch Key TK2/TK3 Threshold Control Register	0xFF00_FF00

31	30	29	28	27	26	25	24
HTH3							
23	22	21	20	19	18	17	16
LTH3							
15	14	13	12	11	10	9	8
HTH2							
7	6	5	4	3	2	1	0
LTH2							

Bits	Description	
[31:24]	**HTH3**	**High Threshold of TK3** High level for TK3 threshold control.
[23:16]	**LTH3**	**Low Threshold of TK3** Low level for TK3 threshold control.
[15:8]	**HTH2**	**High Threshold of TK2** High level for TK2 threshold control.
[7:0]	**LTH2**	**Low Threshold of TK2** Low level for TK2 threshold control.

Touch Key TK4/TK5 Threshold Control Register (TK_TH4_5)

Register	Offset	R/W	Description	Reset Value
TK_TH4_5	TK_BA+0x4C	R/W	Touch Key TK4/TK5 Threshold Control Register	0xFF00_FF00

31	30	29	28	27	26	25	24
HTH5							
23	22	21	20	19	18	17	16
LTH5							
15	14	13	12	11	10	9	8
HTH4							
7	6	5	4	3	2	1	0
LTH4							

Bits	Description	
[31:24]	**HTH5**	**High Threshold of TK5** High level for TK5 threshold control.
[23:16]	**LTH5**	**Low Threshold of TK5** Low level for TK5 threshold control.
[15:8]	**HTH4**	**High Threshold of TK4** High level for TK4 threshold control.
[7:0]	**LTH4**	**Low Threshold of TK4** Low level for TK4 threshold control.

Touch Key TK6/TK7 Threshold Control Register (TK_TH6_7)

Register	Offset	R/W	Description	Reset Value
TK_TH6_7	TK_BA+0x50	R/W	Touch Key TK6/TK7 Threshold Control Register	0xFF00_FF00

31	30	29	28	27	26	25	24
HTH7							
23	22	21	20	19	18	17	16
LTH7							
15	14	13	12	11	10	9	8
HTH6							
7	6	5	4	3	2	1	0
LTH6							

Bits	Description	
[31:24]	**HTH7**	**High Threshold of TK7** High level for TK7 threshold control.
[23:16]	**LTH7**	**Low Threshold of TK7** Low level for TK7 threshold control.
[15:8]	**HTH6**	**High Threshold of TK6** High level for TK6 threshold control.
[7:0]	**LTH6**	**Low Threshold of TK6** Low level for TK6 threshold control.

Touch Key TK8/TK9 Threshold Control Register (TK_TH8_9)

Register	Offset	R/W	Description	Reset Value
TK_TH8_9	TK_BA+0x54	R/W	Touch Key TK8/TK9 Threshold Control Register	0xFF00_FF00

31	30	29	28	27	26	25	24
HTH9							
23	22	21	20	19	18	17	16
LTH9							
15	14	13	12	11	10	9	8
HTH8							
7	6	5	4	3	2	1	0
LTH8							

Bits	Description	
[31:24]	**HTH9**	**High Threshold of TK9** High level for TK9 threshold control.
[23:16]	**LTH9**	**Low Threshold of TK9** Low level for TK9 threshold control.
[15:8]	**HTH8**	**High Threshold of TK8** High level for TK8 threshold control.
[7:0]	**LTH8**	**Low Threshold of TK8** Low level for TK8 threshold control.

Touch Key TK10/TK11 Threshold Control Register (TK_TH10_11)

Register	Offset	R/W	Description	Reset Value
TK_TH10_11	TK_BA+0x58	R/W	Touch Key TK10/TK11 Threshold Control Register	0xFF00_FF00

31	30	29	28	27	26	25	24
HTH11							
23	22	21	20	19	18	17	16
LTH11							
15	14	13	12	11	10	9	8
HTH10							
7	6	5	4	3	2	1	0
LTH10							

Bits	Description	
[31:24]	**HTH11**	**High Threshold of TK11** High level for TK11 threshold control.
[23:16]	**LTH11**	**Low Threshold of TK11** Low level for TK11 threshold control.
[15:8]	**HTH10**	**High Threshold of TK10** High level for TK10 threshold control.
[7:0]	**LTH10**	**Low Threshold of TK10** Low level for TK10 threshold control.

Touch Key TK12/TK13 Threshold Control Register (TK_TH12_13)

Register	Offset	R/W	Description	Reset Value
TK_TH12_13	TK_BA+0x5C	R/W	Touch Key TK12/TK13 Threshold Control Register	0xFF00_FF00

31	30	29	28	27	26	25	24
HTH13							
23	22	21	20	19	18	17	16
LTH13							
15	14	13	12	11	10	9	8
HTH12							
7	6	5	4	3	2	1	0
LTH12							

Bits	Description	
[31:24]	**HTH13**	**High Threshold of TK13** High level for TK13 threshold control.
[23:16]	**LTH13**	**Low Threshold of TK13** Low level for TK13 threshold control.
[15:8]	**HTH12**	**High Threshold of TK12** High level for TK12 threshold control.
[7:0]	**LTH12**	**Low Threshold of TK12** Low level for TK12 threshold control.

Touch Key TK14/TK15 Threshold Control Register (TK_TH14_15)

Register	Offset	R/W	Description	Reset Value
TK_TH14_15	TK_BA+0x60	R/W	Touch Key TK14/TK15 Threshold Control Register	0xFF00_FF00

31	30	29	28	27	26	25	24
HTH15							
23	22	21	20	19	18	17	16
LTH15							
15	14	13	12	11	10	9	8
HTH14							
7	6	5	4	3	2	1	0
LTH14							

Bits	Description	
[31:24]	**HTH15**	**High Threshold of TK15** High level for TK15 threshold control.
[23:16]	**LTH15**	**Low Threshold of TK15** Low level for TK15 threshold control.
[15:8]	**HTH14**	**High Threshold of TK14** High level for TK14 threshold control.
[7:0]	**LTH14**	**Low Threshold of TK14** Low level for TK14 threshold control.

Touch Key TK16 Threshold Control Register (TK_TH16)

Register	Offset	R/W	Description	Reset Value
TK_TH16	TK_BA+0x64	R/W	Touch Key TK16 Threshold Control Register	0x0000_FF00

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
HTH16							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LTH16							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**HTH16**	**High Threshold of TK16** High level for TK16 threshold control.
[7:0]	**LTH16**	**Low Threshold of TK16** Low level for TK16 threshold control.

6.22 CRC Controller (CRC)

6.22.1 Overview

The Cyclic Redundancy Check (CRC) generator can perform CRC calculation with programmable polynomial settings.

6.22.2 Features

- Supports four common polynomials CRC-CCITT, CRC-8, CRC-16, and CRC-32
  - CRC-CCITT: $X^{16} + X^{12} + X^5 + 1$
  - CRC-8: $X^8 + X^2 + X + 1$
  - CRC-16: $X^{16} + X^{15} + X^2 + 1$
  - CRC-32: $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
- Programmable seed value
- Supports programmable order reverse setting for input data and CRC checksum
- Supports programmable 1's complement setting for input data and CRC checksum
- Supports 8/16/32-bit of data width
  - 8-bit write mode: 1-AHB clock cycle operation
  - 16-bit write mode: 2-AHB clock cycle operation
  - 32-bit write mode: 4-AHB clock cycle operation
- Supports using PDMA to write data to perform CRC operation

6.22.3 Block Diagram

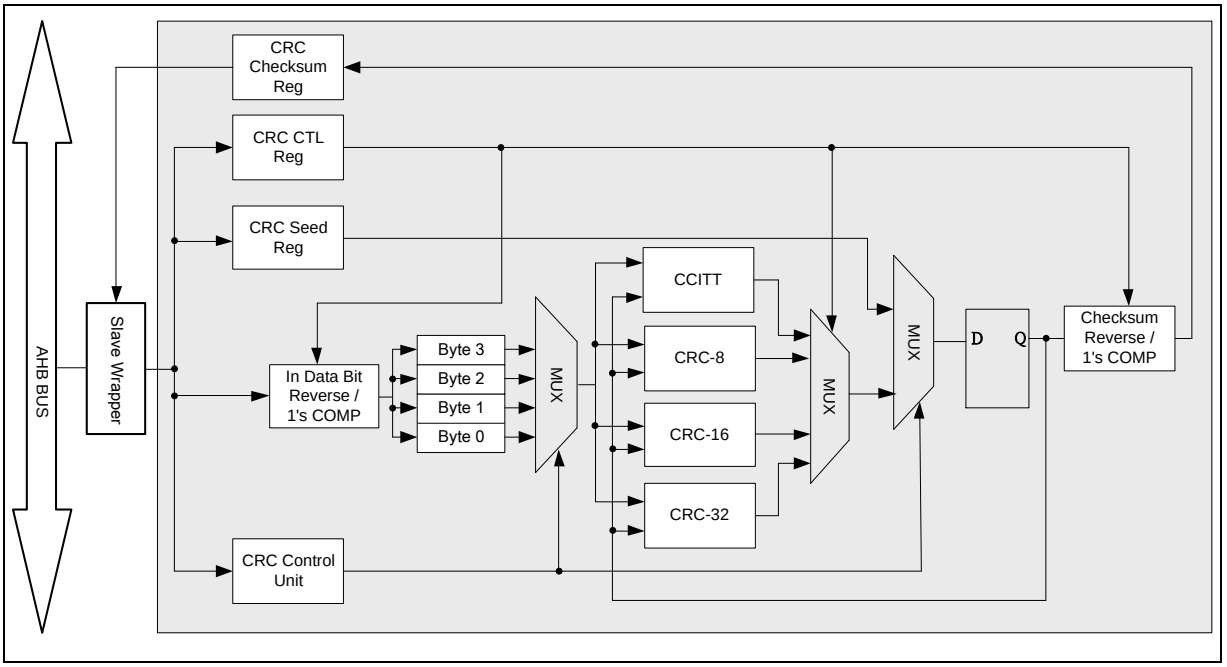


Figure 6.22-1 CRC Generator Block Diagram

6.22.4 Basic Configuration

The CRC peripheral clock is enabled in CRCCKEN (CLK_AHBCLK[7]). After CRC is setting, user can start to perform CRC calculate by control CRC's registers.

6.22.5 Functional Description

CRC generator can perform CRC calculation with programmable polynomial settings. The operation polynomial includes CRC-CCITT, CRC-8, CRC-16 and CRC-32; User can choose the CRC operation polynomial mode by setting CRCMODE[1:0] (CRC_CTL[31:30] CRC Polynomial Mode).

The following is a program sequence example.

1. Enable CRC generator by setting CRCEN (CRC_CTL[0] CRC Channel Enable Control).
2. Initial setting for CRC calculation.
   - Configure 1's complement for CRC checksum by setting CHKSFMT (CRC_CTL[27] Checksum Complement).
   - Configure bit order reverse for CRC checksum by setting CHKSREV (CRC_CTL[25] Checksum Bit Order Reverse).
   - Configure 1's complement for CRC write data by setting DATFMT (CRC_CTL[26] Write Data Complement).
   - Configure bit order reverse for CRC write data by setting DATREV (CRC_CTL[24] Write Data Bit Order Reverse).
3. Perform CRC reset to load the initial seed value to CRC circuit by setting CRCRST (CRC_CTL[1] CRC Engine Reset).
4. Write data to CRC_DAT register to calculate CRC checksum.
5. Get the CRC checksum result by reading CRC_CHECKSUM register.

6.22.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
CRC Base Address: **CRC_BA = 0x4003_1000**				
CRC_CTL	CRC_BA+0x00	R/W	CRC Control Register	0x2000_0000
CRC_DAT	CRC_BA+0x04	R/W	CRC Write Data Register	0x0000_0000
CRC_SEED	CRC_BA+0x08	R/W	CRC Seed Register	0xFFFF_FFFF
CRC_CHECKSUM	CRC_BA+0x0C	R	CRC Checksum Register	0x0000_0000

6.22.7 Register Description

CRC Control Register (CRC_CTL)

Register	Offset	R/W	Description	Reset Value
CRC_CTL	CRC_BA+0x00	R/W	CRC Control Register	0x2000_0000

31	30	29	28	27	26	25	24
CRCMODE		DATLEN		CHKSFMT	DATFMT	CHKSREV	DATREV
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						CRCRST	CRCEN

Bits	Description	
[31:30]	**CRCMODE**	**CRC Polynomial Mode** This field indicates the CRC operation polynomial mode. 00 = CRC-CCITT Polynomial mode. 01 = CRC-8 Polynomial mode. 10 = CRC-16 Polynomial mode. 11 = CRC-32 Polynomial mode.
[29:28]	**DATLEN**	**CPU Write Data Length** This field indicates the write data length. 00 = Data length is 8-bit mode. 01 = Data length is 16-bit mode. 1x = Data length is 32-bit mode. **Note:** When the write data length is 8-bit mode, the valid data in CRC_DAT register is only DATA[7:0] bits; if the write data length is 16-bit mode, the valid data in CRC_DAT register is only DATA[15:0].
[27]	**CHKSFMT**	**Checksum 1's Complement** This bit is used to enable the 1's complement function for checksum result in CRC_CHECKSUM register. 0 = 1's complement for CRC checksum Disabled. 1 = 1's complement for CRC checksum Enabled.
[26]	**DATFMT**	**Write Data 1's Complement** This bit is used to enable the 1's complement function for write data value in CRC_DAT register. 0 = 1's complement for CRC writes data in Disabled. 1 = 1's complement for CRC writes data in Enabled.

[25]	**CHKSREV**	**Checksum Bit Order Reverse** This bit is used to enable the bit order reverse function for write data value in CRC_CHECKSUM register. 0 = Bit order reverse for CRC checksum Disabled. 1 = Bit order reverse for CRC checksum Enabled. **Note:** If the checksum result is 0xDD7B0F2E, the bit order reverse for CRC checksum is 0x74F0DEBB.
[24]	**DATREV**	**Write Data Bit Order Reverse** This bit is used to enable the bit order reverse function for write data value in CRC_DAT register. 0 = Bit order reversed for CRC write data in Disabled. 1 = Bit order reversed for CRC write data in Enabled (per byte). **Note:** If the write data is 0xAABBCCDD, the bit order reverse for CRC write data is 0x55DD33BB.
[23:2]	**Reserved**	Reserved.
[1]	**CRCRST**	**CRC Engine Reset** 0 = No effect. 1 = Reset the internal CRC state machine. The others contents of CRC_CTL register will not be cleared. **Note1:** This bit will be cleared automatically. **Note2:** Setting this bit will reload the seed value from CRC_SEED register as checksum initial vale.
[0]	**CRCEN**	**CRC Channel Enable Bit** 0 = No effect. 1 = CRC operation Enabled.

CRC Write Data Register (CRC_DAT)

Register	Offset	R/W	Description	Reset Value
CRC_DAT	CRC_BA+0x04	R/W	CRC Write Data Register	0x0000_0000

31	30	29	28	27	26	25	24
DATA							
23	22	21	20	19	18	17	16
DATA							
15	14	13	12	11	10	9	8
DATA							
7	6	5	4	3	2	1	0
DATA							

Bits	Description	
[31:0]	DATA	**CRC Write Data Bits** User can write data directly by CPU mode or use PDMA function to write data to this field to perform CRC operation. **Note:** When the write data length is 8-bit mode, the valid data in CRC_DAT register is only DATA[7:0] bits; if the write data length is 16-bit mode, the valid data in CRC_DAT register is only DATA[15:0].

CRC Seed Register (CRC_SEED)

Register	Offset	R/W	Description	Reset Value
CRC_SEED	CRC_BA+0x08	R/W	CRC Seed Register	0xFFFF_FFFF

31	30	29	28	27	26	25	24
SEED							
23	22	21	20	19	18	17	16
SEED							
15	14	13	12	11	10	9	8
SEED							
7	6	5	4	3	2	1	0
SEED							

Bits	Description	
[31:0]	SEED	**CRC Seed Value** This field indicates the CRC seed value.

CRC Checksum Register (CRC_CHECKSUM)

Register	Offset	R/W	Description	Reset Value
CRC_CHECKSUM	CRC_BA+0x0C	R	CRC Checksum Register	0x0000_0000

31	30	29	28	27	26	25	24
CHECKSUM							
23	22	21	20	19	18	17	16
CHECKSUM							
15	14	13	12	11	10	9	8
CHECKSUM							
7	6	5	4	3	2	1	0
CHECKSUM							

Bits	Description	
[31:0]	CHECKSUM	**CRC Checksum Results** This field indicates the CRC checksum result.

6.23 Enhanced 12-bit Analog-to-Digital Converter (EADC)

6.23.1 Overview

The M4TK series contains one 12-bit successive approximation analog-to-digital converter (SAR A/D converter) with 16 external input channels and 3 internal channels. The A/D converter can be started by software trigger, PWM0/1 triggers, timer0~3 overflow pulse triggers, ADINT0, ADINT1 interrupt EOC (End of conversion) pulse trigger and external pin (STADC) input signal.

6.23.2 Features

- Analog input voltage range: 0~V_{REF} (Max to AV_{DD}).
- Reference voltage from V_{REF} pin or AV_{DD}.
- 12-bit resolution and 10-bit accuracy is guaranteed.
- Up to 16 single-end analog external input channels or 8 pair differential analog input channels.
- 3 internal channels, they are band-gap voltage (V_{BG}), temperature sensor (V_{TEMP}), and Battery power (V_{BAT})
- Four ADC interrupts (ADINT0~3) with individual interrupt vector addresses.
- Maximum ADC clock frequency is 20 MHz.
- Up to 1 Msps conversion rate.
- Configurable ADC internal sampling time.
- Up to 19 sample modules
  - Each of sample module 0~15 which is configurable for ADC converter channel EADC_CH0~15 and trigger source.
  - Sample module 16~18 is fixed for ADC channel 16, 17, 18 input sources as band-gap voltage, temperature sensor, and battery power (V_{BAT}).
  - Double buffer for sample module 0~3
  - Configurable sampling time for each sample module.
  - Conversion results are held in 19 data registers with valid and overrun indicators.
- An A/D conversion can be started by:
  - Write 1 to SWTRGn (EADC_SWTRG[n] , n = 0~18)
  - External pin STADC
  - Timer0~3 overflow pulse triggers
  - ADINT0 and ADINT1 interrupt EOC (End of conversion) pulse triggers
  - PWM triggers
- Supports PDMA transfer

6.23.3 Block Diagram

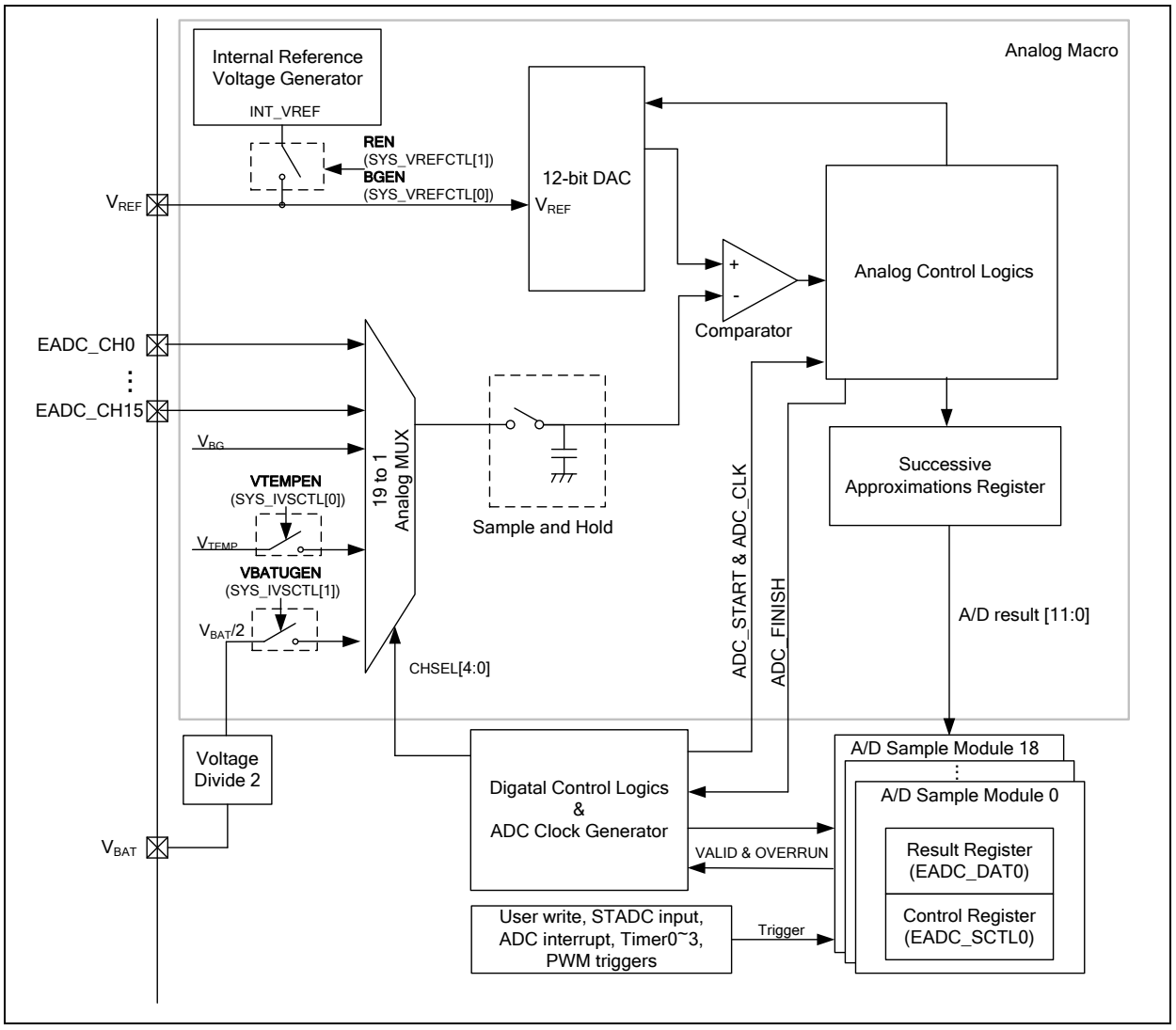


Figure 6.23-1 ADC Converter Block Diagram

6.23.4 Basic Configuration

The EADC pins are configured in SYS_GPB_MFPL and SYS_GPB_MFPH registers for M4TK. The External pin STADC is configured in SYS_GPD_MFPL register for M4TK. The ADC Controller clock source is enabled by EADCCKEN (CLK_APBCLK0[28]). After the EADC pins is configured to ADC analog input, DINOFF (PB_DINOFF[31:16] in GPIO register) should be set to 1 to disable digital input path.

6.23.5 Operation Procedure

The EADC controller consists of a 19 channel analog switch , 19 sample modules and a 12-bit successive approximation analog-to-digital converter. The EADC operation is based on sample

module 0~18, each of them has its configuration to decide which trigger source to start the conversion, which channel to convert. Sample module 0~15 can be configured to EADC_CH0~15 channel, and different trigger source. It provides user a flexible means to get the over-sampling results. The sample module 0~3 and sample module 4~15 are shows as Figure 6.23-2.

