ARM Cortex™-M0

32-BIT MICROCONTROLLER

NuMicro™ NUC100 Series
NUC130/NUC140
Technical Reference Manual

Contents

Figures

Tables

1 GENERAL DESCRIPTION

The NuMicro™ NUC100 Series is 32-bit microcontrollers with embedded ARM® Cortex™-M0 core for industrial control and applications which need rich communication interfaces. The Cortex™-M0 is the newest ARM® embedded processor with 32-bit performance and at a cost equivalent to traditional 8-bit microcontroller. NuMicro™ NUC100 Series includes NUC100, NUC120, NUC130 and NUC140 product line.

The NuMicro™ NUC130 Automotive Line with CAN function embeds Cortex™-M0 core running up to 50 MHz with 32K/64K/128K-byte embedded flash, 4K/8K/16K-byte embedded SRAM, and 4K-byte loader ROM for the ISP. It also equips with plenty of peripheral devices, such as Timers, Watchdog Timer, RTC, PDMA, UART, SPI, I^{2}C, I^{2}S, PWM Timer, GPIO, LIN, CAN, PS/2, 12-bit ADC, Analog Comparator, Low Voltage Reset Controller and Brown-out Detector.

The NuMicro™ NUC140 Connectivity Line with USB 2.0 full-speed and CAN functions embeds Cortex™-M0 core running up to 50 MHz with 32K/64K/128K-byte embedded flash, 4K/8K/16K-byte embedded SRAM, and 4K-byte loader ROM for the ISP.. It also equips with plenty of peripheral devices, such as Timers, Watchdog Timer, RTC, PDMA, UART, SPI, I^{2}C, I^{2}S, PWM Timer, GPIO, LIN, CAN, PS/2, USB 2.0 FS Device, 12-bit ADC, Analog Comparator, Low Voltage Reset Controller and Brown-out Detector.

Product Line	UART	SPI	I^{2}C	USB	LIN	CAN	PS/2	I^{2}S
NUC100	●	●	●				●	●
NUC120	●	●	●	●			●	●
NUC130	●	●	●		●	●	●	●
NUC140	●	●	●	●	●	●	●	●

Table 1-1 Connectivity Supported Table

2 FEATURES

The equipped features are dependent on the product line and their sub products.

2.1 NuMicro™ NUC130 Features – Automotive Line

- Core
  - ARM® Cortex™-M0 core runs up to 50 MHz
  - One 24-bit system timer
  - Supports low power sleep mode
  - Single-cycle 32-bit hardware multiplier
  - NVIC for the 32 interrupt inputs, each with 4-levels of priority
  - Serial Wire Debug supports with 2 watchpoints/4 breakpoints
- Build-in LDO for wide operating voltage ranges from 2.5 V to 5.5 V
- Flash Memory
  - 32K/64K/128K bytes Flash for program code
  - 4KB flash for ISP loader
  - Support In-system program (ISP) application code update
  - 512 byte page erase for flash
  - Configurable data flash address and size for 128KB system, fixed 4KB data flash for the 32KB and 64KB system
  - Support 2 wire ICP update through SWD/ICE interface
  - Support fast parallel programming mode by external programmer
- SRAM Memory
  - 4K/8K/16K bytes embedded SRAM
  - Support PDMA mode
- PDMA (Peripheral DMA)
  - Support 9 channels PDMA for automatic data transfer between SRAM and peripherals
- Clock Control
  - Flexible selection for different applications
  - Built-in 22.1184 MHz high speed OSC for system operation
    - Trimmed to ± 1 % at +25 ℃ and V_{DD} = 5 V
    - Trimmed to ± 3 % at -40 ℃ ~ +85 ℃ and V_{DD} = 2.5 V ~ 5.5 V
  - Built-in 10 KHz low speed OSC for Watchdog Timer and Wake-up operation
  - Support one PLL, up to 50 MHz, for high performance system operation
  - External 4~24 MHz high speed crystal input for precise timing operation
  - External 32.768 kHz low speed crystal input for RTC function and low power system operation
- GPIO
  - Four I/O modes:
    - Quasi bi-direction
    - Push-Pull output
    - Open-Drain output
    - Input only with high impendence
  - TTL/Schmitt trigger input selectable
  - I/O pin can be configured as interrupt source with edge/level setting
  - High driver and high sink IO mode support
- Timer

- Support 4 sets of 32-bit timers with 24-bit up-timer and one 8-bit pre-scale counter
- Independent clock source for each timer
- Provides one-shot, periodic, toggle and continuous counting operation modes
- Support event counting function
- Support input capture function