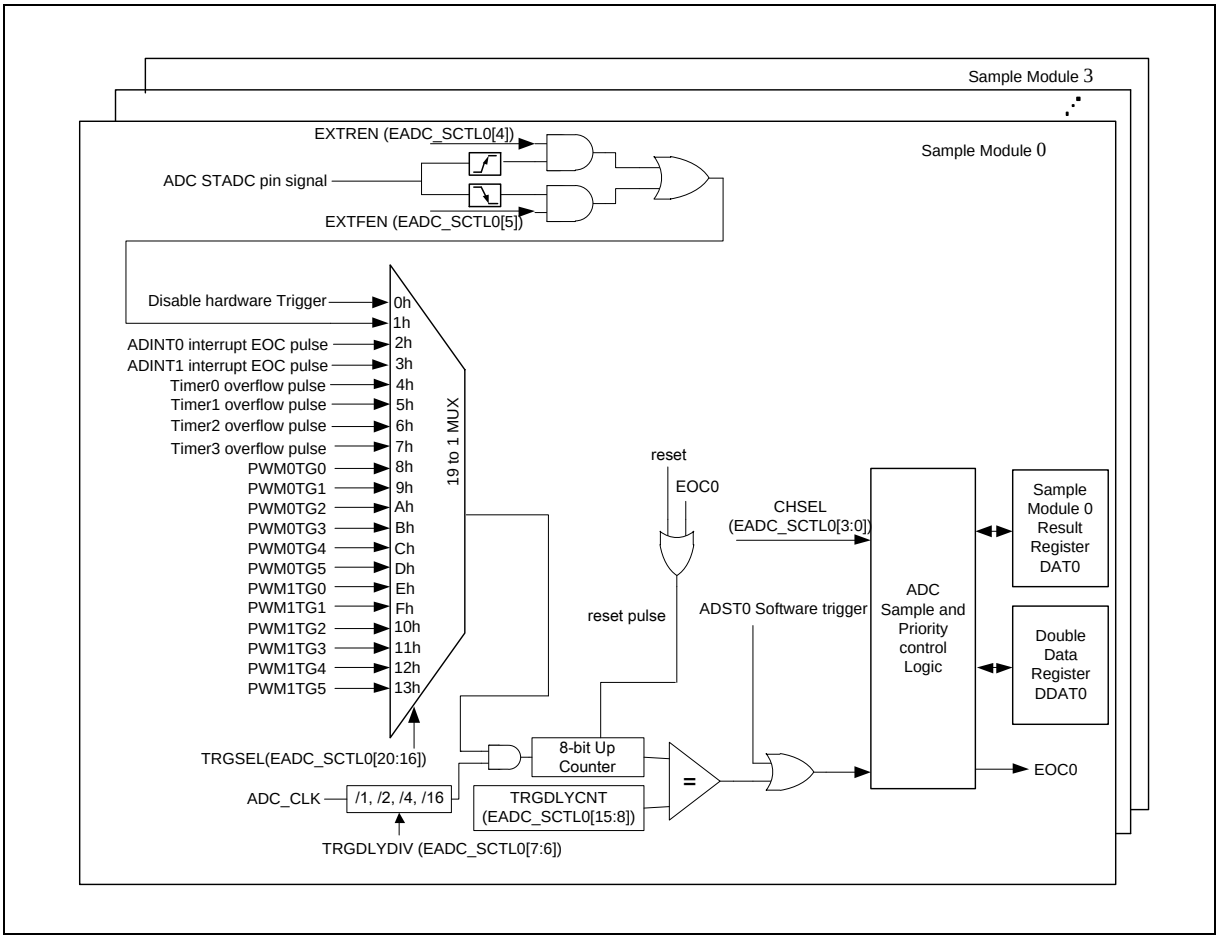


Figure 6.23-2 Sample Module 0~3 Block Diagram

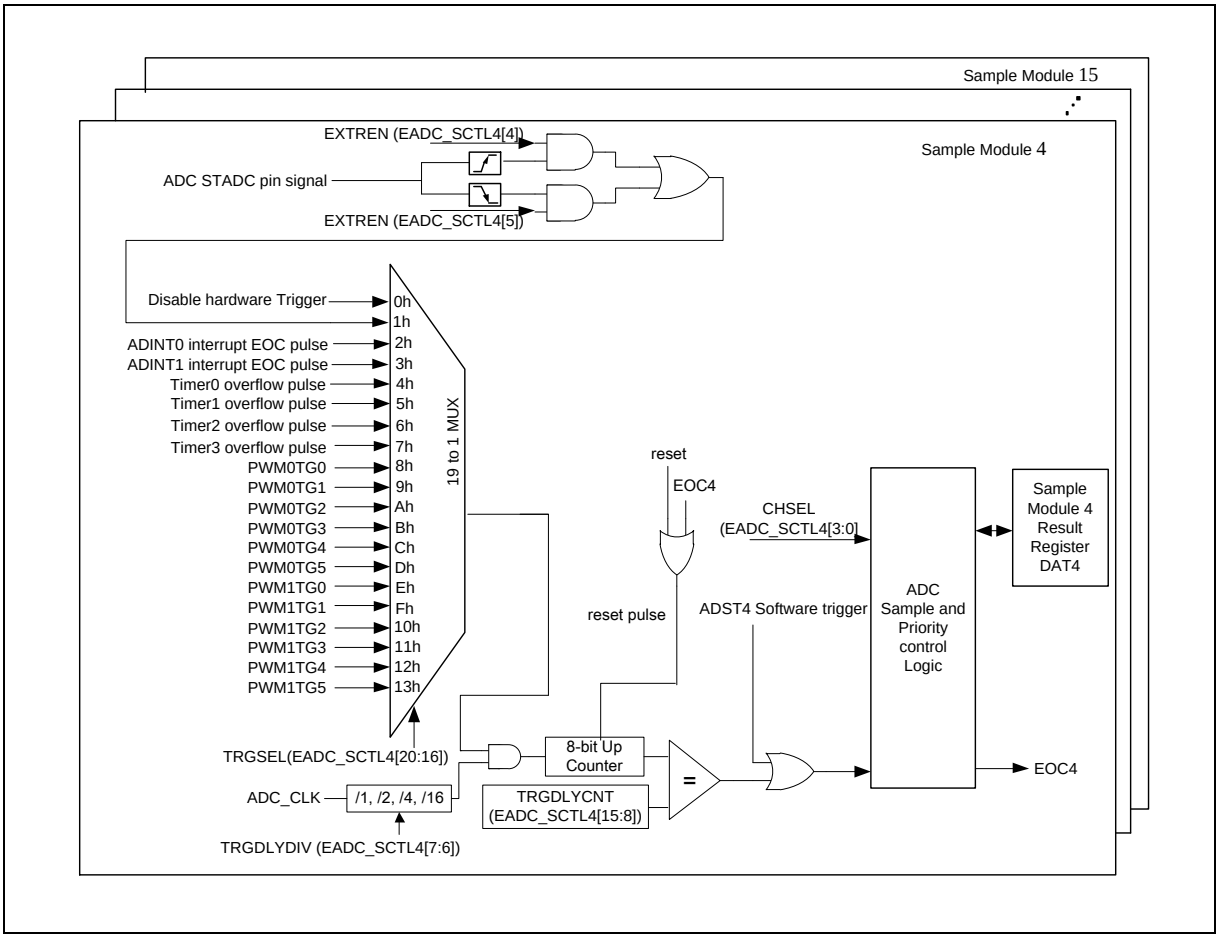


Figure 6.23-3 Sample Module 4~15 Block Diagram

Sample module 16~18 can convert internal channel (V_{BG}, V_{TEMP}, V_{BAT}) and can be triggered by user write SWTRGn (EADC_SWTRG[n], n = 16~18). The Figure 6.23-4 shows the sample module 16~18.

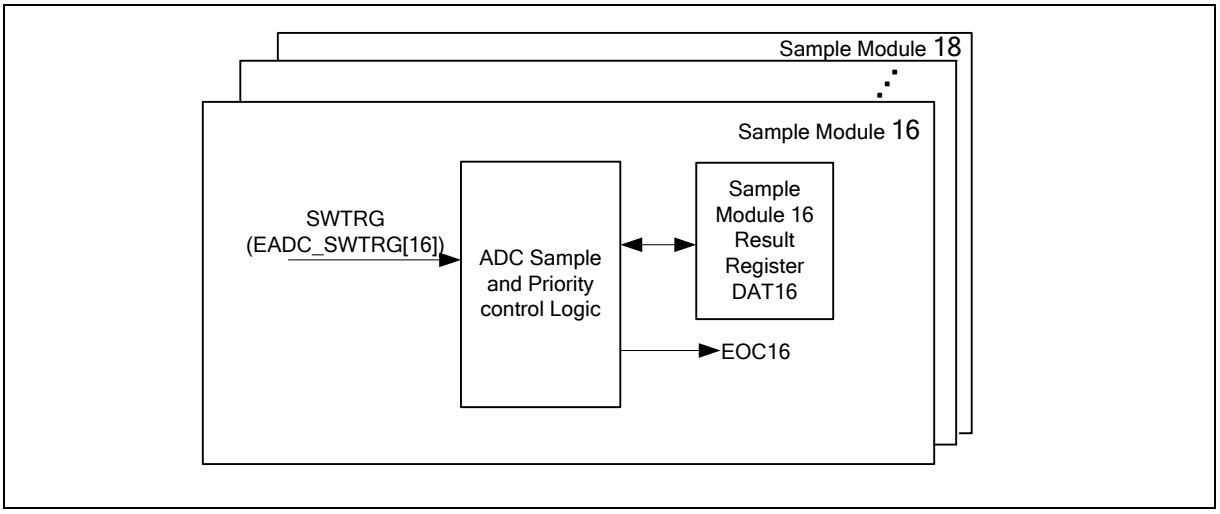


Figure 6.23-4 Sample Module 16~18 Block Diagram

The ADC conversion trigger sources in sample module 0~15 are listed below:

- Write 1 to SWTRGn (EADC_SWTRG[n], n = 0~15)
- External pin STADC
- Timer0~3 overflow pulse triggers
- ADINT0, ADINT1 ADC interrupt EOC (End of conversion) pulse triggers
- PWM triggers

The ADINT0 or ADINT1 interrupt pulses are generated whenever the specific sample module A/D EOC (End of conversion) pulse is generated. ADINT0 or ADINT1 interrupt pulse triggers can be fed back to trigger another A/D conversion, and is useful if a continuous scan conversion is needed.

6.23.5.1 ADC Clock Generator

The maximum EADC clock frequency is up to 20 MHz and the maximum sampling rate is up to 1Msps. It needs 20 EADC clocks to complete an A/D conversion by default setting. The EADC peripheral clock source is from PCLK1 clock, the ADC clock frequency is divided by an 8-bit pre-scalar with following formula:

The EADC clock frequency = (PCLK1) / (EADCDIV (CLK_CLKDIV0[23:16])+1).

The Figure 6.23-5 shows the EADC clock control.

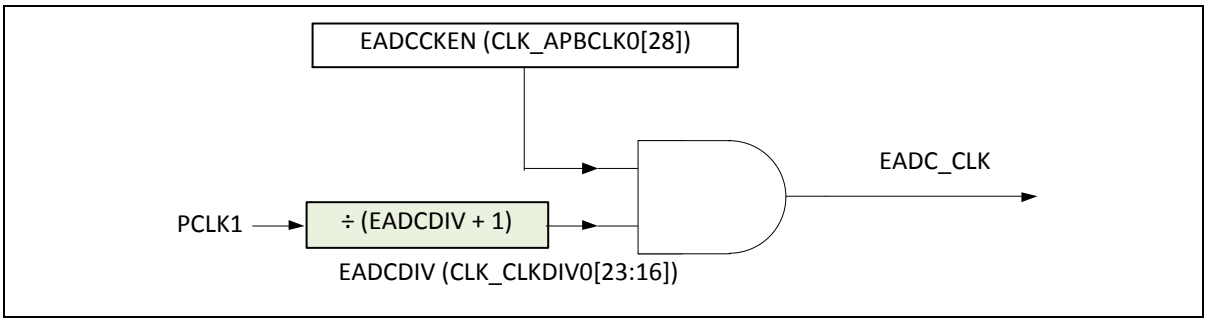


Figure 6.23-5 EADC Clock Control

6.23.5.2 ADC Software Trigger

When a ADC conversion is performed for specified single channel on the sample module, the operations are as follows:

1. A/D conversion is started when the SWTRGn (EADC_SWTRG[n], n=0~18) is set to 1 by user or other trigger inputs.
2. When A/D conversion is finished, the 12-bit result is stored in the ADC data register EADC_DATn (n=0~18) corresponding to the sample module.
3. Set SPLIEn (EADC_INTSRCm[n], n=0~18, m=0~3) to define which sample module affects ADIFm (EADC_STATUS2[m], m=0~3) flag.
4. On completion of conversion, the ADIFm (EADC_STATUS2[m], m=0~3) is set to 1 and ADC interrupt (ADINTm, m=0~3) is requested if the ADCIENm (EADC_CTL[5:2], m=0~3) bit is set to 1.
5. The SWTRGn (n=0~18) bit remains 1 during A/D conversion. When A/D conversion ends, the SWTRGn (n=0~18) bit is automatically cleared to 0 and the A/D converter will do another pending conversion.

If more than one sample module is enabled to conver analog signal, the sample module specified

channel with highest priority is firstly converted and other enabled sample module will be pended. The lower number sample module has higher priority. The sample module 0 is highest priority and the sample module 18 is lowest priority.

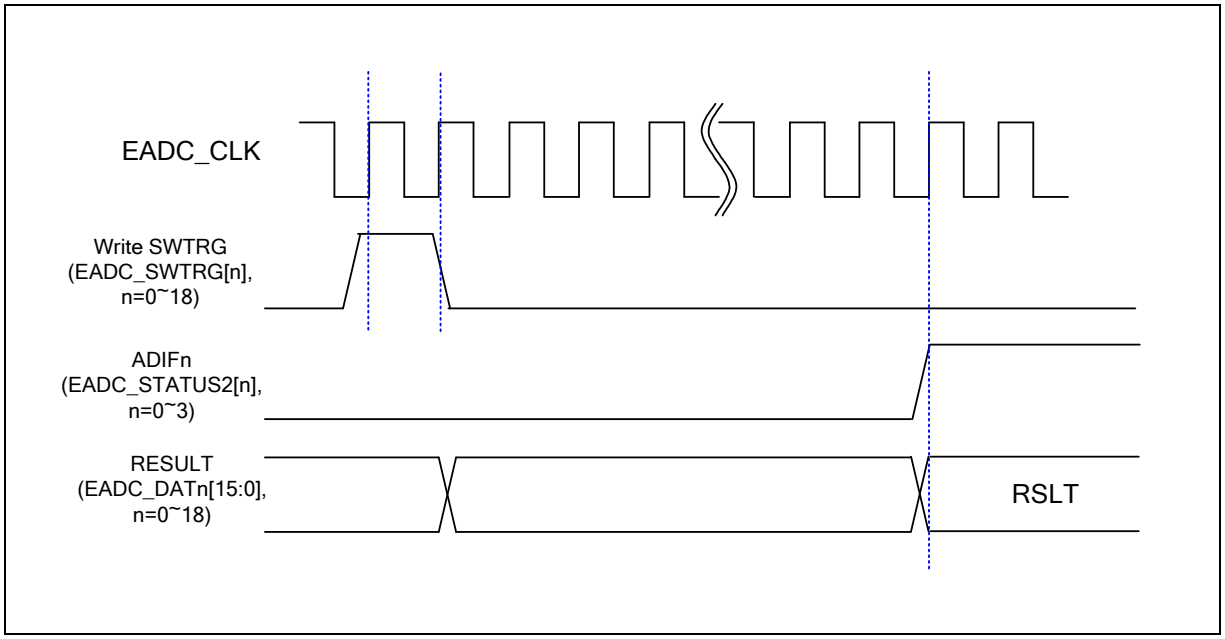


Figure 6.23-6 Example ADC Conversion Timing Diagram, n=0~18

6.23.5.3 ADC Conversion Priority

There is a priority group converter for determining the conversion order when multiple sample module trigger flags are set at the same time.

Sample module with lower number has higher priority than the higher number sample module, if two sample module are triggered at the same time, the sample module with lower number will start to convert ADC first.

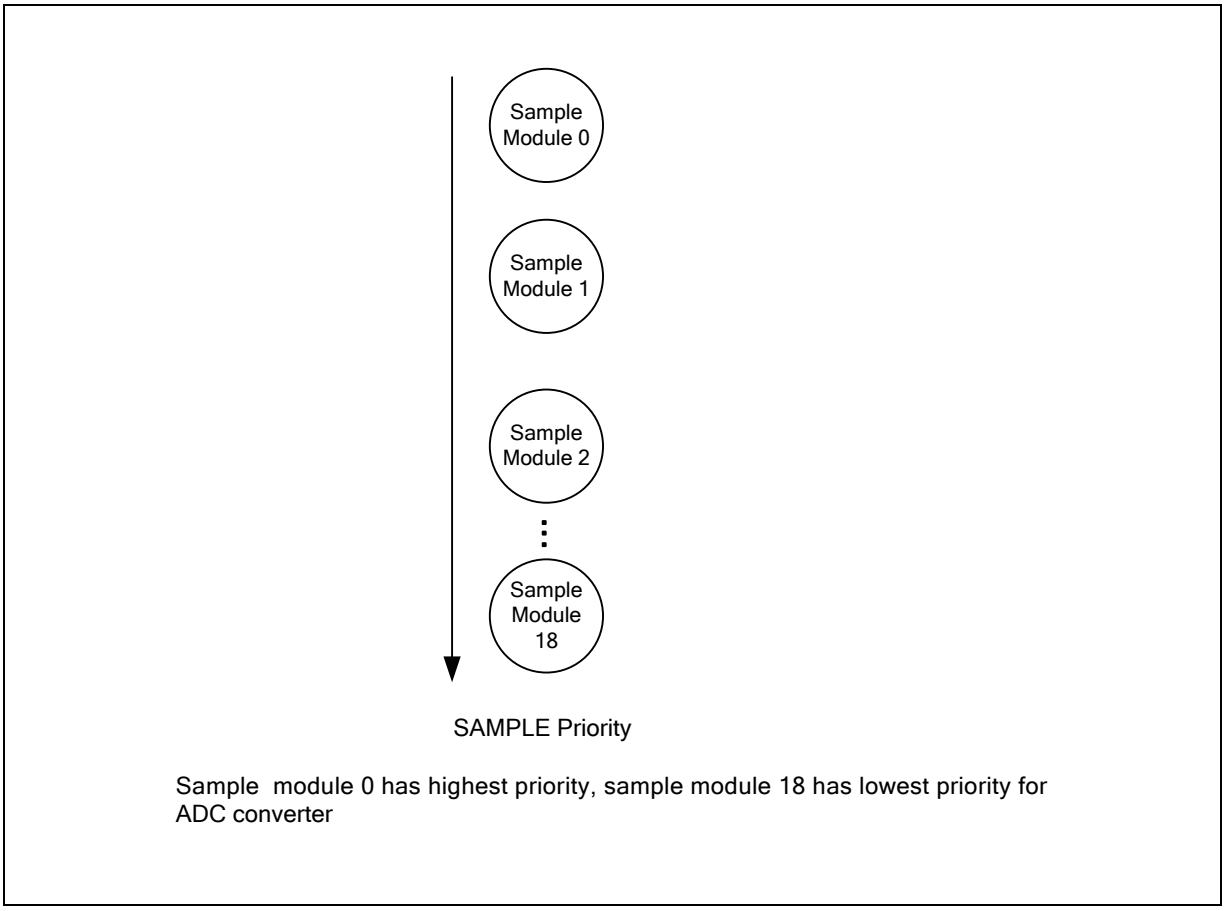


Figure 6.23-7 Sample module Conversion Priority Arbitrator Diagram

6.23.5.4 ADC Sample Module End of Conversion (EOC) Interrupt Operation

There are 4 ADC interrupts ADINT0~3, and each of these interrupts has its own interrupt vector address and ADINT0/ADINT1 can be configured to set multiple sample module EOC pulse (sample module 0~18 End of conversion pulses) as its interrupt trigger source.

When ADCIEN0 (EADC_CTL[2]) = 1 and SPLIEn (EADC_INTSRC0[n],n=0~18) = 1, all sample module EOC (End of conversion) pulses can cause an ADINT0 interrupt.

The ADINT0, ADINT1 interrupt pulses are generated whenever the specific sample module A/D EOC pulse is generated. It also can be the sample module conversion trigger sources, and user can use it to do the ADC continuous scan conversion.

Example for “Continuous scan”:

1. If ADC sample module 2 EOC2 pulse is selected as ADINT0 interrupt trigger SPLIE2 (EADC_INTSRC0[2]) = 1 and ADINT0 is selected as sample module 0, 1, 2 hardware conversion trigger.
2. Set software trigger $SWTRG_2$ (EADC_SWTRG[2]) to 1 to start a sample module 2 ADC conversion, after the conversion completes, it generates an EOC2 pulse signal and ADINT0 interrupt pulse at end of sample module 2 ADC conversion, ADINT0 interrupt pulse will trigger the sample module 0, 1, 2 to start the ADC conversions.
3. ADINT0 interrupt pulse repeats to trigger sample module 0, 1, 2 ADC conversions automatically.
4. Clear TRGSEL (EADC_SCTL2[20:16]) to 0 to disable sample module 2 ADINT0 interrupt pulse hardware trigger, if needs to stop the continuous scan.

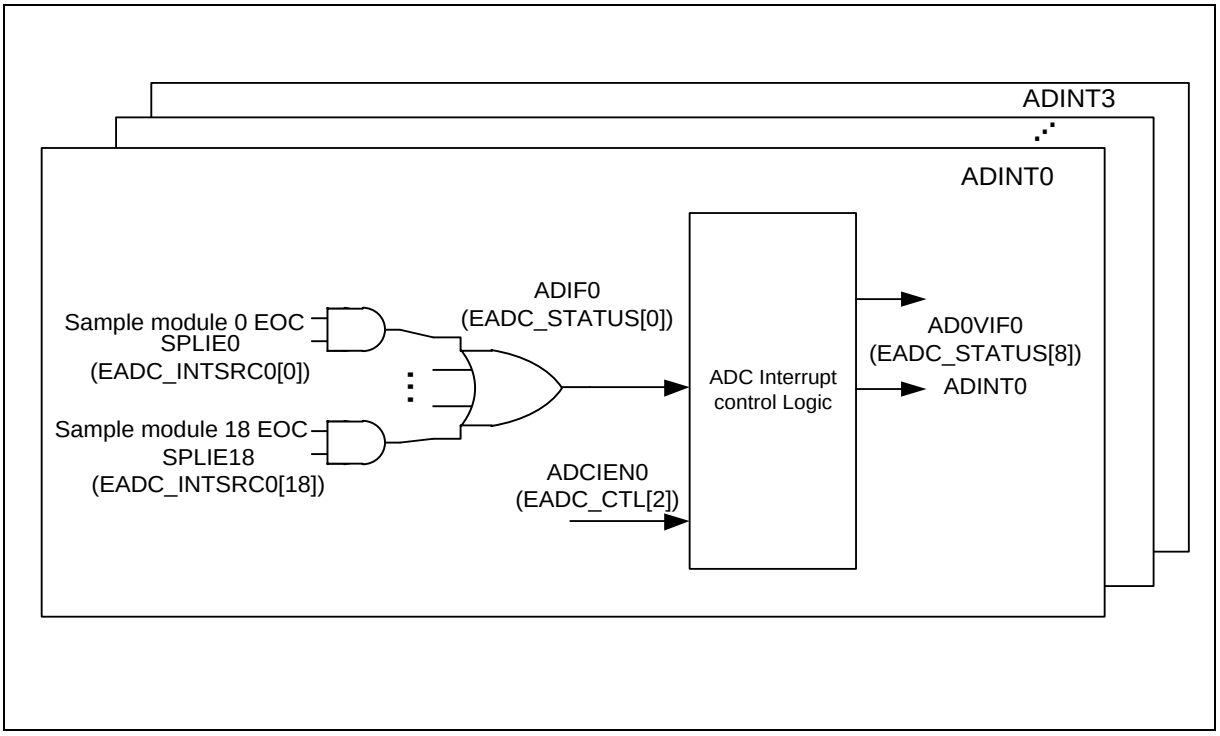


Figure 6.23-8 Specific Sample Module A/D EOC Signal for ADINT0~3 Interrupt

6.23.5.5 ADC Trigger by External Pin STADC ,Timer Trigger and PWM Trigger

A/D conversion can be triggered by external pin STADC request. Setting the TRGSEL (EADC_SCTLn[20:16], n=0~15) to 0x01 is to select external trigger input from the STADC pin. User can set EXTFEN (EADC_SCTLn[5], n=0~15) and EXTREN (EADC_SCTLn[4], n=0~15) to enable pin STADC trigger condition is falling or rising edge. There is a de-bounce circuit to detect falling or rising edge. If rising edge trigger condition is selected, the low state must be kept at least 2 PCLK cycles and the following high state must be kept at least 3 PCLK cycles. If falling edge trigger condition is selected, the high state must be kept at least 2 PCLK cycles and the following low state must be kept at least 3 PCLK cycles. Pulse that is shorter than this specification will be ignored.

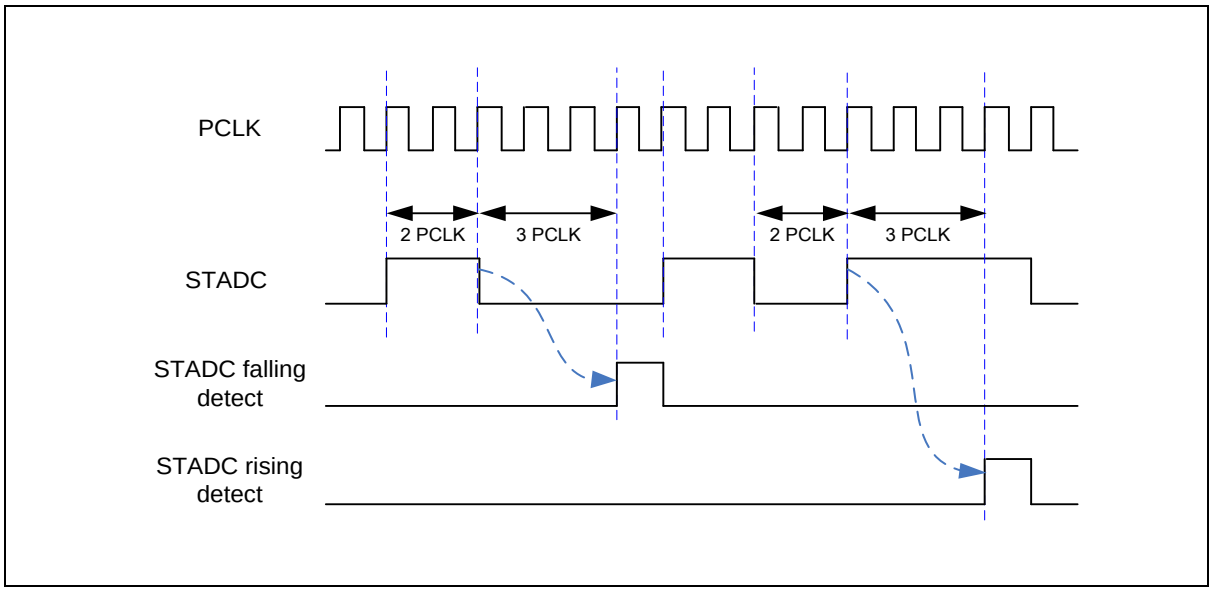


Figure 6.23-9 STADC De-bounce Timing Diagram

There are 4 Timer trigger sources and 12 PWM trigger sources (rising, falling PWM edge or center point of PWM) which can be selected to configure sample module 0~15 for ADC start trigger. The detail trigger conditions are descripted at PWM_EADCTS0, PWM_EADCTS1 and TIMER0_CTL ~ TIMER3_CTL register.

6.23.5.6 ADC Trigger Delay

ADC controller also allows user to configure the amount of delay prior to ADC start after hardware detected the trigger. User can configure the trigger delay time by setting TRGDLYCNT (EADC_SCTLn[15:8], n=0~15) and TRGDLYDIV (EADC_SCTLn[7:6], n=8~15). The Figure 6.23-10 shows the programmable delay time for PWM-triggered ADC start conversion.

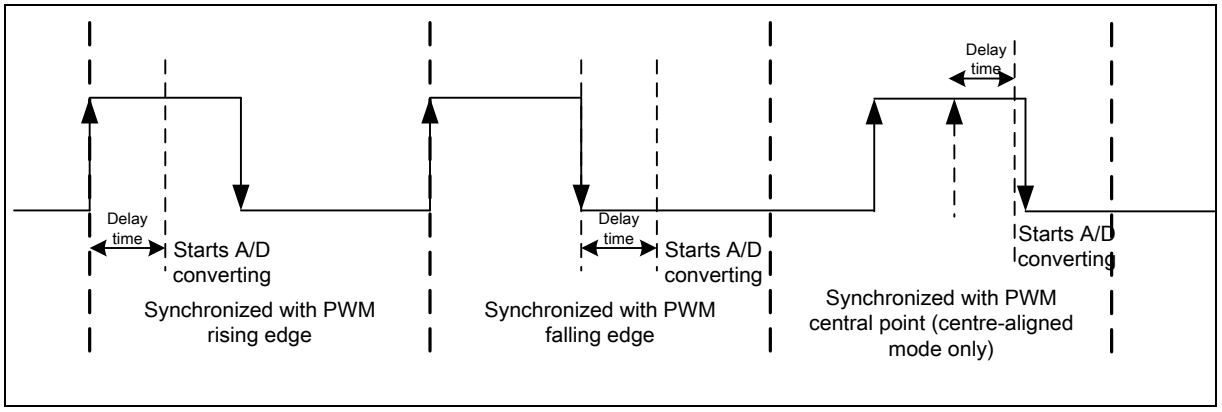


Figure 6.23-10 PWM-triggered ADC Start Conversion

The Figure 6.23-11 shows the programmable delay time for other trigger source.

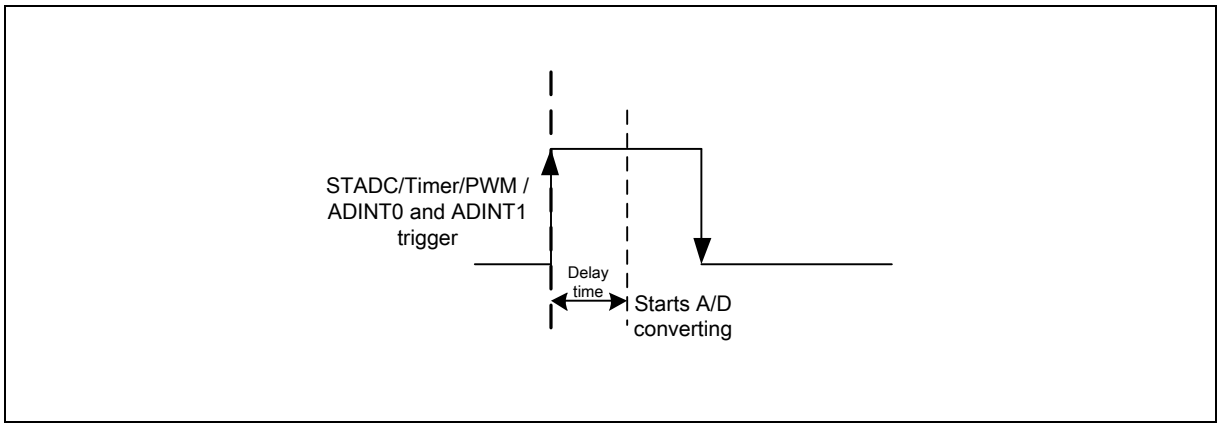


Figure 6.23-11 External triggered ADC Start Conversion

6.23.5.7 Input Sampling and A/D Conversion Time

The A/D converter sample the analog input when A/D conversion start delay time (Td) has passed after SWTRGn (EADC_SWTRG[n], n=0~18) is set to 1, then start conversion. Due to ADC clock is generated by PCLK divided by (EADCDIV(CLK_CLKDIV0[23:16])+1), the maximum delay time from user write SWTRGn to A/D start sampling analog input time is two ADC clock cycles. The start delay time is shown below:

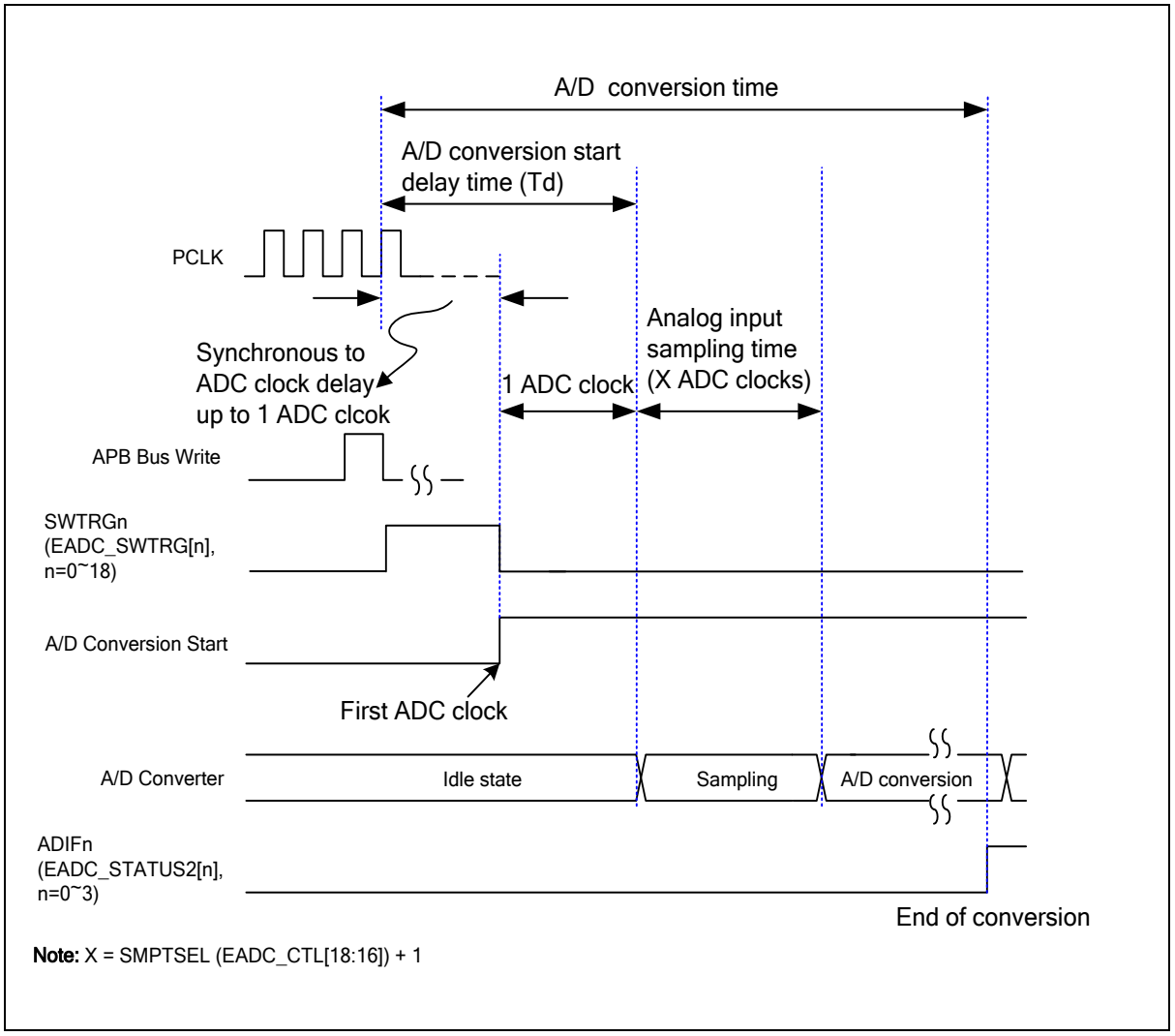
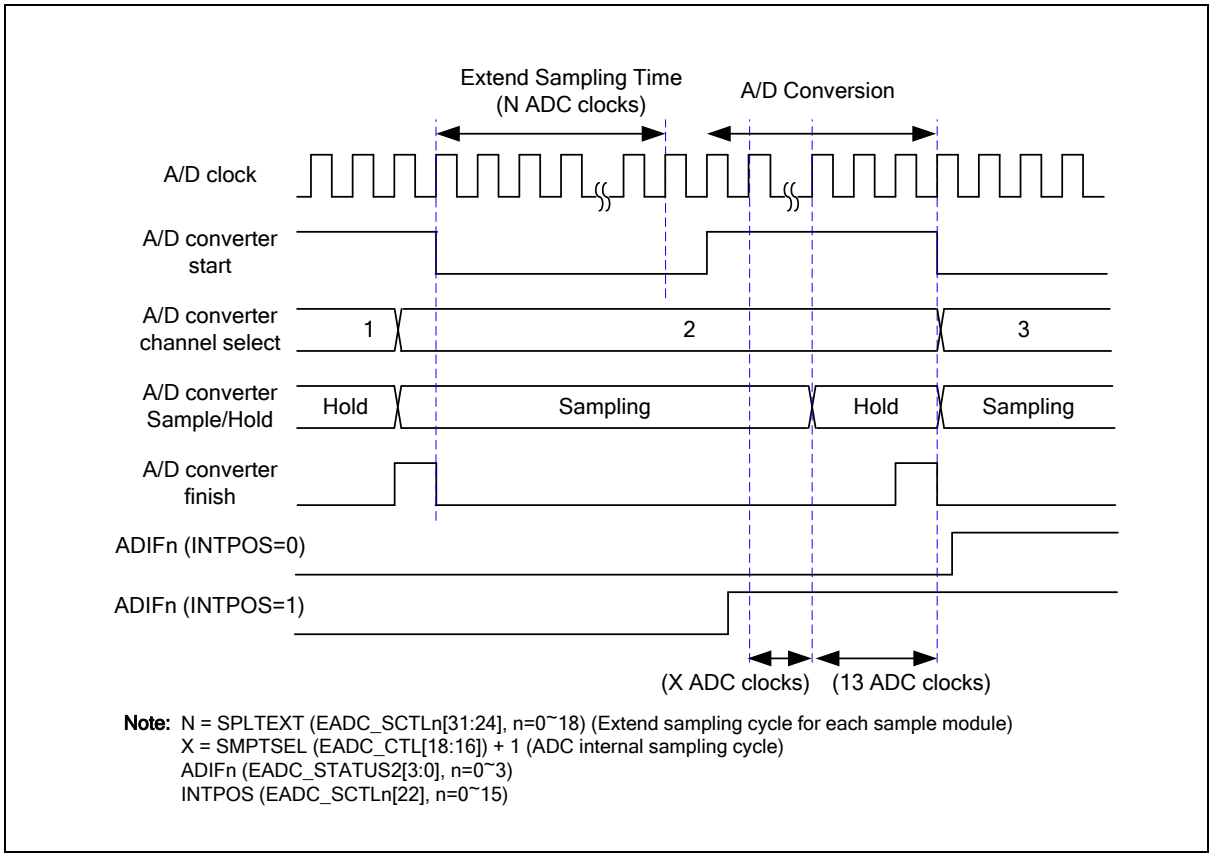


Figure 6.23-12 Conversion Start Delay Timing Diagram

6.23.5.8 A/D Extend Sampling Time

When A/D operation at high ADC clock rate, the sampling time of analog input voltage may not enough if the analog channel has heavy loading to cause fully charge time is longer. User can set SMPTSEL (EADC_CTL[18:16]) to select the sampling cycle in ADC. User also can set extend sampling time by writing EXTSMPT (EADC_SCTLn[31:24], n=0~15) for each sample module. The A/D extend sampling time is present between A/D controller judge which channel to be converting and A/D start to conversion. The range of extend sampling time is from 0 ~255 ADC clock. The extended sampling time is shown below:



Note: N = SPLTEXT (EADC_SCTLn[31:24], n=0~18) (Extend sampling cycle for each sample module)
X = SMPTSEL (EADC_CTL[18:16]) + 1 (ADC internal sampling cycle)
ADIFn (EADC_STATUS2[3:0], n=0~3)
INTPOS (EADC_SCTLn[22], n=0~15)

Figure 6.23-13 A/D Extend Sampling Timing Diagram

6.23.5.9 Conversion Result Monitor by Compare Mode

The ADC controller provides four sets of compare registers EADC_CMP0 ~ EADC_CMP3 to monitor a maximum of four specified sample module 0~18 conversion results from A/D conversion module, as shown in theFigure 6.23-14. User can select which sample module result to be monitored by set CMPSPL (EADC_CMPn[7:3], n =0~3) and CMPCOND (EADC_CMPn[2] , n =0~3) is used to check conversion result is less than specify value or greater than (equal to) value specified in CMPDAT (EADC_CMPn[27:16] , n =0~3). When the conversion of the sample module specified by CMPSPL is completed, the comparing action will be triggered one time automatically. When the compare result meets the compare condition, the internal compare match counter will increase 1. If the compare result does not meet the condition, the compare match counter will reset to 0. When counter value reach the setting of (CMPMCNT (EADC_CMPn[11:8])+1, n =0~3) then ADCMPFn (EADC_STATUS2[7:4] , n =0~3) bit will be set to 1, if ADCMPIE (EADC_CMPn[1] , n =0~3) is set then an ADINT3 interrupt request is generated. User can use it to monitor the external analog input pin voltage transition. Detailed logics diagram is shown below:

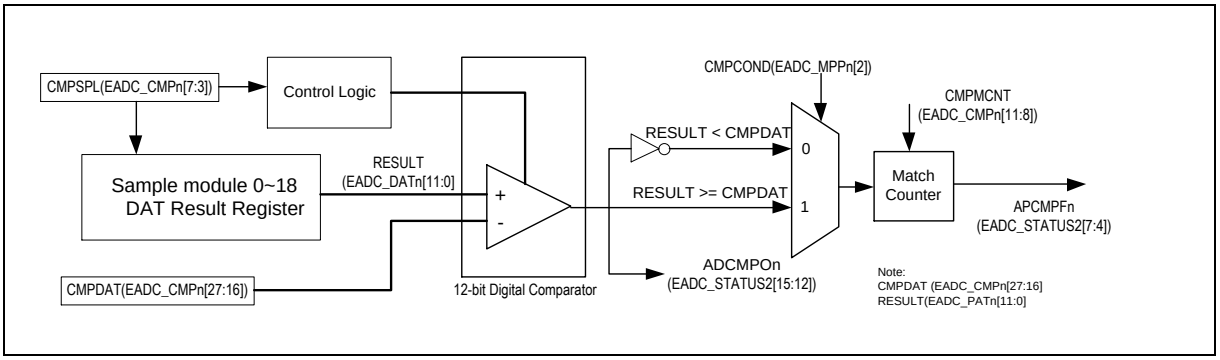


Figure 6.23-14 A/D Conversion Result Monitor Logics Diagram

The ADC controller supports a window compare mode. User can set CMPWEN (EADC_CMP0[15]/ EADC_CMP2[15]) to enable this function. If user enable this function, ADCMPF0 (EADC_STATUS2[4]) will be set when both EADC_CMP0 and EADC_CMP1 compared condition matched. ADCMPF2 (EADC_STATUS2[6]) will be set when both EADC_CMP2 and EADC_CMP3 compared condition matched.

6.23.5.10 Differential Mode

The ADC controller supports analog differential mode. If user enable DIFFEN (EADC_CTL[8]), the differential mode will enable.

Differential analog input voltage (Vdiff) = Vplus - Vminus, where Vplus is the analog input; Vminus is the inverted analog input.

In differential analog input mode, only the even number of the two corresponding channels needs to be enabled in CHSEL (EADC_SCTLn[3:0]). The conversion result will be placed to the corresponding data register of the enabled channel. The conversion result will store with 2'complement format when DMOF (EADC_CTL[9]) = 1.

Differential Analog Input Paired Channel	**ADC Analog Input**	
	V_{plus}	V_{minus}
0	EADC_CH0	EADC_CH1
1	EADC_CH2	EADC_CH3
2	EADC_CH4	EADC_CH5
3	EADC_CH6	EADC_CH7
4	EADC_CH8	EADC_CH9
5	EADC_CH10	EADC_CH11
6	EADC_CH12	EADC_CH13
7	EADC_CH14	EADC_CH15

Table 6-43 EADC Differential Model Channel Table

6.23.5.11 Double Buffer Mode

The ADC controller supports a double buffer mode in sample module 0~3. If user enable DBMEN (EADC_SCTLn[23], n=0~3), the double buffer mode will enable. In double buffer mode, after first time

A/D convert finish, the VALID (EADC_DATn[17], n=0~3) will set to high, but VALID[n] (EADC_STATUS0[n], n=0~3) will keep low. And the second time A/D converts finish, VALID (EADC_DDATn[17],n=0~3) will set to high, and VALID (EADC_STATUS0[n], n=0~3) will set to high at the same time. After VALID (EADC_STATUS [n], n=0~3) is high, user can get the ADC results from EADC_DATn and EADC_DDATn register.

6.23.5.12 PDMA request

The ADC controller supports PDMA. User can enable PDMAEN (EADC_CTL[11]) and configure PDMA channel's source address as EADC_CURDAT (EADC_BA+0x4C). After enable PDMAEN and PDMA channel enable, if any VALID (EADC_DATn[17],n=0~18) is high, ADC controller will send request to PDMA and PDMA will read EADC_CURDAT to get result. The EADC_CURDAT register is a shadow register of highest priority EADC_DAT register. The lower number sample module is higher priority. After PDMA read EADC_CURDAT register, the VAILD of the shadow EADC_DAT register will be automatically cleared.

6.23.5.13 Interrupt Sources

The A/D converter generates ADIFn (EADC_STATUS2[3:0], n=0~3) at the start of conversion or the end of conversion decide by INTPOS (EADC_SCTLn[22], n=0~15). If ADCIENn (EADC_CTL[5:2], n=0~3) is set then conversion end interrupt request ADINTn (n=0~3) is generated.