- Watchdog Timer
  - Multiple clock sources
  - 8 selectable time out period from 1.6ms ~ 26.0sec (depends on clock source)
  - WDT can wake-up from power down or idle mode
  - Interrupt or reset selectable on watchdog time-out
- RTC
  - Support software compensation by setting frequency compensate register (FCR)
  - Support RTC counter (second, minute, hour) and calendar counter (day, month, year)
  - Support Alarm registers (second, minute, hour, day, month, year)
  - Selectable 12-hour or 24-hour mode
  - Automatic leap year recognition
  - Support 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
  - Support wake-up function
- PWM/Capture
  - Built-in up to four 16-bit PWM generators provide eight PWM outputs or four complementary paired PWM outputs
  - Each PWM generator equipped with one clock source selector, one clock divider, one 8-bit prescaler and one Dead-Zone generator for complementary paired PWM
  - Up to eight 16-bit digital Capture timers (shared with PWM timers) provide eight rising/falling capture inputs
  - Support Capture interrupt
- UART
  - Up to three UART controllers
  - UART ports with flow control (TXD, RXD, CTS and RTS)
  - UART0 with 64-byte FIFO is for high speed
  - UART1/2(optional) with 16-byte FIFO for standard device
  - Support IrDA (SIR) and LIN function
  - Support RS-485 9-bit mode and direction control.
  - Programmable baud-rate generator up to 1/16 system clock
  - Support PDMA mode
- SPI
  - Up to four sets of SPI controller
  - Master up to 32 MHz, and Slave up to 10 MHz (chip working @ 5V)
  - Support SPI master/slave mode
  - Full duplex synchronous serial data transfer
  - Variable length of transfer data from 1 to 32 bits
  - MSB or LSB first data transfer
  - Rx and Tx on both rising or falling edge of serial clock independently
  - 2 slave/device select lines when it is as the master, and 1 slave/device select line when it is as the slave
  - Support byte suspend mode in 32-bit transmission
  - Support PDMA mode
  - Support three wire, no slave select signal, bi-direction interface

- I^{2}C
  - Up to two sets of I^{2}C device
  - 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
  - Support multiple address recognition (four slave address with mask option)
- I^{2}S
  - Interface with external audio CODEC
  - Operate as either master or slave mode
  - Capable of handling 8-, 16-, 24- and 32-bit word sizes
  - Mono and stereo audio data supported
  - I^{2}S and MSB justified data format supported
  - Two 8 word FIFO data buffers are provided, one for transmit and one for receive
  - Generates interrupt requests when buffer levels cross a programmable boundary
  - Support two DMA requests, one for transmit and one for receive
- PS/2 Device Controller
  - Host communication inhibit and request to send detection
  - Reception frame error detection
  - Programmable 1 to 16 bytes transmit buffer to reduce CPU intervention
  - Double buffer for data reception
  - S/W override bus
- CAN 2.0
  - Supports CAN protocol version 2.0 part A and B
  - Bit rates up to 1M bit/s
  - 32 Message Objects
  - Each Message Object has its won identifier mask
  - Programmable FIFO mode (concatenation of Message Object)
  - Maskable interrupt
  - Disabled Automatic Re-transmission mode for Time Triggered CAN applications
  - Support power down wake-up function
- EBI (External bus interface) support (100-pin and 64-pin Package Only)
  - Accessible space: 64KB in 8-bit mode or 128KB in 16-bit mode
  - Support 8-/16-bit data width
  - Support byte write in 16-bit data width mode
- ADC
  - 12-bit SAR ADC with 700K SPS
  - Up to 8-ch single-end input or 4-ch differential input
  - Single scan/single cycle scan/continuous scan
  - Each channel with individual result register
  - Scan on enabled channels
  - Threshold voltage detection
  - Conversion start by software programming or external input

- Support PDMA mode

- Analog Comparator
  - Up to two analog comparator
  - External input or internal bandgap voltage selectable at negative node
  - Interrupt when compare result change
  - Power down wake-up
- One built-in temperature sensor with 1℃ resolution
- Brown-Out detector
  - With 4 levels: 4.5 V/3.8 V/2.7 V/2.2 V
  - Support Brown-Out Interrupt and Reset option
- Low Voltage Reset
  - Threshold voltage levels: 2.0 V
- Operating Temperature: -40℃~85℃
- Packages:
  - All Green package (RoHS)
  - LQFP 100-pin / 64-pin / 48-pin