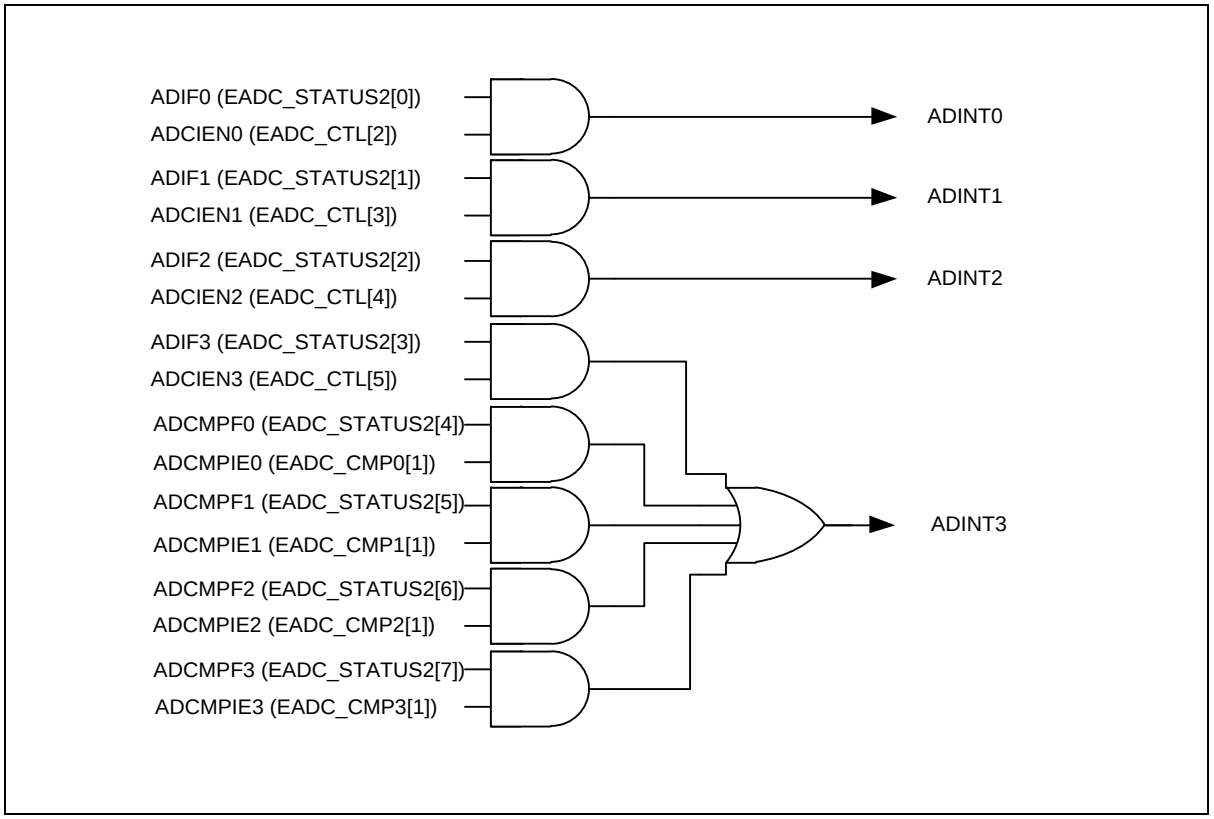


Figure 6.23-15 A/D Controller Interrupts

6.23.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 0 can be written

Register	Offset	R/W	Description	Reset Value
EADC Base Address: **EADC_BA = 0x4004_3000**				
EADC_DAT0	EADC_BA+0x00	R	A/D Data Register 0 for Sample Module 0	0x0000_0000
EADC_DAT1	EADC_BA+0x04	R	A/D Data Register 1 for Sample Module 1	0x0000_0000
EADC_DAT2	EADC_BA+0x08	R	A/D Data Register 2 for Sample Module 2	0x0000_0000
EADC_DAT3	EADC_BA+0x0C	R	A/D Data Register 3 for Sample Module 3	0x0000_0000
EADC_DAT4	EADC_BA+0x10	R	A/D Data Register 4 for Sample Module 4	0x0000_0000
EADC_DAT5	EADC_BA+0x14	R	A/D Data Register 5 for Sample Module 5	0x0000_0000
EADC_DAT6	EADC_BA+0x18	R	A/D Data Register 6 for Sample Module 6	0x0000_0000
EADC_DAT7	EADC_BA+0x1C	R	A/D Data Register 7 for Sample Module 7	0x0000_0000
EADC_DAT8	EADC_BA+0x20	R	A/D Data Register 8 for Sample Module 8	0x0000_0000
EADC_DAT9	EADC_BA+0x24	R	A/D Data Register 9 for Sample Module 9	0x0000_0000
EADC_DAT10	EADC_BA+0x28	R	A/D Data Register 10 for Sample Module 10	0x0000_0000
EADC_DAT11	EADC_BA+0x2C	R	A/D Data Register 11 for Sample Module 11	0x0000_0000
EADC_DAT12	EADC_BA+0x30	R	A/D Data Register 12 for Sample Module 12	0x0000_0000
EADC_DAT13	EADC_BA+0x34	R	A/D Data Register 13 for Sample Module 13	0x0000_0000
EADC_DAT14	EADC_BA+0x38	R	A/D Data Register 14 for Sample Module 14	0x0000_0000
EADC_DAT15	EADC_BA+0x3C	R	A/D Data Register 15 for Sample Module 15	0x0000_0000
EADC_DAT16	EADC_BA+0x40	R	A/D Data Register 16 for Sample Module 16	0x0000_0000
EADC_DAT17	EADC_BA+0x44	R	A/D Data Register 17 for Sample Module 17	0x0000_0000
EADC_DAT18	EADC_BA+0x48	R	A/D Data Register 18 for Sample Module 18	0x0000_0000
EADC_CURDAT	EADC_BA+0x4C	R	EADC PDMA Current Transfer Data Register	0x0000_0000
EADC_CTL	EADC_BA+0x50	R/W	A/D Control Register	0x0005_0000
EADC_SWTRG	EADC_BA+0x54	W	A/D Sample Module Software Start Register	0x0000_0000
EADC_PENDSTS	EADC_BA+0x58	R/W	A/D Start of Conversion Pending Flag Register	0x0000_0000
EADC_OVSTS	EADC_BA+0x5C	R/W	A/D Sample Module Start of Conversion Overrun Flag Register	0x0000_0000
EADC_SCTL0	EADC_BA+0x80	R/W	A/D Sample Module 0 Control Register	0x0000_0000
EADC_SCTL1	EADC_BA+0x84	R/W	A/D Sample Module 1 Control Register	0x0000_0000

Register	Offset	R/W	Description	Reset Value
EADC Base Address: **EADC_BA = 0x4004_3000**				
EADC_SCTL2	EADC_BA+0x88	R/W	A/D Sample Module 2 Control Register	0x0000_0000
EADC_SCTL3	EADC_BA+0x8C	R/W	A/D Sample Module 3 Control Register	0x0000_0000
EADC_SCTL4	EADC_BA+0x90	R/W	A/D Sample Module 4 Control Register	0x0000_0000
EADC_SCTL5	EADC_BA+0x94	R/W	A/D Sample Module 5 Control Register	0x0000_0000
EADC_SCTL6	EADC_BA+0x98	R/W	A/D Sample Module 6 Control Register	0x0000_0000
EADC_SCTL7	EADC_BA+0x9C	R/W	A/D Sample Module 7 Control Register	0x0000_0000
EADC_SCTL8	EADC_BA+0xA0	R/W	A/D Sample Module 8 Control Register	0x0000_0000
EADC_SCTL9	EADC_BA+0xA4	R/W	A/D Sample Module 9 Control Register	0x0000_0000
EADC_SCTL10	EADC_BA+0xA8	R/W	A/D Sample Module 10 Control Register	0x0000_0000
EADC_SCTL11	EADC_BA+0xAC	R/W	A/D Sample Module 11 Control Register	0x0000_0000
EADC_SCTL12	EADC_BA+0xB0	R/W	A/D Sample Module 12 Control Register	0x0000_0000
EADC_SCTL13	EADC_BA+0xB4	R/W	A/D Sample Module 13 Control Register	0x0000_0000
EADC_SCTL14	EADC_BA+0xB8	R/W	A/D Sample Module 14 Control Register	0x0000_0000
EADC_SCTL15	EADC_BA+0xBC	R/W	A/D Sample Module 15 Control Register	0x0000_0000
EADC_SCTL16	EADC_BA+0xC0	R/W	A/D Sample Module 16 Control Register	0x0000_0000
EADC_SCTL17	EADC_BA+0xC4	R/W	A/D Sample Module 17 Control Register	0x0000_0000
EADC_SCTL18	EADC_BA+0xC8	R/W	A/D Sample Module 18 Control Register	0x0000_0000
EADC_INTSRC0	EADC_BA+0xD0	R/W	ADC interrupt 0 Source Enable Control Register.	0x0000_0000
EADC_INTSRC1	EADC_BA+0xD4	R/W	ADC interrupt 1 Source Enable Control Register.	0x0000_0000
EADC_INTSRC2	EADC_BA+0xD8	R/W	ADC interrupt 2 Source Enable Control Register.	0x0000_0000
EADC_INTSRC3	EADC_BA+0xDC	R/W	ADC interrupt 3 Source Enable Control Register.	0x0000_0000
EADC_CMP0	EADC_BA+0xE0	R/W	A/D Result Compare Register 0	0x0000_0000
EADC_CMP1	EADC_BA+0xE4	R/W	A/D Result Compare Register 1	0x0000_0000
EADC_CMP2	EADC_BA+0xE8	R/W	A/D Result Compare Register 2	0x0000_0000
EADC_CMP3	EADC_BA+0xEC	R/W	A/D Result Compare Register 3	0x0000_0000
EADC_STATUS0	EADC_BA+0xF0	R	A/D Status Register 0	0x0000_0000
EADC_STATUS1	EADC_BA+0xF4	R	A/D Status Register 1	0x0000_0000
EADC_STATUS2	EADC_BA+0xF8	R/W	A/D Status Register 2	0x0000_0000

Register	Offset	R/W	Description	Reset Value
EADC Base Address: **EADC_BA = 0x4004_3000**				
EADC_STATUS3	EADC_BA+0xFC	R	A/D Status Register 3	0x0000_001F
EADC_DDAT0	EADC_BA+0x100	R	A/D Double Data Register 0 for Sample Module 0	0x0000_0000
EADC_DDAT1	EADC_BA+0x104	R	A/D Double Data Register 1 for Sample Module 1	0x0000_0000
EADC_DDAT2	EADC_BA+0x108	R	A/D Double Data Register 2 for Sample Module 2	0x0000_0000
EADC_DDAT3	EADC_BA+0x10C	R	A/D Double Data Register 3 for Sample Module 3	0x0000_0000

6.23.7 Register Description

A/D Data Registers (EADC_DAT0~18)

Register	Offset	R/W	Description	Reset Value
EADC_DAT0	EADC_BA+0x00	R	A/D Data Register 0 for Sample Module 0	0x0000_0000
EADC_DAT1	EADC_BA+0x04	R	A/D Data Register 1 for Sample Module 1	0x0000_0000
EADC_DAT2	EADC_BA+0x08	R	A/D Data Register 2 for Sample Module 2	0x0000_0000
EADC_DAT3	EADC_BA+0x0C	R	A/D Data Register 3 for Sample Module 3	0x0000_0000
EADC_DAT4	EADC_BA+0x10	R	A/D Data Register 4 for Sample Module 4	0x0000_0000
EADC_DAT5	EADC_BA+0x14	R	A/D Data Register 5 for Sample Module 5	0x0000_0000
EADC_DAT6	EADC_BA+0x18	R	A/D Data Register 6 for Sample Module 6	0x0000_0000
EADC_DAT7	EADC_BA+0x1C	R	A/D Data Register 7 for Sample Module 7	0x0000_0000
EADC_DAT8	EADC_BA+0x20	R	A/D Data Register 8 for Sample Module 8	0x0000_0000
EADC_DAT9	EADC_BA+0x24	R	A/D Data Register 9 for Sample Module 9	0x0000_0000
EADC_DAT10	EADC_BA+0x28	R	A/D Data Register 10 for Sample Module 10	0x0000_0000
EADC_DAT11	EADC_BA+0x2C	R	A/D Data Register 11 for Sample Module 11	0x0000_0000
EADC_DAT12	EADC_BA+0x30	R	A/D Data Register 12 for Sample Module 12	0x0000_0000
EADC_DAT13	EADC_BA+0x34	R	A/D Data Register 13 for Sample Module 13	0x0000_0000
EADC_DAT14	EADC_BA+0x38	R	A/D Data Register 14 for Sample Module 14	0x0000_0000
EADC_DAT15	EADC_BA+0x3C	R	A/D Data Register 15 for Sample Module 15	0x0000_0000
EADC_DAT16	EADC_BA+0x40	R	A/D Data Register 16 for Sample Module 16	0x0000_0000
EADC_DAT17	EADC_BA+0x44	R	A/D Data Register 17 for Sample Module 17	0x0000_0000
EADC_DAT18	EADC_BA+0x48	R	A/D Data Register 18 for Sample Module 18	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved						**VALID**	**OV**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RESULT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESULT							

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**VALID**	**Valid Flag** This bit is set to 1 when corresponding sample module channel analog input conversion is completed and cleared by hardware after EADC_DAT register is read. 0 = Data in RESULT[11:0] bits is not valid. 1 = Data in RESULT[11:0] bits is valid.
[16]	**OV**	**Overrun Flag** If converted data in RESULT[11:0] has not been read before new conversion result is loaded to this register, OV is set to 1. 0 = Data in RESULT[11:0] is recent conversion result. 1 = Data in RESULT[11:0] is overwrite. **Note:** It is cleared by hardware after EADC_DAT register is read.
[15:0]	**RESULT**	**A/D Conversion Result** This field contains 12 bits conversion result. When DMOF (EADC_CTL[9]) is set to 0, 12-bit ADC conversion result with unsigned format will be filled in RESULT[11:0] and zero will be filled in RESULT[15:12]. When DMOF (EADC_CTL[9]) set to 1, 12-bit ADC conversion result with 2'complement format will be filled in RESULT[11:0] and signed bits to will be filled in RESULT[15:12].

EADC PDMA Current Transfer Data Register (EADC_CURDAT)

Register	Offset	R/W	Description	Reset Value
EADC_CURDAT	EADC_BA+0x4C	R	EADC PDMA Current Transfer Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved						**CURDAT**	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CURDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CURDAT							

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17:0]	**CURDAT**	**ADC PDMA Current Transfer Data Register** This register is a shadow register of EADC_DATn (n=0~18) for PDMA support. This is a read only register. **NOTE:** After PDMA read this register, the VAILD of the shadow EADC_DAT register will be automatically cleared.

A/D Control Register (EADC_CTL)

Register	Offset	R/W	Description	Reset Value
EADC_CTL	EADC_BA+0x50	R/W	A/D Control Register	0x0005_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					SMPTSEL		
15	14	13	12	11	10	9	8
Reserved				PDMAEN	Reserved	DMOF	DIFFEN
7	6	5	4	3	2	1	0
Reserved		ADCIEN3	ADCIEN2	ADCIEN1	ADCIEN0	ADCRST	ADCEN

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:16]	**SMPTSEL**	**ADC Internal Sampling Time Selection** ADC internal sampling cycle = SMPTSEL + 1. 000 = 1 ADC clock sampling time. 001 = 2 ADC clock sampling time. 010 = 3 ADC clock sampling time. 011 = 4 ADC clock sampling time. 100 = 5 ADC clock sampling time. 101 = 6 ADC clock sampling time. 110 = 7 ADC clock sampling time. 111 = 8 ADC clock sampling time.
[15:12]	**Reserved**	Reserved.
[11]	**PDMAEN**	**PDMA Transfer Enable Bit** When A/D conversion is completed, the converted data is loaded into EADC_DATn (n: 0 ~ 18) register, user can enable this bit to generate a PDMA data transfer request. 0 = PDMA data transfer Disabled. 1 = PDMA data transfer Enabled. **Note:** When set this bit field to 1, user should set ADCIENn (EADC_CTL[5:2], n=0~3) = 0 to disable interrupt.
[10]	**Reserved**	Reserved.
[9]	**DMOF**	**ADC Differential Input Mode Output Format** 0 = A/D conversion result will be filled in RESULT (EADC_DATn[15:0] , n= 0 ~18) with unsigned format. 1 = A/D conversion result will be filled in RESULT (EADC_DATn[15:0] , n= 0 ~18) with 2'complement format. **Note:** This bit must be set to 0 in single-end analog input mode.

Bits	Description	
[8]	**DIFFEN**	**Differential Analog Input Mode Enable Bit** 0 = Single-end analog input mode. 1 = Differential analog input mode.
[7:6]	**Reserved**	Reserved.
[5]	**ADCIEN3**	**Specific Sample Module A/D ADINT3 Interrupt Enable Bit** The A/D converter generates a conversion end ADIF3 (EADC_STATUS2[3]) upon the end of specific sample module A/D conversion. If ADCIEN3 bit is set then conversion end interrupt request ADINT3 is generated. 0 = Specific sample module A/D ADINT3 interrupt function Disabled. 1 = Specific sample module A/D ADINT3 interrupt function Enabled.
[4]	**ADCIEN2**	**Specific Sample Module A/D ADINT2 Interrupt Enable Bit** The A/D converter generates a conversion end ADIF2 (EADC_STATUS2[2]) upon the end of specific sample module A/D conversion. If ADCIEN2 bit is set then conversion end interrupt request ADINT2 is generated. 0 = Specific sample module A/D ADINT2 interrupt function Disabled. 1 = Specific sample module A/D ADINT2 interrupt function Enabled.
[3]	**ADCIEN1**	**Specific Sample Module A/D ADINT1 Interrupt Enable Bit** The A/D converter generates a conversion end ADIF1 (EADC_STATUS2[1]) upon the end of specific sample module A/D conversion. If ADCIEN1 bit is set then conversion end interrupt request ADINT1 is generated. 0 = Specific sample module A/D ADINT1 interrupt function Disabled. 1 = Specific sample module A/D ADINT1 interrupt function Enabled.
[2]	**ADCIEN0**	**Specific Sample Module A/D ADINT0 Interrupt Enable Bit** The A/D converter generates a conversion end ADIF0 (EADC_STATUS2[0]) upon the end of specific sample module A/D conversion. If ADCIEN0 bit is set then conversion end interrupt request ADINT0 is generated. 0 = Specific sample module A/D ADINT0 interrupt function Disabled. 1 = Specific sample module A/D ADINT0 interrupt function Enabled.
[1]	**ADCRST**	**ADC A/D Converter Control Circuits Reset** 0 = No effect. 1 = Cause ADC control circuits reset to initial state, but not change the ADC registers value. **Note:** ADCRST bit remains 1 during ADC reset, when ADC reset end, the ADCRST bit is automatically cleared to 0.
[0]	**ADCEN**	**A/D Converter Enable Bit** 0 = ADC Disabled. 1 = ADC 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 power consumption.

A/D Sample Module Software Start Register (EADC_SWTRG)

Register	Offset	R/W	Description	Reset Value
EADC_SWTRG	EADC_BA+0x54	W	A/D Sample Module Software Start Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**SWTRG**		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SWTRG							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SWTRG							

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:0]	**SWTRG**	**A/D Sample Module 0~18 Software Force to Start ADC Conversion** 0 = No effect. 1 = Cause an ADC conversion when the priority is given to sample module. **Note:** After write this register to start ADC conversion, the EADC_PENDSTS register will show which sample module will conversion. If user want to disable the conversion of the sample module, user can write EADC_PENDSTS register to clear it.

A/D Sample Module Start of Conversion Pending Flag Register (EADC_PENDSTS)

Register	Offset	R/W	Description	Reset Value
EADC_PENDSTS	EADC_BA+0x58	R/W	A/D Start of Conversion Pending Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved					STPF		
15	14	13	12	11	10	9	8
STPF							
7	6	5	4	3	2	1	0
STPF							

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:0]	**STPF**	**A/D Sample Module 0~18 Start of Conversion Pending Flag** **Read:** 0 = There is no pending conversion for sample module. 1 = Sample module ADC start of conversion is pending. **Write:** 1 = Clear pending flag and stop conversion for corresponding sample module. **Note1:** This bit remains 1 during pending state, when the respective ADC conversion is end, the STPFn (n=0~18) bit is automatically cleared to 0. **Note2:** After stopping current conversion, the corresponding EADC_DATn (n=0~18) keeps its original value.

A/D Sample Module Overrun Flag Register (EADC_OVSTS)

Register	Offset	R/W	Description	Reset Value
EADC_OVSTS	EADC_BA+0x5C	R/W	A/D Sample Module Start of Conversion Overrun Flag Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**SPOVF**		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SPOVF							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SPOVF							

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:0]	**SPOVF**	**A/D SAMPLE0~18 Overrun Flag** 0 = No sample module event overrun. 1 = Indicates a new sample module event is generated while an old one event is pending. **Note:** This bit is cleared by writing 1 to it.

A/D Sample Module 0~3 Control Registers (EADC_SCTL0~EADC_SCTL3)

Register	Offset	R/W	Description	Reset Value
EADC_SCTL0	EADC_BA+0x80	R/W	A/D Sample Module 0 Control Register	0x0000_0000
EADC_SCTL1	EADC_BA+0x84	R/W	A/D Sample Module 1 Control Register	0x0000_0000
EADC_SCTL2	EADC_BA+0x88	R/W	A/D Sample Module 2 Control Register	0x0000_0000
EADC_SCTL3	EADC_BA+0x8C	R/W	A/D Sample Module 3 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
EXTSMPT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DBMEN	INTPOS	Reserved	TRGSEL				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TRGDLYCNT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TRGDLYDIV		EXTFEN	EXTREN	CHSEL			

Bits	Description	
[31:24]	**EXTSMPT**	**ADC Sampling Time Extend** When A/D converting at high conversion rate, the sampling time of analog input voltage may not enough if input channel loading is heavy, user can extend A/D sampling time after trigger source is coming to get enough sampling time. The range of start delay time is from 0~255 ADC clock.
[23]	**DBMEN**	**Double Buffer Mode Enable Bit** 0 = Sample has one sample result register. (default). 1 = Sample has two sample result registers.
[22]	**INTPOS**	**Interrupt Flag Position Select** 0 = Set ADIFn (EADC_STATUS2[n], n=0~3) at A/D end of conversion. 1 = Set ADIFn (EADC_STATUS2[n], n=0~3) at A/D start of conversion.
[21]	**Reserved**	Reserved.

Bits	Description	
[20:16]	**TRGSEL**	**A/D Sample Module Start of Conversion Trigger Source Selection** 0H = Disable trigger. 1H = External trigger from STADC pin input. 2H = ADC ADINT0 interrupt EOC (End of conversion) pulse trigger. 3H = ADC ADINT1 interrupt EOC (End of conversion) pulse trigger. 4H = Timer0 overflow pulse trigger. 5H = Timer1 overflow pulse trigger. 6H = Timer2 overflow pulse trigger. 7H = Timer3 overflow pulse trigger. 8H = PWM0TG0. 9H = PWM0TG1. AH = PWM0TG2. BH = PWM0TG3. CH = PWM0TG4. DH = PWM0TG5. EH = PWM1TG0. FH = PWM1TG1. 10H = PWM1TG2. 11H = PWM1TG3. 12H = PWM1TG4. 13H = PWM1TG5. other = Reserved. **NOTE:** Refer PWM_EADCTS0, PWM_EADCTS1 and TIMERn_CTL (n=0~3) to get more information for PWM trigger and timer trigger.
[15:8]	**TRGDLYCNT**	**A/D Sample Module Start of Conversion Trigger Delay Time** Trigger delay time = TRGDLYCNT x ADC_CLK x n (n=1,2,4,16 from TRGDLYDIV setting).
[7:6]	**TRGDLYDIV**	**A/D Sample Module Start of Conversion Trigger Delay Clock Divider Selection** Trigger delay clock frequency: 00 = ADC_CLK/1. 01 = ADC_CLK/2. 10 = ADC_CLK/4. 11 = ADC_CLK/16.
[5]	**EXTFEN**	**A/D External Trigger Falling Edge Enable Bit** 0 = Falling edge Disabled when A/D selects STADC as trigger source. 1 = Falling edge Enabled when A/D selects STADC as trigger source.
[4]	**EXTREN**	**A/D External Trigger Rising Edge Enable Bit** 0 = Rising edge Disabled when A/D selects STADC as trigger source. 1 = Rising edge Enabled when A/D selects STADC as trigger source.

Bits	Description	
[3:0]	**CHSEL**	**A/D Sample Module Channel Selection** 00H = EADC_CH0. 01H = EADC_CH1. 02H = EADC_CH2. 03H = EADC_CH3. 04H = EADC_CH4. 05H = EADC_CH5. 06H = EADC_CH6. 07H = EADC_CH7. 08H = EADC_CH8. 09H = EADC_CH9. 0AH = EADC_CH10. 0BH = EADC_CH11. 0CH = EADC_CH12. 0DH = EADC_CH13. 0EH = EADC_CH14. 0FH = EADC_CH15.

A/D Sample Module 4~15 Control Registers (EADC_SCTL4~EADC_SCTL15)

Register	Offset	R/W	Description	Reset Value
EADC_SCTL4	EADC_BA+0x90	R/W	A/D Sample Module 4 Control Register	0x0000_0000
EADC_SCTL5	EADC_BA+0x94	R/W	A/D Sample Module 5 Control Register	0x0000_0000
EADC_SCTL6	EADC_BA+0x98	R/W	A/D Sample Module 6 Control Register	0x0000_0000
EADC_SCTL7	EADC_BA+0x9C	R/W	A/D Sample Module 7 Control Register	0x0000_0000
EADC_SCTL8	EADC_BA+0xA0	R/W	A/D Sample Module 8 Control Register	0x0000_0000
EADC_SCTL9	EADC_BA+0xA4	R/W	A/D Sample Module 9 Control Register	0x0000_0000
EADC_SCTL10	EADC_BA+0xA8	R/W	A/D Sample Module 10 Control Register	0x0000_0000
EADC_SCTL11	EADC_BA+0xAC	R/W	A/D Sample Module 11 Control Register	0x0000_0000
EADC_SCTL12	EADC_BA+0xB0	R/W	A/D Sample Module 12 Control Register	0x0000_0000
EADC_SCTL13	EADC_BA+0xB4	R/W	A/D Sample Module 13 Control Register	0x0000_0000
EADC_SCTL14	EADC_BA+0xB8	R/W	A/D Sample Module 14 Control Register	0x0000_0000
EADC_SCTL15	EADC_BA+0xBC	R/W	A/D Sample Module 15 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
EXTSMPT							
23	22	21	20	19	18	17	16
Reserved	INTPOS	Reserved	TRGSEL				
15	14	13	12	11	10	9	8
TRGDLYCNT							
7	6	5	4	3	2	1	0
TRGDLYDIV		EXTFEN	EXTREN	CHSEL			

Bits	Description	
[31:24]	**EXTSMPT**	**ADC Sampling Time Extend** When A/D converting at high conversion rate, the sampling time of analog input voltage may not enough if input channel loading is heavy, SW can extend A/D sampling time after trigger source is coming to get enough sampling time. The range of start delay time is from 0~255 ADC clock.
[23]	**Reserved**	Reserved.
[22]	**INTPOS**	**Interrupt Flag Position Select** 0 = Set ADIFn (EADC_STATUS2[n], n=0~3) at A/D end of conversion. 1 = Set ADIFn (EADC_STATUS2[n], n=0~3) at A/D start of conversion.
[21]	**Reserved**	Reserved.

Bits	Description	
[20:16]	**TRGSEL**	**A/D Sample Module Start of Conversion Trigger Source Selection** 0H = Disable trigger. 1H = External trigger from STADC pin input. 2H = ADC ADINT0 interrupt EOC pulse trigger. 3H = ADC ADINT1 interrupt EOC pulse trigger. 4H = Timer0 overflow pulse trigger. 5H = Timer1 overflow pulse trigger. 6H = Timer2 overflow pulse trigger. 7H = Timer3 overflow pulse trigger. 8H = PWM0TG0. 9H = PWM0TG1. AH = PWM0TG2. BH = PWM0TG3. CH = PWM0TG4. DH = PWM0TG5. EH = PWM1TG0. FH = PWM1TG1. 10H = PWM1TG2. 11H = PWM1TG3. 12H = PWM1TG4. 13H = PWM1TG5. other = Reserved. **NOTE:** Refer PWM_EADCTS0, PWM_EADCTS1 and TIMERn_CTL (n=0~3) to get more information for PWM trigger and timer trigger.
[15:8]	**TRGDLYCNT**	**A/D Sample Module Start of Conversion Trigger Delay Time** Trigger delay time = TRGDLYCNT x ADC_CLK x n (n=1,2,4,16 from TRGDLYDIV setting).
[7:6]	**TRGDLYDIV**	**A/D Sample Module Start of Conversion Trigger Delay Clock Divider Selection** Trigger delay clock frequency: 00 = ADC_CLK/1. 01 = ADC_CLK/2. 10 = ADC_CLK/4. 11 = ADC_CLK/16.
[5]	**EXTFEN**	**A/D External Trigger Falling Edge Enable Bit** 0 = Falling edge Disabled when A/D selects STADC as trigger source. 1 = Falling edge Enabled when A/D selects STADC as trigger source.
[4]	**EXTREN**	**A/D External Trigger Rising Edge Enable Bit** 0 = Rising edge Disabled when A/D selects STADC as trigger source. 1 = Rising edge Enabled when A/D selects STADC as trigger source.

Bits	Description	
[3:0]	**CHSEL**	**A/D Sample Module Channel Selection** 00H = EADC_CH0. 01H = EADC_CH1. 02H = EADC_CH2. 03H = EADC_CH3. 04H = EADC_CH4. 05H = EADC_CH5. 06H = EADC_CH6. 07H = EADC_CH7. 08H = EADC_CH8. 09H = EADC_CH9. 0AH = EADC_CH10. 0BH = EADC_CH11. 0CH = EADC_CH12. 0DH = EADC_CH13. 0EH = EADC_CH14. 0FH = EADC_CH15.

A/D Sample Module 16~18 Control Registers (EADC_SCTL16~EADC_SCTL18)

Register	Offset	R/W	Description	Reset Value
EADC_SCTL16	EADC_BA+0xC0	R/W	A/D Sample Module 16 Control Register	0x0000_0000
EADC_SCTL17	EADC_BA+0xC4	R/W	A/D Sample Module 17 Control Register	0x0000_0000
EADC_SCTL18	EADC_BA+0xC8	R/W	A/D Sample Module 18 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
EXTSMPT							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:24]	**EXTSMPT**	**ADC Sampling Time Extend** When A/D converting at high conversion rate, the sampling time of analog input voltage may not enough if input channel loading is heavy, SW can extend A/D sampling time after trigger source is coming to get enough sampling time. The range of start delay time is from 0~255 ADC clock.
[23:0]	**Reserved**	Reserved.

A/D Interrupt Source Enable Control Registers (EADC_INTSRC0~EADC_INTSRC3)

Register	Offset	R/W	Description	Reset Value
EADC_INTSRC0	EADC_BA+0xD0	R/W	ADC interrupt 0 Source Enable Control Register.	0x0000_0000
EADC_INTSRC1	EADC_BA+0xD4	R/W	ADC interrupt 1 Source Enable Control Register.	0x0000_0000
EADC_INTSRC2	EADC_BA+0xD8	R/W	ADC interrupt 2 Source Enable Control Register.	0x0000_0000
EADC_INTSRC3	EADC_BA+0xDC	R/W	ADC interrupt 3 Source Enable Control Register.	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					SPLIE18	SPLIE17	SPLIE16
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SPLIE15	SPLIE14	SPLIE13	SPLIE12	SPLIE11	SPLIE10	SPLIE9	SPLIE8
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SPLIE7	SPLIE6	SPLIE5	SPLIE4	SPLIE3	SPLIE2	SPLIE1	SPLIE0

Bits	Description	
[18]	**SPLIE18**	**Sample Module 18 Interrupt Enable Bit** 0 = Sample Module 18 interrupt Disabled. 1 = Sample Module 18 interrupt Enabled.
[17]	**SPLIE17**	**Sample Module 17 Interrupt Enable Bit** 0 = Sample Module 17 interrupt Disabled. 1 = Sample Module 17 interrupt Enabled.
[16]	**SPLIE16**	**Sample Module 16 Interrupt Enable Bit** 0 = Sample Module 16 interrupt Disabled. 1 = Sample Module 16 interrupt Enabled.
[15]	**SPLIE15**	**Sample Module 15 Interrupt Enable Bit** 0 = Sample Module 15 interrupt Disabled. 1 = Sample Module 15 interrupt Enabled.
[14]	**SPLIE14**	**Sample Module 14 Interrupt Enable Bit** 0 = Sample Module 14 interrupt Disabled. 1 = Sample Module 14 interrupt Enabled.
[13]	**SPLIE13**	**Sample Module 13 Interrupt Enable Bit** 0 = Sample Module 13 interrupt Disabled. 1 = Sample Module 13 interrupt Enabled.
[12]	**SPLIE12**	**Sample Module 12 Interrupt Enable Bit** 0 = Sample Module 12 interrupt Disabled. 1 = Sample Module 12 interrupt Enabled.

Bits	Description	
[11]	**SPLIE11**	**Sample Module 11 Interrupt Enable Bit** 0 = Sample Module 11 interrupt Disabled. 1 = Sample Module 11 interrupt Enabled.
[10]	**SPLIE10**	**Sample Module 10 Interrupt Enable Bit** 0 = Sample Module 10 interrupt Disabled. 1 = Sample Module 10 interrupt Enabled.
[9]	**SPLIE9**	**Sample Module 9 Interrupt Enable Bit** 0 = Sample Module 9 interrupt Disabled. 1 = Sample Module 9 interrupt Enabled.
[8]	**SPLIE8**	**Sample Module 8 Interrupt Enable Bit** 0 = Sample Module 8 interrupt Disabled. 1 = Sample Module 8 interrupt Enabled.
[7]	**SPLIE7**	**Sample Module 7 Interrupt Enable Bit** 0 = Sample Module 7 interrupt Disabled. 1 = Sample Module 7 interrupt Enabled.
[6]	**SPLIE6**	**Sample Module 6 Interrupt Enable Bit** 0 = Sample Module 6 interrupt Disabled. 1 = Sample Module 6 interrupt Enabled.
[5]	**SPLIE5**	**Sample Module 5 Interrupt Enable Bit** 0 = Sample Module 5 interrupt Disabled. 1 = Sample Module 5 interrupt Enabled.
[4]	**SPLIE4**	**Sample Module 4 Interrupt Enable Bit** 0 = Sample Module 4 interrupt Disabled. 1 = Sample Module 4 interrupt Enabled.
[3]	**SPLIE3**	**Sample Module 3 Interrupt Enable Bit** 0 = Sample Module 3 interrupt Disabled. 1 = Sample Module 3 interrupt Enabled.
[2]	**SPLIE2**	**Sample Module 2 Interrupt Enable Bit** 0 = Sample Module 2 interrupt Disabled. 1 = Sample Module 2 interrupt Enabled.
[1]	**SPLIE1**	**Sample Module 1 Interrupt Enable Bit** 0 = Sample Module 1 interrupt Disabled. 1 = Sample Module 1 interrupt Enabled.
[0]	**SPLIE0**	**Sample Module 0 Interrupt Enable Bit** 0 = Sample Module 0 interrupt Disabled. 1 = Sample Module 0 interrupt Enabled.

A/D Result Compare Register 0/1/2/3 (EADC_CMP0/1/2/3)

Register	Offset	R/W	Description	Reset Value
EADC_CMP0	EADC_BA+0xE0	R/W	A/D Result Compare Register 0	0x0000_0000
EADC_CMP1	EADC_BA+0xE4	R/W	A/D Result Compare Register 1	0x0000_0000
EADC_CMP2	EADC_BA+0xE8	R/W	A/D Result Compare Register 2	0x0000_0000
EADC_CMP3	EADC_BA+0xEC	R/W	A/D Result Compare Register 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				CMPDAT			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CMPDAT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CMPWEN	Reserved			CMPMCNT			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMPSPL					CMPCOND	ADCMPIE	ADCMPEN

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:16]	**CMPDAT**	**Comparison Data** The 12 bits data is used to compare with conversion result of specified sample module. User can use it to monitor the external analog input pin voltage transition without imposing a load on software.
[15]	**CMPWEN**	**Compare Window Mode Enable Bit** 0 = ADCMPF0 (EADC_STATUS2[4]) will be set when EADC_CMP0 compared condition matched. ADCMPF2 (EADC_STATUS2[6]) will be set when EADC_CMP2 compared condition matched 1 = ADCMPF0 (EADC_STATUS2[4]) will be set when both EADC_CMP0 and EADC_CMP1 compared condition matched. ADCMPF2 (EADC_STATUS2[6]) will be set when both EADC_CMP2 and EADC_CMP3 compared condition matched. **Note:** This bit is only present in EADC_CMP0 and EADC_CMP2 register.
[14:12]	**Reserved**	Reserved.
[11:8]	**CMPMCNT**	**Compare Match Count** When the specified A/D sample module analog conversion result matches the compare condition defined by CMPCOND (EADC_CMPn[2], n=0~3), the internal match counter will increase 1. If the compare result does not meet the compare condition, the internal compare match counter will reset to 0. When the internal counter reaches the value to (CMPMCNT +1), the ADCMPFn (EADC_STATUS2[7:4], n=0~3) will be set.

Bits	Description	
[7:3]	**CMPSPL**	**Compare Sample Module Selection** 00000 = Sample Module 0 conversion result EADC_DAT0 is selected to be compared. 00001 = Sample Module 1 conversion result EADC_DAT1 is selected to be compared. 00010 = Sample Module 2 conversion result EADC_DAT2 is selected to be compared. 00011 = Sample Module 3 conversion result EADC_DAT3 is selected to be compared. 00100 = Sample Module 4 conversion result EADC_DAT4 is selected to be compared. 00101 = Sample Module 5 conversion result EADC_DAT5 is selected to be compared. 00110 = Sample Module 6 conversion result EADC_DAT6 is selected to be compared. 00111 = Sample Module 7 conversion result EADC_DAT7 is selected to be compared. 01000 = Sample Module 8 conversion result EADC_DAT8 is selected to be compared. 01001 = Sample Module 9 conversion result EADC_DAT9 is selected to be compared. 01010 = Sample Module 10 conversion result EADC_DAT10 is selected to be compared. 01011 = Sample Module 11 conversion result EADC_DAT11 is selected to be compared. 01100 = Sample Module 12 conversion result EADC_DAT12 is selected to be compared. 01101 = Sample Module 13 conversion result EADC_DAT13 is selected to be compared. 01110 = Sample Module 14 conversion result EADC_DAT14 is selected to be compared. 01111 = Sample Module 15 conversion result EADC_DAT15 is selected to be compared. 10000 = Sample Module 16 conversion result EADC_DAT16 is selected to be compared. 10001 = Sample Module 17 conversion result EADC_DAT17 is selected to be compared. 10010 = Sample Module 18 conversion result EADC_DAT18 is selected to be compared.
[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 (EADC_CMPn [27:16]), 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 (EADC_CMPn [27:16]), the internal match counter will increase one. **Note:** When the internal counter reaches the value to (CMPMCNT (EADC_CMPn[11:8], n=0~3) +1), the CMPF bit will be set.
[1]	**ADCMPIE**	**A/D Result Compare Interrupt Enable Bit** 0 = Compare function interrupt Disabled. 1 = Compare function interrupt Enabled. If the compare function is enabled and the compare condition matches the setting of CMPCOND (EADC_CMPn[2], n=0~3) and CMPMCNT (EADC_CMPn[11:8], n=0~3), ADCMPFn (EADC_STATUS2[7:4], n=0~3) will be asserted, in the meanwhile, if ADCMPIE is set to 1, a compare interrupt request is generated.
[0]	**ADCMPEN**	**A/D Result Compare Enable Bit** 0 = Compare Disabled. 1 = Compare Enabled. Set this bit to 1 to enable compare CMPDAT (EADC_CMPn[27:16], n=0~3) with specified sample module conversion result when converted data is loaded into EADC_DAT register.

A/D Status Register 0 (EADC_STATUS0)

Register	Offset	R/W	Description	Reset Value
EADC_STATUS0	EADC_BA+0xF0	R	A/D Status Register 0	0x0000_0000

31	30	29	28	27	26	25	24
OV[15:8]							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
OV[7:0]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
VALID[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
VALID[7:0]							

Bits	Description	
[31:16]	**OV[15:0]**	**EADC_DAT0~15 Overrun Flag** It is a mirror to OV bit in sample module A/D result data register EADC_DATn. (n=0~18).
[15:0]	**VALID[15:0]**	**EADC_DAT0~15 Data Valid Flag** It is a mirror of VALID bit in sample module A/D result data register EADC_DATn. (n=0~18).

A/D Status Register 1 (EADC_STATUS1)

Register	Offset	R/W	Description	Reset Value
EADC_STATUS1	EADC_BA+0xF4	R	A/D Status Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**OV[18:16]**		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**VALID[18:16]**		

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18:16]	**OV[18:16]**	**EADC_DAT16~18 Overrun Flag** It is a mirror to OV bit in sample module A/D result data register EADC_DATn. (n=0~18).
[15:3]	**Reserved**	Reserved.
[2:0]	**VALID[18:16]**	**EADC_DAT16~18 Data Valid Flag** It is a mirror of VALID bit in sample module A/D result data register EADC_DATn. (n=0~18).

A/D Status Register 2 (EADC_STATUS2)

Register	Offset	R/W	Description	Reset Value
EADC_STATUS2	EADC_BA+0xF8	R/W	A/D Status Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				AOV	AVALID	STOVF	ADOVIF
23	22	21	20	19	18	17	16
BUSY	Reserved		CHANNEL				
15	14	13	12	11	10	9	8
ADCMPO3	ADCMPO2	ADCMPO1	ADCMPO0	ADOVIF3	ADOVIF2	ADOVIF1	ADOVIF0
7	6	5	4	3	2	1	0
ADCMPF3	ADCMPF2	ADCMPF1	ADCMPF0	ADIF3	ADIF2	ADIF1	ADIF0

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27]	**AOV**	**for All Sample Module A/D Result Data Register Overrun Flags Check** n=0~18. 0 = None of sample module data register overrun flag OVn (EADC_DATn[16]) is set to 1. 1 = Any one of sample module data register overrun flag OVn (EADC_DATn[16]) is set to 1. **Note:** This bit will keep 1 when any OVn Flag is equal to 1.
[26]	**AVALID**	**for All Sample Module A/D Result Data Register EADC_DAT Data Valid Flag Check** n=0~18. 0 = None of sample module data register valid flag VALIDn (EADC_DATn[17]) is set to 1. 1 = Any one of sample module data register valid flag VALIDn (EADC_DATn[17]) is set to 1. **Note:** This bit will keep 1 when any VALIDn Flag is equal to 1.
[25]	**STOVF**	**for All A/D Sample Module Start of Conversion Overrun Flags Check** n=0~18. 0 = None of sample module event overrun flag SPOVFn (EADC_OVSTS[n]) is set to 1. 1 = Any one of sample module event overrun flag SPOVFn (EADC_OVSTS[n]) is set to 1. **Note:** This bit will keep 1 when any SPOVFn Flag is equal to 1.
[24]	**ADOVIF**	**All A/D Interrupt Flag Overrun Bits Check** n=0~3. 0 = None of ADINT interrupt flag ADOVIFn (EADC_STATUS2[11:8]) is overwritten to 1. 1 = Any one of ADINT interrupt flag ADOVIFn (EADC_STATUS2[11:8]) is overwritten to 1. **Note:** This bit will keep 1 when any ADOVIFn Flag is equal to 1.

Bits	Description	
[23]	**BUSY**	**Busy/Idle** 0 = EADC is in idle state. 1 = EADC is busy at conversion. **Note:** This bit is read only.
[22:21]	**Reserved**	Reserved.
[20:16]	**CHANNEL**	**Current Conversion Channel** This filed reflects ADC current conversion channel when BUSY=1. It is read only. 00H = EADC_CH0. 01H = EADC_CH1. 02H = EADC_CH2. 03H = EADC_CH3. 04H = EADC_CH4. 05H = EADC_CH5. 06H = EADC_CH6. 07H = EADC_CH7. 08H = EADC_CH8. 09H = EADC_CH9. 0AH = EADC_CH10. 0BH = EADC_CH11. 0CH = EADC_CH12. 0DH = EADC_CH13. 0EH = EADC_CH14. 0FH = EADC_CH15. 10H = VBG. 11H = VTEMP. 12H = VBAT.
[15]	**ADCMPO3**	**ADC Compare 3 Output Status** The 12 bits compare3 data CMPDAT3 (EADC_CMP3[27:16]) is used to compare with conversion result of specified sample module. User can use it to monitor the external analog input pin voltage status. 0 = Conversion result in EADC_DAT less than CMPDAT3 setting. 1 = Conversion result in EADC_DAT great than or equal CMPDAT3 setting.
[14]	**ADCMPO2**	**ADC Compare 2 Output Status** The 12 bits compare2 data CMPDAT2 (EADC_CMP2[27:16]) is used to compare with conversion result of specified sample module. User can use it to monitor the external analog input pin voltage status. 0 = Conversion result in EADC_DAT less than CMPDAT2 setting. 1 = Conversion result in EADC_DAT great than or equal CMPDAT2 setting.
[13]	**ADCMPO1**	**ADC Compare 1 Output Status** The 12 bits compare1 data CMPDAT1 (EADC_CMP1[27:16]) is used to compare with conversion result of specified sample module. User can use it to monitor the external analog input pin voltage status. 0 = Conversion result in EADC_DAT less than CMPDAT1 setting. 1 = Conversion result in EADC_DAT great than or equal CMPDAT1 setting.