2.2 NuMicro™ NUC140 Features – Connectivity Line

- Core
  - ARM® Cortex™-M0 core runs up to 50 MHz
  - One 24-bit system timer
  - Supports low power sleep mode
  - Single-cycle 32-bit hardware multiplier
  - NVIC for the 32 interrupt inputs, each with 4-levels of priority
  - Serial Wire Debug supports with 2 watchpoints/4 breakpoints
- Build-in LDO for wide operating voltage ranges from 2.5 V to 5.5 V
- Flash Memory
  - 32K/64K/128K bytes Flash for program code
  - 4KB flash for ISP loader
  - Support In-system program (ISP) application code update
  - 512 byte page erase for flash
  - Configurable data flash address and size for 128KB system, fixed 4KB data flash for the 32KB and 64KB system
  - Support 2 wire ICP update through SWD/ICE interface
  - Support fast parallel programming mode by external programmer
- SRAM Memory
  - 4K/8K/16K bytes embedded SRAM
  - Support PDMA mode
- PDMA (Peripheral DMA)
  - Support 9 channels PDMA for automatic data transfer between SRAM and peripherals
- Clock Control
  - Flexible selection for different applications
  - Built-in 22.1184 MHz high speed OSC for system operation
    - Trimmed to ± 1 % at +25 ℃ and V_{DD} = 5 V
    - Trimmed to ± 3 % at -40 ℃ ~ +85 ℃ and V_{DD} = 2.5 V ~ 5.5 V
  - Built-in 10 KHz low speed OSC for Watchdog Timer and Wake-up operation
  - Support one PLL, up to 50 MHz, for high performance system operation
  - External 4~24 MHz high speed crystal input for USB and precise timing operation
  - External 32.768 kHz low speed crystal input for RTC function and low power system operation
- GPIO
  - Four I/O modes:
    - Quasi bi-direction
    - Push-Pull output
    - Open-Drain output
    - Input only with high impendence
  - TTL/Schmitt trigger input selectable
  - I/O pin can be configured as interrupt source with edge/level setting
  - High driver and high sink IO mode support
- Timer
  - Support 4 sets of 32-bit timers with 24-bit up-timer and one 8-bit pre-scale counter
  - Independent clock source for each timer

- Provides one-shot, periodic, toggle and continuous counting operation modes
- Support event counting function
- Support input capture function

- Watchdog Timer
  - Multiple clock sources
  - 8 selectable time out period from 1.6ms ~ 26.0sec (depends on clock source)
  - WDT can wake-up from power down or idle mode
  - Interrupt or reset selectable on watchdog time-out
- RTC
  - Support software compensation by setting frequency compensate register (FCR)
  - Support RTC counter (second, minute, hour) and calendar counter (day, month, year)
  - Support Alarm registers (second, minute, hour, day, month, year)
  - Selectable 12-hour or 24-hour mode
  - Automatic leap year recognition
  - Support 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
  - Support wake-up function
- PWM/Capture
  - Built-in up to four 16-bit PWM generators provide eight PWM outputs or four complementary paired PWM outputs
  - Each PWM generator equipped with one clock source selector, one clock divider, one 8-bit prescaler and one Dead-Zone generator for complementary paired PWM
  - Up to eight 16-bit digital Capture timers (shared with PWM timers) provide eight rising/falling capture inputs
  - Support Capture interrupt
- UART
  - Up to three UART controllers
  - UART ports with flow control (TXD, RXD, CTS and RTS)
  - UART0 with 64-byte FIFO is for high speed
  - UART1/2(optional) with 16-byte FIFO for standard device
  - Support IrDA (SIR) and LIN function
  - Support RS-485 9-bit mode and direction control.
  - Programmable baud-rate generator up to 1/16 system clock
  - Support PDMA mode
- SPI
  - Up to four sets of SPI controller
  - Master up to 32 MHz, and Slave up to 10 MHz (chip working @ 5V)
  - Support SPI master/slave mode
  - Full duplex synchronous serial data transfer
  - Variable length of transfer data from 1 to 32 bits
  - MSB or LSB first data transfer
  - Rx and Tx on both rising or falling edge of serial clock independently
  - 2 slave/device select lines when it is as the master, and 1 slave/device select line when it is as the slave
  - Support byte suspend mode in 32-bit transmission
  - Support PDMA mode
  - Support three wire, no slave select signal, bi-direction interface

- I^{2}C
  - Up to two sets of I^{2}C device
  - 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
  - Support multiple address recognition (four slave address with mask option)
- I^{2}S
  - Interface with external audio CODEC
  - Operate as either master or slave mode
  - Capable of handling 8-, 16-, 24- and 32-bit word sizes
  - Mono and stereo audio data supported
  - I^{2}S and MSB justified data format supported
  - Two 8 word FIFO data buffers are provided, one for transmit and one for receive
  - Generates interrupt requests when buffer levels cross a programmable boundary
  - Support two DMA requests, one for transmit and one for receive
- CAN 2.0
  - Supports CAN protocol version 2.0 part A and B
  - Bit rates up to 1M bit/s
  - 32 Message Objects
  - Each Message Object has its won identifier mask
  - Programmable FIFO mode (concatenation of Message Object)
  - Maskable interrupt
  - Disabled Automatic Re-transmission mode for Time Triggered CAN applications
  - Support power down wake-up function
- PS/2 Device Controller
  - Host communication inhibit and request to send detection
  - Reception frame error detection
  - Programmable 1 to 16 bytes transmit buffer to reduce CPU intervention
  - Double buffer for data reception
  - S/W override bus
- USB 2.0 Full-Speed Device
  - One set of USB 2.0 FS Device 12Mbps
  - On-chip USB Transceiver
  - Provide 1 interrupt source with 4 interrupt events
  - Support Control, Bulk In/Out, Interrupt and Isochronous transfers
  - Auto suspend function when no bus signaling for 3 ms
  - Provide 6 programmable endpoints
  - Include 512 Bytes internal SRAM as USB buffer
  - Provide remote wake-up capability
- EBI (External bus interface) support (100-pin and 64-pin Package Only)
  - Accessible space: 64KB in 8-bit mode or 128KB in 16-bit mode
  - Support 8-/16-bit data width