Bits	Description	
[12]	**ADCMPO0**	**ADC Compare 0 Output Status** The 12 bits compare0 data CMPDAT0 (EADC_CMP0[27:16]) is used to compare with conversion result of specified sample module. User can use it to monitor the external analog input pin voltage status. 0 = Conversion result in EADC_DAT less than CMPDAT0 setting. 1 = Conversion result in EADC_DAT great than or equal CMPDAT0 setting.
[11]	**ADOVIF3**	**A/D ADINT3 Interrupt Flag Overrun** 0 = ADINT3 interrupt flag is not overwritten to 1. 1 = ADINT3 interrupt flag is overwritten to 1. **Note:** This bit is cleared by writing 1 to it.
[10]	**ADOVIF2**	**A/D ADINT2 Interrupt Flag Overrun** 0 = ADINT2 interrupt flag is not overwritten to 1. 1 = ADINT2 interrupt flag is overwritten to 1. **Note:** This bit is cleared by writing 1 to it.
[9]	**ADOVIF1**	**A/D ADINT1 Interrupt Flag Overrun** 0 = ADINT1 interrupt flag is not overwritten to 1. 1 = ADINT1 interrupt flag is overwritten to 1. **Note:** This bit is cleared by writing 1 to it.
[8]	**ADOVIF0**	**A/D ADINT0 Interrupt Flag Overrun** 0 = ADINT0 interrupt flag is not overwritten to 1. 1 = ADINT0 interrupt flag is overwritten to 1. **Note:** This bit is cleared by writing 1 to it.
[7]	**ADCMPF3**	**ADC Compare 3 Flag** When the specific sample module A/D conversion result meets setting condition in EADC_CMP3 then this bit is set to 1. 0 = Conversion result in EADC_DAT does not meet EADC_CMP3 register setting. 1 = Conversion result in EADC_DAT meets EADC_CMP3 register setting. **Note:** This bit is cleared by writing 1 to it.
[6]	**ADCMPF2**	**ADC Compare 2 Flag** When the specific sample module A/D conversion result meets setting condition in EADC_CMP2 then this bit is set to 1. 0 = Conversion result in EADC_DAT does not meet EADC_CMP2 register setting. 1 = Conversion result in EADC_DAT meets EADC_CMP2 register setting. **Note:** This bit is cleared by writing 1 to it.
[5]	**ADCMPF1**	**ADC Compare 1 Flag** When the specific sample module A/D conversion result meets setting condition in EADC_CMP1 then this bit is set to 1. 0 = Conversion result in EADC_DAT does not meet EADC_CMP1 register setting. 1 = Conversion result in EADC_DAT meets EADC_CMP1 register setting. **Note:** This bit is cleared by writing 1 to it.

Bits	Description	
[4]	**ADCMPF0**	**ADC Compare 0 Flag** When the specific sample module A/D conversion result meets setting condition in EADC_CMP0 then this bit is set to 1. 0 = Conversion result in EADC_DAT does not meet EADC_CMP0 register setting. 1 = Conversion result in EADC_DAT meets EADC_CMP0 register setting. **Note:** This bit is cleared by writing 1 to it.
[3]	**ADIF3**	**A/D ADINT3 Interrupt Flag** 0 = No ADINT3 interrupt pulse received. 1 = ADINT3 interrupt pulse has been received. **Note1:** This bit is cleared by writing 1 to it. **Note2:**This bit indicates whether an A/D conversion of specific sample module has been completed
[2]	**ADIF2**	**A/D ADINT2 Interrupt Flag** 0 = No ADINT2 interrupt pulse received. 1 = ADINT2 interrupt pulse has been received. **Note1:** This bit is cleared by writing 1 to it. **Note2:**This bit indicates whether an A/D conversion of specific sample module has been completed
[1]	**ADIF1**	**A/D ADINT1 Interrupt Flag** 0 = No ADINT1 interrupt pulse received. 1 = ADINT1 interrupt pulse has been received. **Note1:** This bit is cleared by writing 1 to it. **Note2:**This bit indicates whether an A/D conversion of specific sample module has been completed
[0]	**ADIF0**	**A/D ADINT0 Interrupt Flag** 0 = No ADINT0 interrupt pulse received. 1 = ADINT0 interrupt pulse has been received. **Note1:** This bit is cleared by writing 1 to it. **Note2:**This bit indicates whether an A/D conversion of specific sample module has been completed

A/D Status Register 3 (EADC_STATUS3)

Register	Offset	R/W	Description	Reset Value
EADC_STATUS3	EADC_BA+0xFC	R	A/D Status Register 3	0x0000_001F

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**CURSPL**				

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4:0]	**CURSPL**	**ADC Current Sample Module** This register show the current ADC is controlled by which sample module control logic modules. If the ADC is Idle, this bit filed will set to 0x1F. This is a read only register.

A/D Double Data Registers for A/D Data Registers (EADC_DDAT0~3)

Register	Offset	R/W	Description	Reset Value
EADC_DDAT0	EADC_BA+0x100	R	A/D Double Data Register 0 for Sample Module 0	0x0000_0000
EADC_DDAT1	EADC_BA+0x104	R	A/D Double Data Register 1 for Sample Module 1	0x0000_0000
EADC_DDAT2	EADC_BA+0x108	R	A/D Double Data Register 2 for Sample Module 2	0x0000_0000
EADC_DDAT3	EADC_BA+0x10C	R	A/D Double Data Register 3 for Sample Module 3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved						VALID	OV
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RESULT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RESULT							

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**VALID**	**Valid Flag** 0 = Double data in RESULT (EADC_DDATn[15:0]) is not valid. 1 = Double data in RESULT (EADC_DDATn[15:0]) is valid. This bit is set to 1 when corresponding sample module channel analog input conversion is completed and cleared by hardware after EADC_DDATn register is read. (n=0~3).
[16]	**OV**	**Overrun Flag** 0 = Data in RESULT (EADC_DATn[15:0], n=0~3) is recent conversion result. 1 = Data in RESULT (EADC_DATn[15:0], n=0~3) is overwrite. If converted data in RESULT[15:0] has not been read before new conversion result is loaded to this register, OV is set to 1. It is cleared by hardware after EADC_DDAT register is read.
[15:0]	**RESULT**	**A/D Conversion Results** This field contains 12 bits conversion results. When the DMOF (EADC_CTL[9]) is set to 0, 12-bit ADC conversion result with unsigned format will be filled in RESULT [11:0] and zero will be filled in RESULT [15:12]. When DMOF (EADC_CTL[9]) set to 1, 12-bit ADC conversion result with 2'complement format will be filled in RESULT [11:0] and signed bits to will be filled in RESULT [15:12].

6.24 Digital to Analog Converter (DAC)

6.24.1 Overview

The DAC module is a 12-bit, voltage output digital-to-analog converter. It can be used in conjunction with the PDMA controller. The DAC integrates a voltage output buffer that can be used to reduce output impendence and drive external loads directly without having to add an external operational amplifier.

6.24.2 Features

- Analog output voltage range: 0~AV_{DD}.
- Reference voltage from internal reference voltage (INT_VREF), V_{REF} pin or AV_{DD}.
- DAC maximum conversion updating rate 1M sps.
- Supports voltage output buffer mode and bypass voltage output buffer mode.
- Supports software and hardware trigger to start DAC conversion.
- Supports PDMA request.

6.24.3 Block Diagram

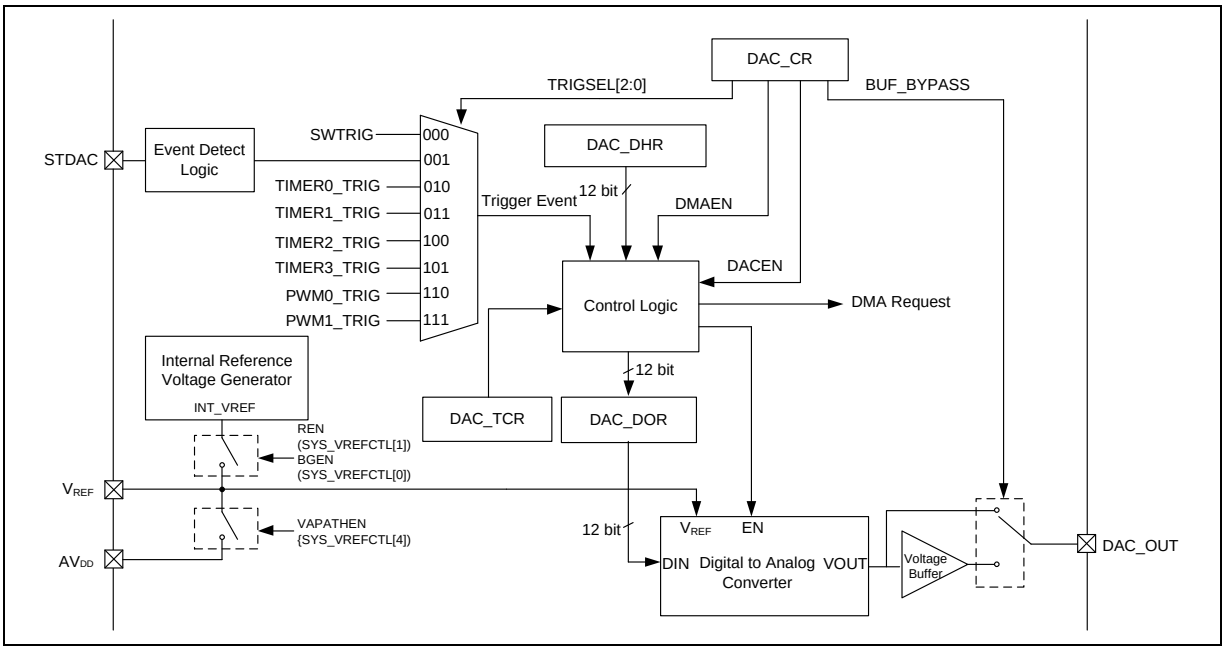


Figure 6.24-1 Digital-to-Analog Converter Block Diagram

6.24.4 Basic Configuration

The DAC output pin is configured in SYS_GPB_MFPL[3:0] Multi-function Register. The DAC Controller clock source is enabled by DACCKEN (CLK_APBCLK1[12]).

6.24.5 Functional Description

6.24.5.1 DAC Output

The DAC is a 12-bit voltage output digital-to-analog converter. The DAC integrates a voltage output buffer that can be used to reduce output impendence and drive external loads directly without having to add an external operational amplifier. The DAC channel output buffer can be enabled and disabled by BYPASS (DAC_CTL[8]). The maximum DAC output voltage is limited to the selected reference voltage source.

6.24.5.2 DAC Reference Voltage

The DAC reference voltage is shared with EADC reference voltage and it is configured by VREFCTL (SYS_VREFCTL[4:0]) in system manager control registers. The reference voltage for the DAC can be configured from external reference voltage pin (V_{REF}) or internal reference voltage generator (INT_VREF) or analog power pin (AV_{DD}).

6.24.5.3 DAC Data Format

The DAC supports conversion data left alignment or right alignment mode. Depending on the selected configuration mode, the data needs to be written into the specified register as follows:

- 12-bit left alignment: user has to load data into DAC_DAT[15:4] bits. DAC_DAT[31:16] and DAC_DAT[3:0] are ignored in DAC conversion.
- 12-bit right alignment: user has to load data into DAC_DAT[11:0] bits, DAC_DAT[31:12] are ignored in DAC conversion.

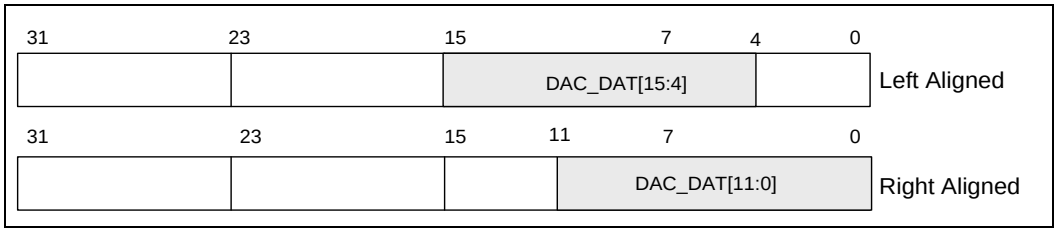


Figure 6.24-2 Data Holding Register Format

6.24.5.4 DAC Conversion

Any data transfer to the DAC channel is performed by loading the data into DAC_DAT register. Figure 6.24-3 shows the DAC conversion started by software write operation. When user writes the conversion data to data holding register DAC_DAT, the data is loaded into data output register DAC_DATOUT by hardware and DAC starts data conversion after one PCLK (APB clock) clock cycle. Figure 6.24-4 shows the DAC conversion started by hardware trigger (external pin STDAC, timer trigger event or PWM timer trigger event). The data stored in the DAC_DAT register is automatically transferred to the data output buffer DAC_DATOUT after occurring one PCLK (APB clock) the event.

When DAC data output register DAC_DATOUT is loaded with the DAC_DAT contents, the analog output voltage becomes available after specified conversion settling time. The conversion settling time is 8us when 12-bit input code transition from lowest code (0x000) to highest code (0xFFF).

Two adjacent codes conversion settling time is 1us. The DAC controller provides a 10-bit time counter for user to count the conversion time period. In continuous conversion operation, user needs to write appropriate value to SETTLET (DAC_TCTL[9:0]) to define DAC conversion time period. The value must be longer than DAC conversion settling time which is specified in DAC electric characteristic table. For example, when DAC controller APB clock speed is 72MHz and DAC conversion settling time is 8us, the selected SETTLET value must be greater than 0x241. When the conversion is started, the conversion finish flag FINISH (DAC_STATUS[0]) is cleared to 0 by hardware and set to 1 after the time counter counts to SETTLET.

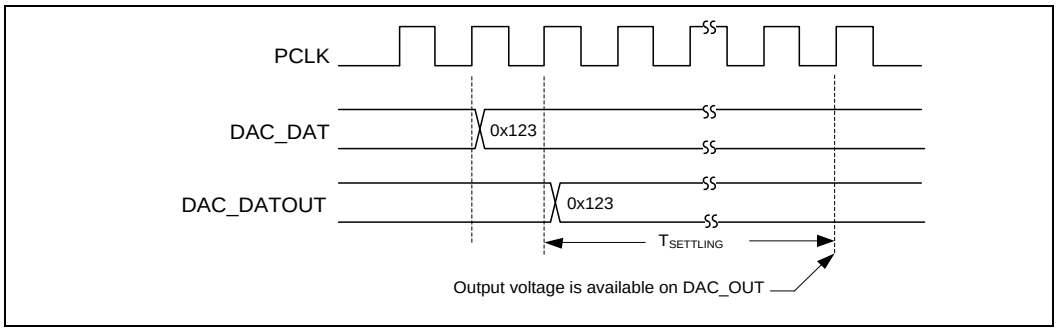


Figure 6.24-3 DAC Conversion Started by Software Write Trigger

6.24.5.5 DAC Output Voltage

Digital inputs are converted to output voltage on a linear conversion between 0 and reference voltage V_{REF}. The analog output voltage on DAC pin is determined by the following equation:

$$DAC\_OUT = V_{REF} \times DATOUT[11:0]/4096$$

6.24.5.6 DAC Trigger Selection

The DAC conversion can be started by writing DAC_DAT, software trigger or hardware trigger. When TRGEN (DAC_CTL[4]) is 0, the data conversion is started by writing DAC_DAT register. When TRGEN (DAC_CTL[4]) is 1, the data conversion is started by external STDAC pin, timer event, or PWM timer event. If the software trigger is selected, the conversion starts once the SWTRG (DAC_SWTRG[0]) is set to 1. The SWTRG is cleared to 0 by hardware automatically when DAC_DATOUT has been loaded with DAC_DAT content. The TRGSEL (DAC_CTL[7:5]) determines which one of eight events is selected to start the conversion.

When DAC detects a rising edge on the selected trigger event input, the last data stored in DAC_DAT is transferred into the DAC_DATOUT[11:0] and DAC starts converting after one PCLK (APB clock) clock cycle.

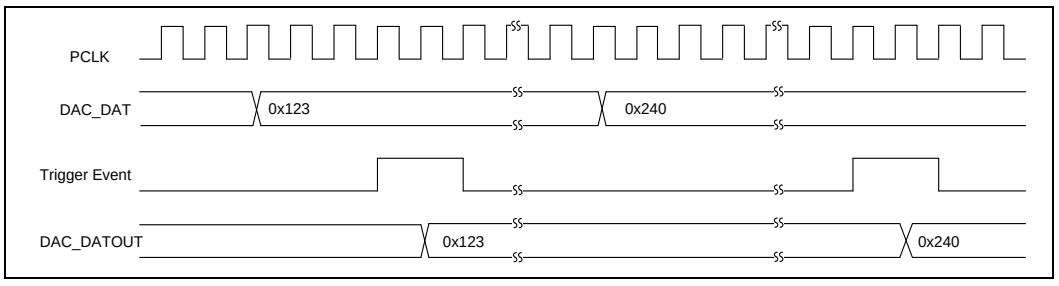


Figure 6.24-4 DAC Conversion Started by Hardware Trigger Event

6.24.5.7 DMA Operation

DAC DMA request is generated when a hardware trigger event occurs while DMAEN (DAC_CTL[2]) is set. The content of DAC_DAT is transferred to the DAC_DATOUT[11:0] and DAC starts data conversion after one PCLK (APB clock) clock cycle. The new transferred data by PDMA in DAC_DAT will be converted when next trigger event arrives. Figure 6.24-5 shows the DAC PDMA underrun condition, when the second DMA request trigger event arrives before the first conversion finish, then no new PDMA request is issued and DMA underrun flag DMAUDR (DAC_STATUS[1]) is set 1 to report the error condition. DMA data transfers are then disabled and no further DMA request is treated and DAC continues to convert last data. An interrupt is also generated if the corresponding DMAURIEN (DAC_CTL[3]) is enabled. User has to change the trigger event frequency in timer or PWM timer and then start DAC conversion again.

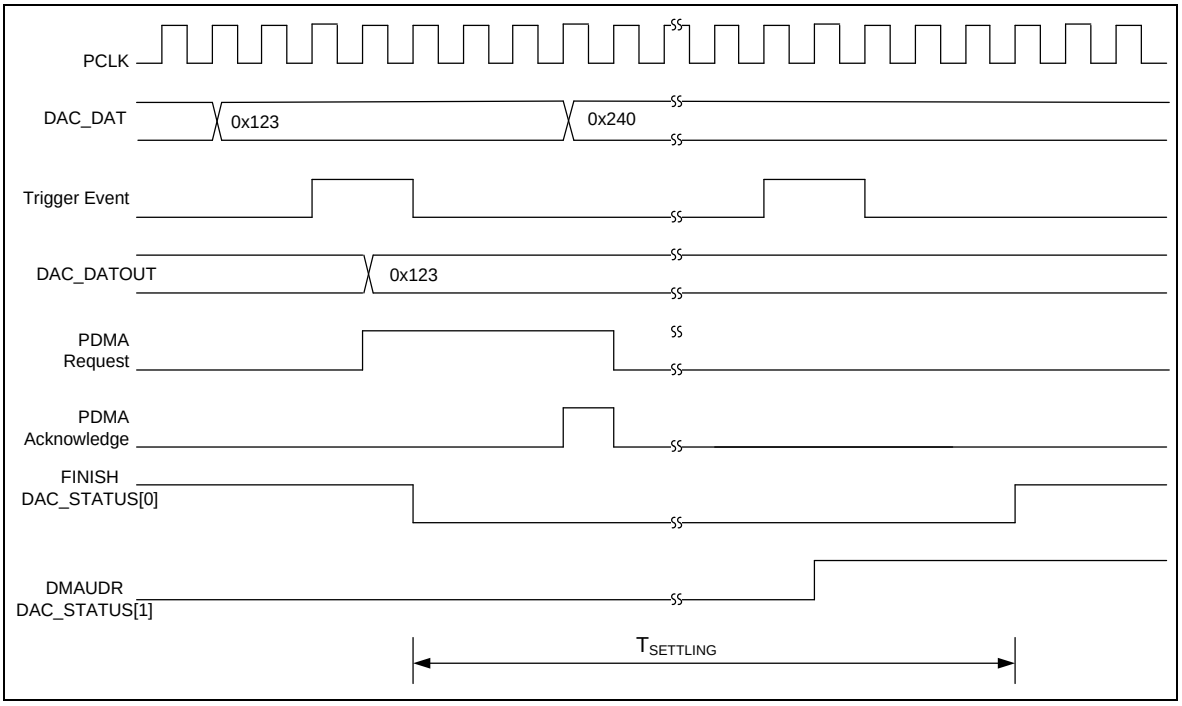


Figure 6.24-5 DAC PDMA Underrun Condition Example

DMA request can also be generated by software enable, user sets DMAEN (DAC_CTL[2]) to 1 and TRGEN (DAC_CTL[4]) to 0, DMA request is generated periodically according to the conversion time defined by SETTLET (DAC_TCTL[9:0]) value. DAC output is updated periodically. When user clears DMAEN (DAC_CTL[2]) to 0, DAC controller will stop issuing next new PDMA transfer request.

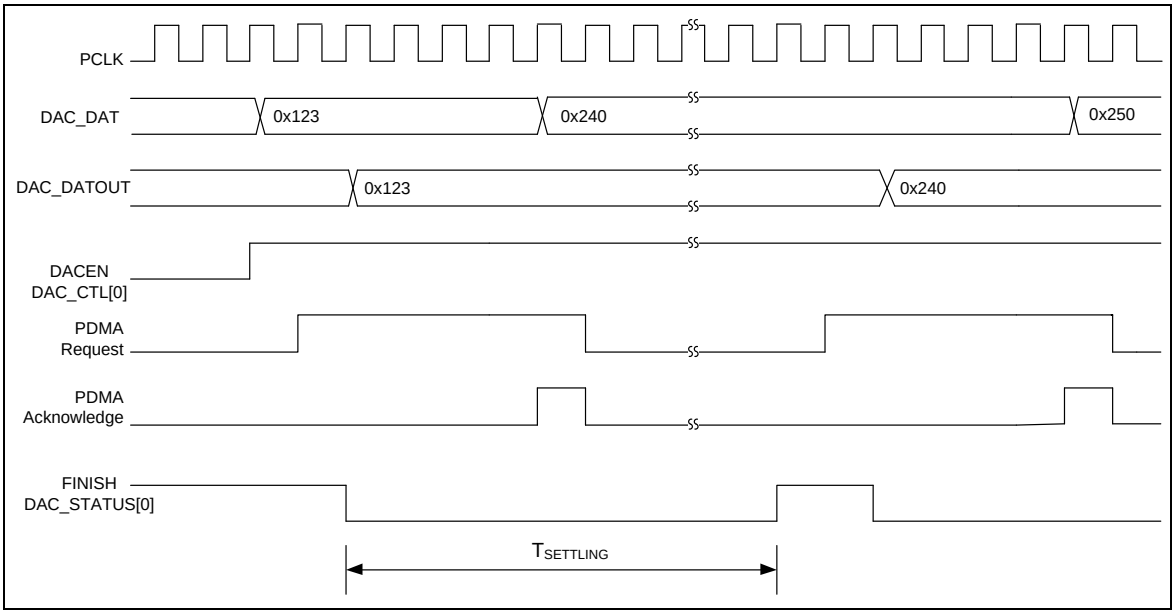


Figure 6.24-6 DAC Continuous Conversion with Software PDMA Mode

6.24.5.8 Interrupt Sources

There are two interrupt sources in DAC controller, one is DAC data conversion finish interrupt and the other is DMA under-run interrupt. When DAC conversion finish, the FINISH (DAC_STATUS[0]) is set to 1 and an interrupt occurs while DACIEN (DAC_CTL[1]) is enabled. If new DMA trigger event occurs during DAC data conversion period, the DMA under run flag DMAUDR (DAC_STATUS[1]) is generated and an interrupt occurs if DMAURIEN (DAC_CTL[3]) is enabled.

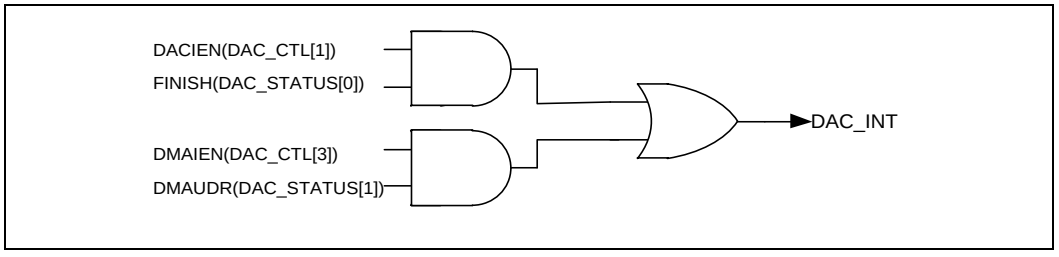


Figure 6.24-7 DAC Interrupt Source

6.24.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
DAC Base Address: **DAC_BA = 0x4004_7000**				
DAC_CTL	DAC_BA+0x00	R/W	DAC Control Register	0x0000_0000
DAC_SWTRG	DAC_BA+0x04	R/W	DAC Software Trigger Control Register	0x0000_0000
DAC_DAT	DAC_BA+0x08	R/W	DAC Data Holding Register	0x0000_0000
DAC_DATOUT	DAC_BA+0x0C	R	DAC Data Output Register	0x0000_0000
DAC_STATUS	DAC_BA+0x10	R/W	DAC Status Register	0x0000_0000
DAC_TCTL	DAC_BA+0x14	R/W	DAC Timing Control Register	0x0000_0000

6.24.7 Register Description

DAC Control Register (DAC_CTL)

Register	Offset	R/W	Description	Reset Value
DAC_CTL	DAC_BA+0x00	R/W	DAC Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		ETRGSEL		Reserved	LALIGN	Reserved	BYPASS
7	6	5	4	3	2	1	0
TRGSEL			TRGEN	DMAURIEN	DMAEN	DACIEN	DACEN

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13:12]	**ETRGSEL**	**External Pin Trigger Selection** 00 = Low level trigger. 01 = High level trigger. 10 = Falling edge trigger. 11 = Rising edge trigger.
[11]	**Reserved**	Reserved.
[10]	**LALIGN**	**DAC Data Left-aligned Enabled Control** 0 = Right alignment. 1 = Left alignment.
[9]	**Reserved**	Reserved.
[8]	**BYPASS**	**Bypass Buffer Mode** 0 = Output voltage buffer Enabled. 1 = Output voltage buffer Disabled.
[7:5]	**TRGSEL**	**Trigger Source Selection** 000 = Software trigger. 001 = External pin STDAC trigger. 010 = Timer 0 trigger. 011 = Timer 1 trigger. 100 = Timer 2 trigger. 101 = Timer 3 trigger. 110 = PWM0 trigger. 111 = PWM1 trigger.
[4]	**TRGEN**	**Trigger Mode Enable Bit** 0 = DAC event trigger mode Disabled.

		1 = DAC event trigger mode Enabled.
[3]	**DMAURIEN**	**DMA Under-run Interrupt Enable Bit** 0 = DMA underrun interrupt Disabled. 1 = DMA underrun interrupt Enabled.
[2]	**DMAEN**	**DMA Mode Enable Bit** 0 = DMA mode Disabled. 1 = DMA mode Enabled.
[1]	**DACIEN**	**DAC Interrupt Enable Bit** 0 = Interrupt is Disabled. 1 = Interrupt is Enabled.
[0]	**DACEN**	**DAC Enable Bit** 0 = DAC is Disabled. 1 = DAC is Enabled.

DAC Software Trigger Control Register (DAC_SWTRG)

Register	Offset	R/W	Description	Reset Value
DAC_SWTRG	DAC_BA+0x04	R/W	DAC Software Trigger Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved							SWTRG

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**SWTRG**	**Software Trigger** 0 = Software trigger Disabled. 1 = Software trigger Enabled. User writes this bit to generate one shot pulse and it is cleared to 0 by hardware automatically; Reading this bit will always get 0.

DAC Data Holding Register (DAC_DAT)

Register	Offset	R/W	Description	Reset Value
DAC_DAT	DAC_BA+0x08	R/W	DAC Data Holding Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
DAC_DAT							
7	6	5	4	3	2	1	0
DAC_DAT							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**DAC_DAT**	**DAC 12-bit Holding Data** These bits are written by user software which specifies 12-bit conversion data for DAC output. The unused bits (DAC_DAT[3:0] in left-alignment mode and DAC_DAT[15:12] in right alignment mode) are ignored by DAC controller hardware. 12 bit left alignment: user has to load data into DAC_DAT[15:4] bits. 12 bit right alignment: user has to load data into DAC_DAT[11:0] bits.

DAC Data Output Register (DAC_DATOUT)

Register	Offset	R/W	Description	Reset Value
DAC_DATOUT	DAC_BA+0x0C	R	DAC Data Output Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				DATOUT			
7	6	5	4	3	2	1	0
DATOUT							

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11:0]	**DATOUT**	**DAC 12-bit Output Data** These bits are current digital data for DAC output conversion. It is loaded from DAC_DAT register and user cannot write it directly.

DAC Status Register (DAC_STATUS)

Register	Offset	R/W	Description	Reset Value
DAC_STATUS	DAC_BA+0x10	R/W	DAC Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							BUSY
7	6	5	4	3	2	1	0
Reserved						DMAUDR	FINISH

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**BUSY**	**DAC Busy Flag (Read Only)** 0 = DAC is ready for next conversion. 1 = DAC is busy in conversion. This is read only bit.
[7:2]	**Reserved**	Reserved.
[1]	**DMAUDR**	**DMA Under Run Interrupt Flag** 0 = No DMA under-run error condition occurred. 1 = DMA under-run error condition occurred. User writes 1 to clear this bit.
[0]	**FINISH**	**DAC Conversion Complete Finish Flag** 0 = DAC is in conversion state. 1 = DAC conversion finish. This bit set to 1 when conversion time counter counts to SETTLET. It is cleared to 0 when DAC starts a new conversion. User writes 1 to clear this bit to 0.

DAC Timing Control Register (DAC_TCTL)

Register	Offset	R/W	Description	Reset Value
DAC_TCTL	DAC_BA+0x14	R/W	DAC Timing Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						SETTLET	
7	6	5	4	3	2	1	0
SETTLET							

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9:0]	**SETTLET**	**DAC Output Settling Time** User software needs to write appropriate value to these bits to meet DAC conversion settling time base on PCLK (APB clock) speed. For example, DAC controller clock speed is 72MHz and DAC conversion setting time is 1 us, SETTLET value must be greater than 0x48.

6.25 Analog Comparator Controller (ACMP)

6.25.1 Overview

The M4TK contains two comparators. The comparator output is logic 1 when positive input is greater than negative input; otherwise, the output is 0. Each comparator can be configured to generate an interrupt when the comparator output value changes.

6.25.2 Features

- Analog input voltage range: 0 ~ AV_{DD} (voltage of AV_{DD} pin)
- Supports hysteresis function
- Supports wake-up function
- Selectable input sources of positive input and negative input
- ACMP0 supports
  - 4 positive sources
    - ACMP0_P0, ACMP0_P1, ACMP0_P2, or ACMP0_P3
  - 4 negative sources
    - ACMP0_N
    - Comparator Reference Voltage (CRV)
    - Internal band-gap voltage (V_{BG})
    - DAC output (DAC_OUT)
- ACMP1 supports
  - 4 positive sources
    - ACMP1_P0, ACMP1_P1, ACMP1_P2, or ACMP1_P3
  - 4 negative sources
    - ACMP1_N
    - Comparator Reference Voltage (CRV)
    - Internal band-gap voltage (V_{BG})
    - DAC output (DAC_OUT)
- Shares one ACMP interrupt vector for all comparators

6.25.3 Block Diagram

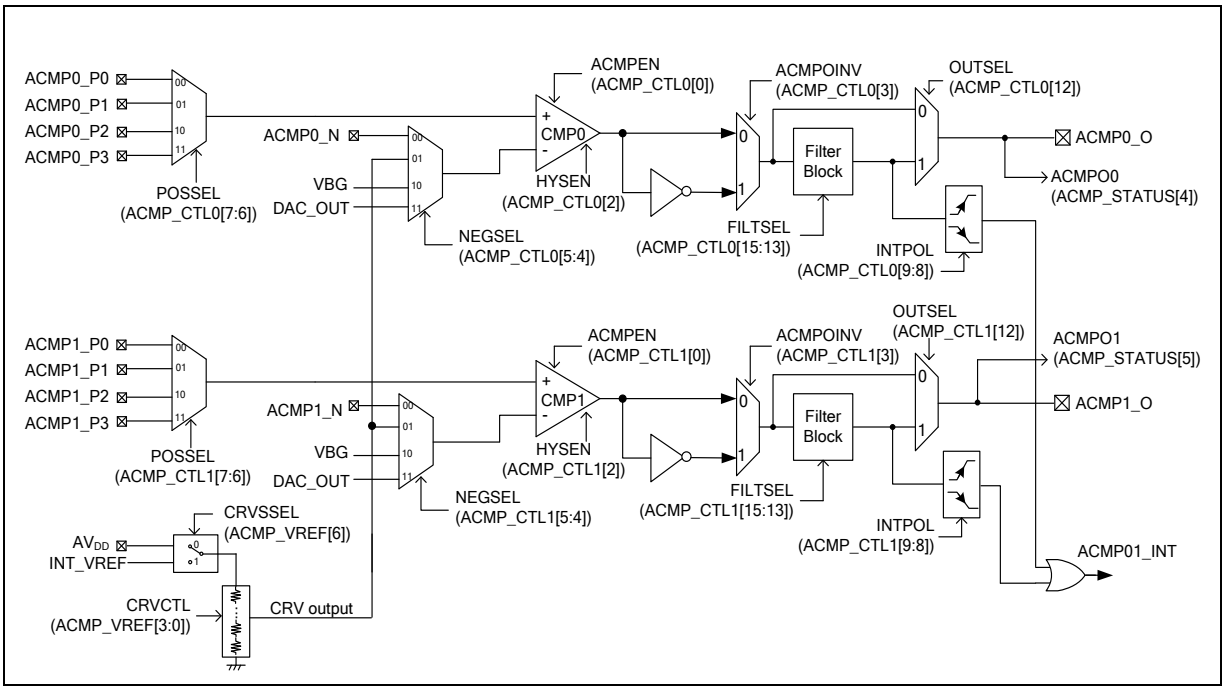


Figure 6.25-1 Analog Comparator Block Diagram

6.25.4 Basic Configuration

ACMP clock source is PCLK and can be enabled by setting ACMP01CKEN (CLK_APBCLK0[7]). The ACMP pin functions are configured in SYS_GPB_MFPL, SYS_GPB_MFPH, SYS_GPC_MFPL, SYS_GPD_MFPL and SYS_GPD_MFPH registers. It is recommended to disable the digital input path of the analog input pins to avoid the leakage current. The digital input path can be disabled by configuring PB_DINOFF and PD_DINOFF registers.

6.25.5 Functional Description

6.25.5.1 Interrupt Sources

The output of comparators is reflected at ACMPO0 (ACMP_STATUS[4]) and ACMPO1 (ACMP_STATUS[5]) . If ACMPIE of ACMP_CTLn (n: 0 ~ 1) register is set to 1, the comparator interrupt will be enabled. If the output state of comparator is changed as the setting of change condition, the comparator interrupt will be asserted and the corresponding flag, ACMPIF0 (ACMP_STATUS[0]) and ACMPIF1 (ACMP_STATUS[1]), will be set to 1. The change condition of comparator output for interrupt flag can be selected by INTPOL (ACMP_CTL0[9:8], ACMP_CTL1[9:8]). If FILTSEL (ACMP_CTL0[15:13], ACMP_CTL1[15:13]) is not set to 0, the interrupt generation is according to the filtered output. The interrupt flag can be cleared to 0 by writing 1.

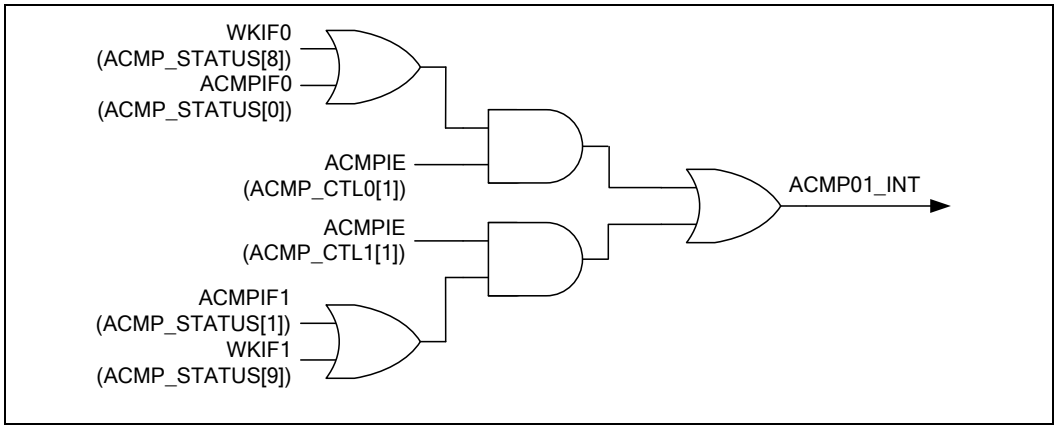


Figure 6.25-2 Comparator Controller Interrupt Sources

6.25.5.2 Hysteresis Function

The analog comparator provides the hysteresis function to make the comparator to have a stable output transition. If comparator output is 0, it will not be changed to 1 until the positive input voltage exceeds the negative input voltage by a high threshold voltage. Similarly, if comparator output is 1, it will not be changed to 0 until the positive input voltage drops lower than the negative input voltage by a low threshold voltage.

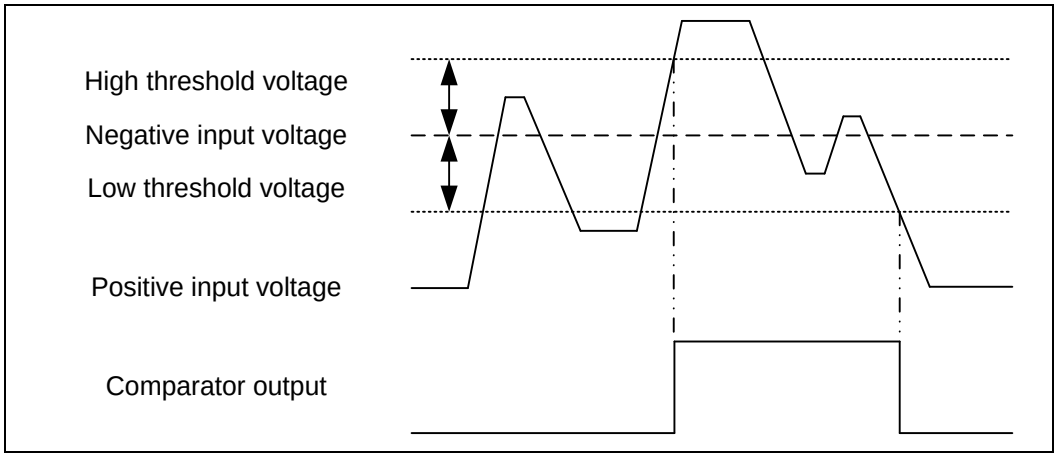


Figure 6.25-3 Comparator Hysteresis Function

6.25.5.3 Filter Function

The analog comparator provides filter function to avoid the un-stable state of comparator output.

By setting FILTSEL (ACMP_CTL0[15:13], ACMP_CTL1[15:13]), the comparator output would be sampled by consecutive PCLKs. With longer sample clocks, the comparator output would be more stable. But the sensitivity of comparator output would be reduced.

6.25.5.4 Comparator Reference Voltage (CRV)

The comparator reference voltage (CRV) module is responsible for generating reference voltage for comparators. The CRV module consists of resistor ladder and analog switch. User can set the CRV output voltage by setting CRVCTL (ACMP_VREF[3:0]). The CRV output voltage can be selected as the negative input of comparator by setting NEGSEL (ACMP_CTL0[5:4], ACMP_CTL1[5:4]).

The resistor ladder will be disabled by hardware to reduce power consumption when NEGSEL (ACMP_CTL0[5:4], ACMP_CTL1[5:4]) is not equal to 01. The reference voltage of resistor ladder can be the voltage of AV_{DD} pin or the INT_VREF voltage controlled by SYS_VREFCTL register.

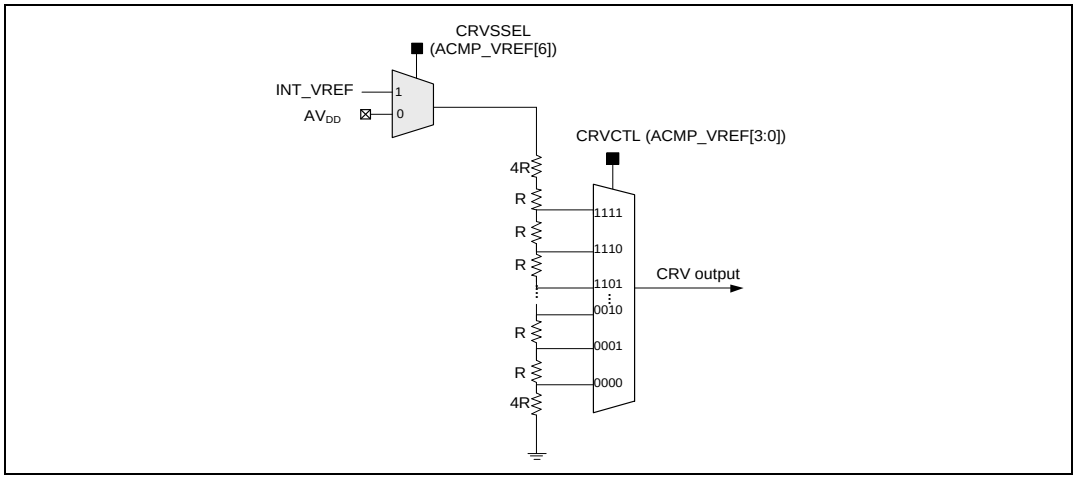


Figure 6.25-4 Comparator Reference Voltage Block Diagram

6.25.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
ACMP Base Address: **ACMP01_BA = 0x4004_5000**				
ACMP_CTL0	ACMP01_BA+0x00	R/W	Analog Comparator 0 Control Register	0x0000_0000
ACMP_CTL1	ACMP01_BA+0x04	R/W	Analog Comparator 1 Control Register	0x0000_0000
ACMP_STATUS	ACMP01_BA+0x08	R/W	Analog Comparator Status Register	0x0000_0000
ACMP_VREF	ACMP01_BA+0x0C	R/W	Analog Comparator Reference Voltage Control Register	0x0000_0000

6.25.7 Register Description

Analog Comparator 0 Control Register (ACMP_CTL0)

Register	Offset	R/W	Description	Reset Value
ACMP_CTL0	ACMP01_BA+0x00	R/W	Analog Comparator 0 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							**WKEN**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
FILTSEL			**OUTSEL**	**Reserved**		**INTPOL**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
POSSEL		**NEGSEL**		**ACMPOINV**	**HYSEN**	**ACMPIE**	**ACMPEN**

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**WKEN**	**Power-down Wake-up Enable Bit** 0 = Wake-up function Disabled. 1 = Wake-up function Enabled.
[15:13]	**FILTSEL**	**Comparator Output Filter Count Selection** 000 = Filter function is Disabled. 001 = ACMP0 output is sampled 1 consecutive PCLK. 010 = ACMP0 output is sampled 2 consecutive PCLKs. 011 = ACMP0 output is sampled 4 consecutive PCLKs. 100 = ACMP0 output is sampled 8 consecutive PCLKs. 101 = ACMP0 output is sampled 16 consecutive PCLKs. 110 = ACMP0 output is sampled 32 consecutive PCLKs. 111 = ACMP0 output is sampled 64 consecutive PCLKs.
[12]	**OUTSEL**	**Comparator Output Select** 0 = Comparator 0 output to ACMP0_O pin is unfiltered comparator output. 1 = Comparator 0 output to ACMP0_O pin is from filter output.
[11:10]	**Reserved**	Reserved.
[9:8]	**INTPOL**	**Interrupt Condition Polarity Selection** ACMPIF0 will be set to 1 when comparator output edge condition is detected. 00 = Rising edge or falling edge. 01 = Rising edge. 10 = Falling edge. 11 = Reserved.
[7:6]	**POSSEL**	**Comparator Positive Input Selection** 00 = Input from ACMP0_P0.

		01 = Input from ACMP0_P1. 10 = Input from ACMP0_P2. 11 = Input from ACMP0_P3.
[5:4]	**NEGSEL**	**Comparator Negative Input Selection** 00 = ACMP0_N pin. 01 = Internal comparator reference voltage (CRV). 10 = Band-gap voltage. 11 = DAC output.
[3]	**ACMPOINV**	**Comparator Output Inverse** 0 = Comparator 0 output inverse Disabled. 1 = Comparator 0 output inverse Enabled.
[2]	**HYSEN**	**Comparator Hysteresis Enable Bit** 0 = Comparator 0 hysteresis Disabled. 1 = Comparator 0 hysteresis Enabled.
[1]	**ACMPIE**	**Comparator Interrupt Enable Bit** 0 = Comparator 0 interrupt Disabled. 1 = Comparator 0 interrupt Enabled. If WKEN (ACMP_CTL0[16]) is set to 1, the wake-up interrupt function will be enabled as well.
[0]	**ACMPEN**	**Comparator Enable Bit** 0 = Comparator 0 Disabled. 1 = Comparator 0 Enabled.

Analog Comparator 1 Control Register (ACMP_CTL1)

Register	Offset	R/W	Description	Reset Value
ACMP_CTL1	ACMP01_BA+0x04	R/W	Analog Comparator 1 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							WKEN
15	14	13	12	11	10	9	8
FILTSEL			OUTSEL	Reserved		INTPOL	
7	6	5	4	3	2	1	0
POSSEL		NEGSEL		ACMPOINV	HYSEN	ACMPIE	ACMPEN

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**WKEN**	**Power-down Wakeup Enable Bit** 0 = Wake-up function Disabled. 1 = Wake-up function Enabled.
[15:13]	**FILTSEL**	**Comparator Output Filter Count Selection** 000 = Filter function is Disabled. 001 = ACMP1 output is sampled 1 consecutive PCLK. 010 = ACMP1 output is sampled 2 consecutive PCLKs. 011 = ACMP1 output is sampled 4 consecutive PCLKs. 100 = ACMP1 output is sampled 8 consecutive PCLKs. 101 = ACMP1 output is sampled 16 consecutive PCLKs. 110 = ACMP1 output is sampled 32 consecutive PCLKs. 111 = ACMP1 output is sampled 64 consecutive PCLKs.
[12]	**OUTSEL**	**Comparator Output Select** 0 = Comparator 1 output to ACMP1_O pin is unfiltered comparator output. 1 = Comparator 1 output to ACMP1_O pin is from filter output.
[11:10]	**Reserved**	Reserved.
[9:8]	**INTPOL**	**Interrupt Condition Polarity Selection** ACMPIF1 will be set to 1 when comparator output edge condition is detected. 00 = Rising edge or falling edge. 01 = Rising edge. 10 = Falling edge. 11 = Reserved.
[7:6]	**POSSEL**	**Comparator Positive Input Selection** 00 = Input from ACMP1_P0. 01 = Input from ACMP1_P1.

Bits	Description	
		10 = Input from ACMP1_P2. 11 = Input from ACMP1_P3.
[5:4]	**NEGSEL**	**Comparator Negative Input Selection** 00 = ACMP1_N pin. 01 = Internal comparator reference voltage (CRV). 10 = Band-gap voltage. 11 = DAC output.
[3]	**ACMPOINV**	**Comparator Output Inverse Control** 0 = Comparator 1 output inverse Disabled. 1 = Comparator 1 output inverse Enabled.
[2]	**HYSEN**	**Comparator Hysteresis Enable Bit** 0 = Comparator 1 hysteresis Disabled. 1 = Comparator 1 hysteresis Enabled.
[1]	**ACMPIE**	**Comparator Interrupt Enable Bit** 0 = Comparator 1 interrupt Disabled. 1 = Comparator 1 interrupt Enabled. If WKEN (ACMP_CTL1[16]) is set to 1, the wake-up interrupt function will be enabled as well.
[0]	**ACMPEN**	**Comparator Enable Bit** 0 = Comparator 1 Disabled. 1 = Comparator 1 Enabled.

Analog Comparator Status Register (ACMP_STATUS)

Register	Offset	R/W	Description	Reset Value
ACMP_STATUS	ACMP01_BA+0x08	R/W	Analog Comparator Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved						WKIF1	WKIF0
7	6	5	4	3	2	1	0
Reserved		ACMPO1	ACMPO0	Reserved		ACMPIF1	ACMPIF0

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**WKIF1**	**Comparator 1 Power-down Wake-up Interrupt Flag** This bit will be set to 1 when ACMP1 wake-up interrupt event occurs. 0 = No power-down wake-up occurred. 1 = Power-down wake-up occurred. **Note:** Write 1 to clear this bit to 0.
[8]	**WKIF0**	**Comparator 0 Power-down Wake-up Interrupt Flag** This bit will be set to 1 when ACMP0 wake-up interrupt event occurs. 0 = No power-down wake-up occurred. 1 = Power-down wake-up occurred. **Note:** Write 1 to clear this bit to 0.
[7:6]	**Reserved**	Reserved.
[5]	**ACMPO1**	**Comparator 1 Output** Synchronized to the PCLK to allow reading by software. Cleared when the comparator 1 is disabled, i.e. ACMPEN (ACMP_CTL1[0]) is cleared to 0.
[4]	**ACMPO0**	**Comparator 0 Output** Synchronized to the PCLK to allow reading by software. Cleared when the comparator 0 is disabled, i.e. ACMPEN (ACMP_CTL0[0]) is cleared to 0.
[3:2]	**Reserved**	Reserved.
[1]	**ACMPIF1**	**Comparator 1 Interrupt Flag** This bit is set by hardware when the edge condition defined by INTPOL (ACMP_CTL1[9:8]) is detected on comparator 1 output. This will cause an interrupt if ACMPIE (ACMP_CTL1[1]) is set to 1. **Note:** Write 1 to clear this bit to 0.
[0]	**ACMPIF0**	**Comparator 0 Interrupt Flag** This bit is set by hardware when the edge condition defined by INTPOL (ACMP_CTL0[9:8]) is detected on comparator 0 output. This will generate an interrupt if

Bits	Description	
		ACMPIE (ACMP_CTL0[1]) is set to 1. **Note:** Write 1 to clear this bit to 0.

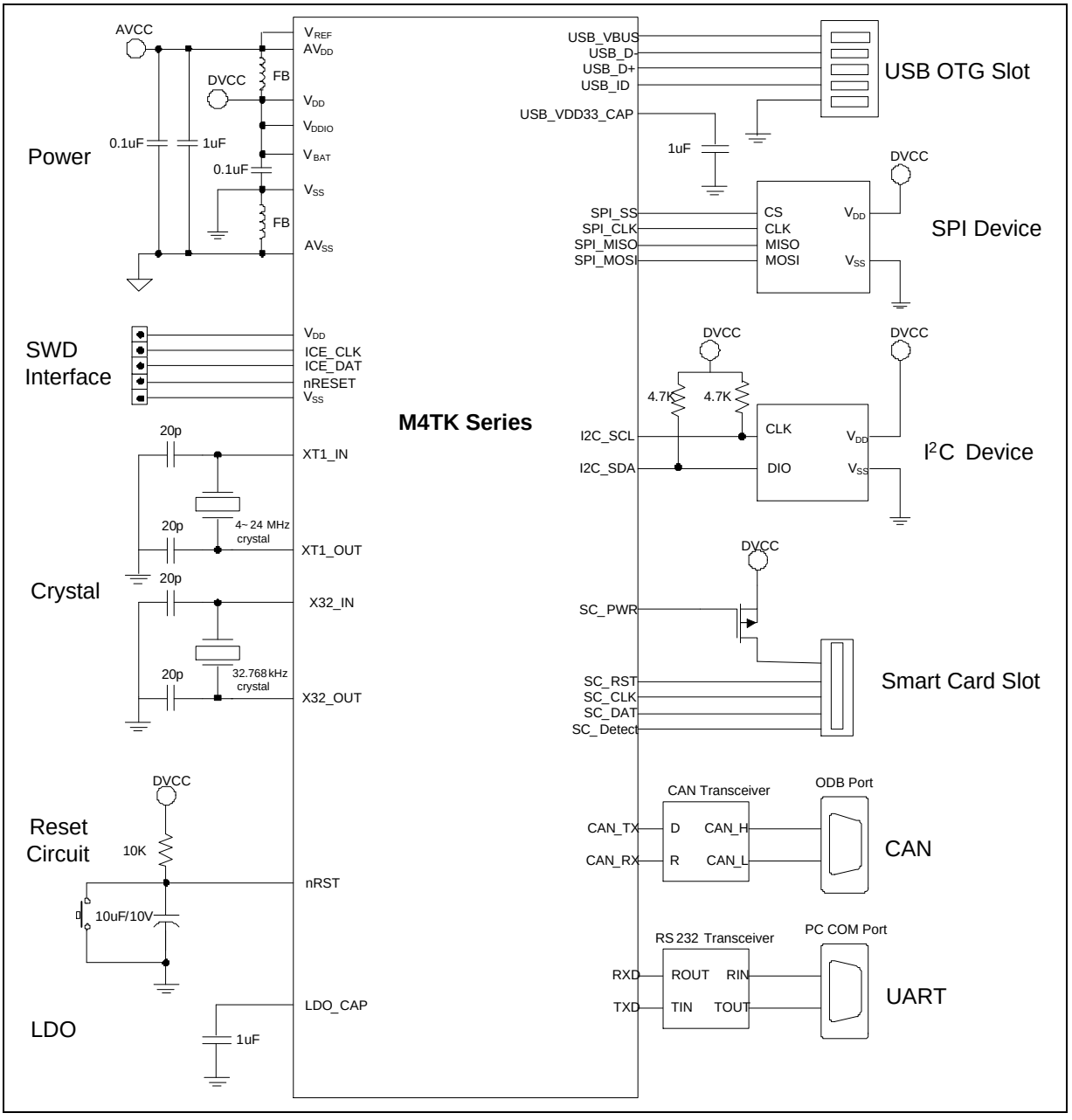
ACMP Reference Voltage Control Register (ACMP_VREF)

Register	Offset	R/W	Description	Reset Value
ACMP_VREF	ACMP01_BA+0x0C	R/W	Analog Comparator Reference Voltage Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**CRVSSEL**	**Reserved**		**CRVCTL**			

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**CRVSSEL**	**CRV Source Voltage Selection** 0 = V_{DDA} is selected as CRV source voltage. 1 = The reference voltage defined by SYS_VREFCTL register is selected as CRV source voltage.
[5:4]	**Reserved**	Reserved.
[3:0]	**CRVCTL**	**Comparator Reference Voltage Setting** CRV = CRV source voltage * (1/6+CRVCTL/24).

7 APPLICATION CIRCUIT

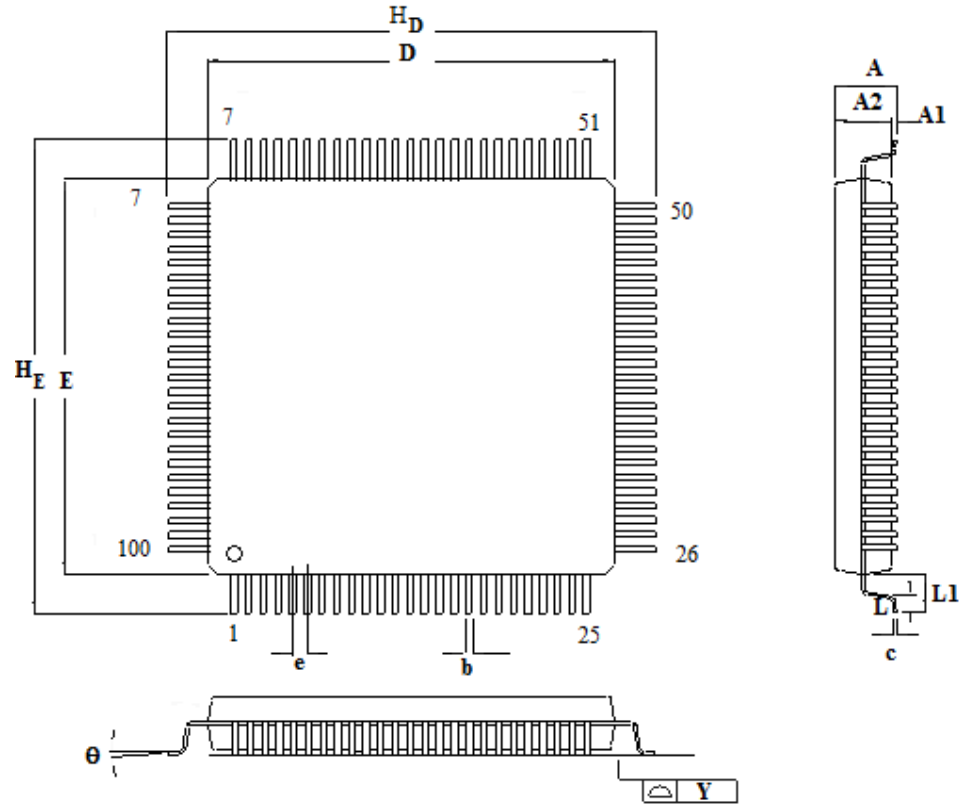


8 ELECTRICAL CHARACTERISTICS

For information on the M4TK series electrical characteristics, please refer to NuMicro® M4TK Series Datasheet.

9 PACKAGE DIMENSIONS

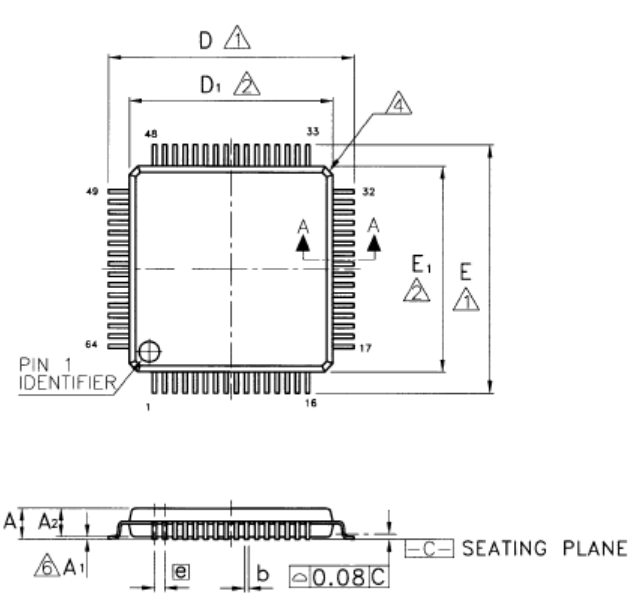
9.1 LQFP 100L (14x14x1.4 mm footprint 2.0 mm)

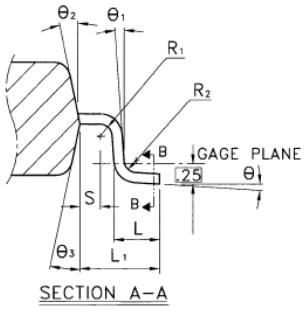


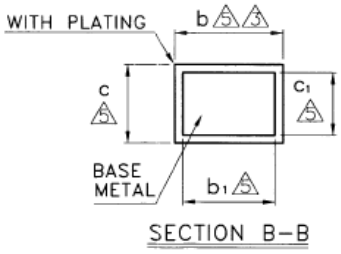
Controlling Dimension : Millimeters

SYMBOLS	MIN.	NOM.	MAX.
A	-	-	1.60
A1	0.05	-	-
A2	1.35	1.40	1.45
b	0.17	0.22	0.27
c	0.10	0.15	0.20
D	13.90	14.00	14.10
E	13.90	14.00	14.10
e	-	0.50	-
HD	15.80	16.00	16.20
HE	15.80	16.00	16.20
L	0.45	0.60	0.75
L1	-	1	-
y	-	-	0.10
θ	0°	-	7°

9.2 LQFP 64L (10x10x1.4 mm footprint 2.0 mm)

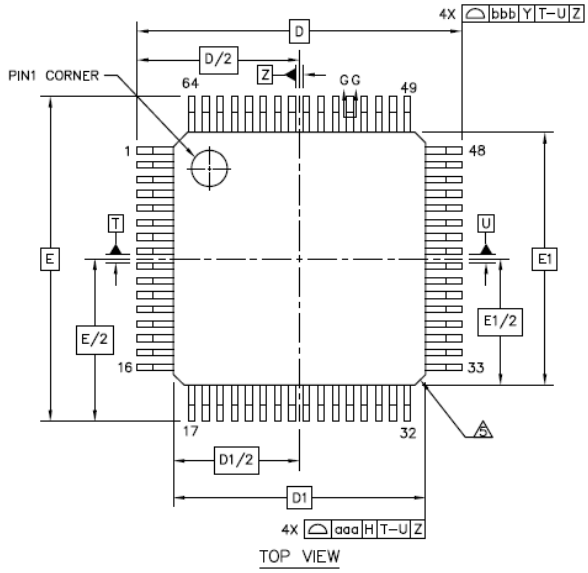


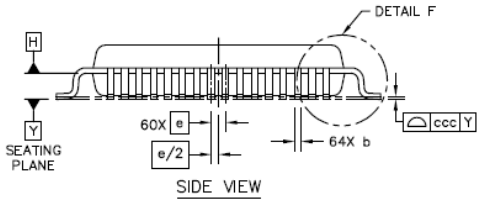




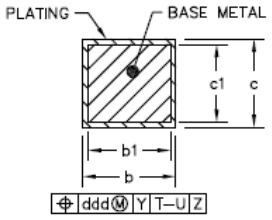
Symbol	Dimension in mm			Dimension in inch		
	Min	Nom	Max	Min	Nom	Max
A	—	—	1.60	—	—	0.063
A1	0.05	—	0.15	0.002	—	0.006
A2	1.35	1.40	1.45	0.053	0.055	0.057
b	0.17	0.22	0.27	0.007	0.009	0.011
b1	0.17	0.20	0.23	0.007	0.008	0.009
c	0.09	—	0.20	0.004	—	0.008
c1	0.09	—	0.16	0.004	—	0.006
D	12.00 BSC			0.472 BSC		
D1	10.00 BSC			0.394 BSC		
E	12.00 BSC			0.472 BSC		
E1	10.00 BSC			0.394 BSC		
e	0.50 BSC			0.020 BSC		
L	0.45	0.60	0.75	0.018	0.024	0.030
L1	1.00 REF			0.039 REF		
R1	0.08	—	—	0.003	—	—
R2	0.08	—	0.20	0.003	—	0.008
S	0.20	—	—	0.008	—	—
θ	0°	3.5°	7°	0°	3.5°	7°
θ1	0°	—	—	0°	—	—
θ2	12°TYP			12°TYP		
θ3	12°TYP			12°TYP		

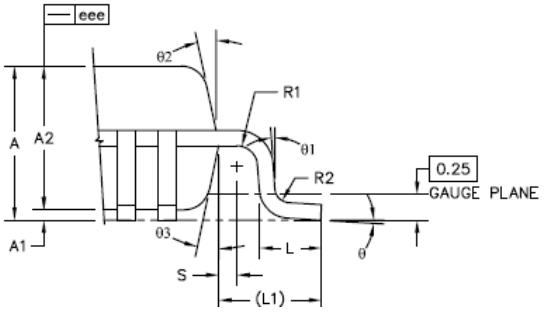
9.3 LQFP 64L (7x7x1.4 mm footprint 2.0 mm)



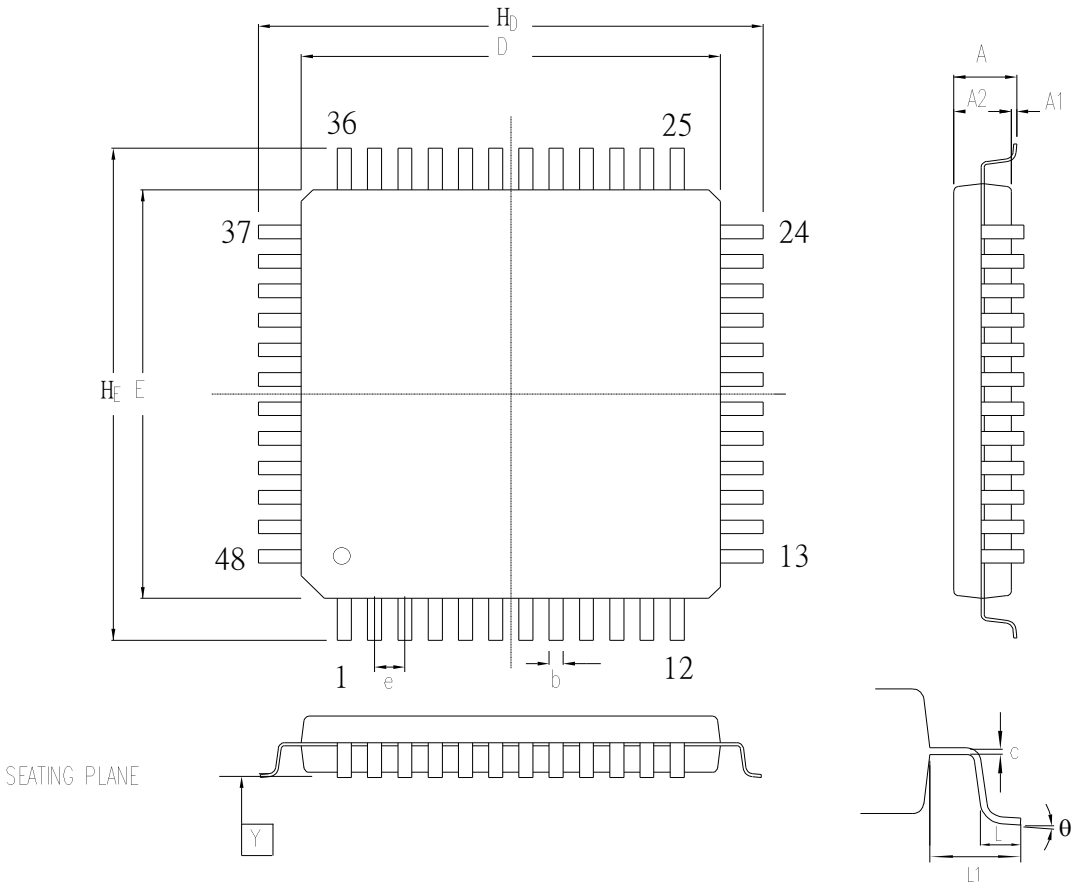


		SYMBOL	MIN	NOM	MAX
TOTAL THICKNESS		A	---	---	1.6
STAND OFF		A1	0.05	---	0.15
MOLD THICKNESS		A2	1.35	1.4	1.45
LEAD WIDTH(PLATING)		b	0.13	0.18	0.23
LEAD WIDTH		b1	0.13	0.16	0.19
L/F THICKNESS(PLATING)		c	0.09	---	0.2
L/F THICKNESS		c1	0.09	---	0.16
BODY SIZE	X	D	9 BSC		
	Y	E	9 BSC		
	X	D1	7 BSC		
	Y	E1	7 BSC		
LEAD PITCH		e	0.4 BSC		
		L	0.45	0.6	0.75
FOOTPRINT		L1	1 REF		
		θ	0°	3.5°	7°
		θ1	0°	---	---
		θ2	11°	12°	13°
		θ3	11°	12°	13°
		R1	0.08	---	---
		R2	0.08	---	0.2
		S	0.2	---	---
PACKAGE EDGE TOLERANCE		aaa	0.2		
LEAD EDGE TOLERANCE		bbb	0.2		
COPLANARITY		ccc	0.08		
LEAD OFFSET		ddd	0.07		
MOLD FLATNESS		eee	0.05		





9.4 LQFP 48L (7x7x1.4mm footprint 2.0mm)



Controlling dimension : Millimeters

Symbol	Dimension in inch			Dimension in mm		
	Min	Nom	Max	Min	Nom	Max
A	—	—	—	—	—	—
A_1	0.002	0.004	0.006	0.05	0.10	0.15
A_2	0.053	0.055	0.057	1.35	1.40	1.45
b	0.006	0.008	0.010	0.15	0.20	0.25
c	0.004	0.006	0.008	0.10	0.15	0.20
D	0.272	0.276	0.280	6.90	7.00	7.10
E	0.272	0.276	0.280	6.90	7.00	7.10
e	0.014	0.020	0.026	0.35	0.50	0.65
H_D	0.350	0.354	0.358	8.90	9.00	9.10
H_E	0.350	0.354	0.358	8.90	9.00	9.10
L	0.018	0.024	0.030	0.45	0.60	0.75
L_1	—	0.039	—	—	1.00	—
Y	—	—	0.004	—	—	0.10
θ	0	—	7	0	—	7

10 REVISION HISTORY

Date	Revision	Description
2016.01.06	1.00	Preliminary version.

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.