  - Support byte write in 16-bit data width mode
- ADC
  - 12-bit SAR ADC with 700K SPS
  - Up to 8-ch single-end input or 4-ch differential input
  - Single scan/single cycle scan/continuous scan
  - Each channel with individual result register
  - Scan on enabled channels
  - Threshold voltage detection
  - Conversion start by software programming or external input
  - Support PDMA Mode
- Analog Comparator
  - Up to two analog comparators
  - External input or internal bandgap voltage selectable at negative node
  - Interrupt when compare result change
  - Power down wake-up
- One built-in temperature sensor with 1℃ resolution
- Brown-Out detector
  - With 4 levels: 4.5 V/3.8 V/2.7 V/2.2 V
  - Support Brown-Out Interrupt and Reset option
- Low Voltage Reset
  - Threshold voltage levels: 2.0 V
- Operating Temperature: -40℃~85℃
- Packages:
  - All Green package (RoHS)
  - LQFP 100-pin / 64-pin / 48-pin

3 PARTS INFORMATION LIST AND PIN CONFIGURATION

3.1 NuMicro™ NUC130 Products Selection Guide

3.1.1 NuMicro™ NUC130 Automotive Line Selection Guide

Part number	APROM	RAM	Data Flash	ISP Loader ROM	I/O	Timer	Connectivity						I^2S	Comp.	PWM	ADC	RTC	EBI	ISP ICP	Package
							UART	SPI	I^2C	USB	LIN	CAN								
NUC130LC1CN	32 KB	4 KB	4 KB	4 KB	up to 35	4x32-bit	3	1	2	-	2	1	1	1	4	8x12-bit	v	-	v	LQFP48
NUC130LD2CN	64 KB	8 KB	4 KB	4 KB	up to 35	4x32-bit	3	1	2	-	2	1	1	1	4	8x12-bit	v	-	v	LQFP48
NUC130LE3CN	128 KB	16 KB	Definable	4 KB	up to 35	4x32-bit	3	1	2	-	2	1	1	1	4	8x12-bit	v	-	v	LQFP48
NUC130RC1CN	32 KB	4 KB	4 KB	4 KB	up to 49	4x32-bit	3	2	2	-	2	1	1	2	6	8x12-bit	v	v	v	LQFP64
NUC130RD2CN	64 KB	8 KB	4 KB	4 KB	up to 49	4x32-bit	3	2	2	-	2	1	1	2	6	8x12-bit	v	v	v	LQFP64
NUC130RD3CN	128 KB	16 KB	Definable	4 KB	up to 49	4x32-bit	3	2	2	-	2	1	1	2	6	8x12-bit	v	v	v	LQFP64
NUC130VE3CN	128 KB	16 KB	Definable	4 KB	up to 80	4x32-bit	3	4	2	-	2	1	1	2	8	8x12-bit	v	v	v	LQFP100

3.2 NuMicro™ NUC140 Products Selection Guide

3.2.1 NuMicro™ NUC140 Connectivity Line Selection Guide

Part number	APROM	RAM	Data Flash	ISP Loader ROM	I/O	Timer	Connectivity						I^2S	Comp.	PWM	ADC	RTC	EBI	ISP ICP	Package
							UART	SPI	I^2C	USB	LIN	CAN								
NUC140LC1CN	32 KB	4 KB	4 KB	4 KB	up to 31	4x32-bit	2	1	2	1	2	1	1	1	4	8x12-bit	v	-	v	LQFP48
NUC140LD2CN	64 KB	8 KB	4 KB	4 KB	up to 31	4x32-bit	2	1	2	1	2	1	1	1	4	8x12-bit	v	-	v	LQFP48
NUC140LE3CN	128 KB	16 KB	Definable	4 KB	up to 31	4x32-bit	2	1	2	1	2	1	1	1	4	8x12-bit	v	-	v	LQFP48
NUC140RC1CN	32 KB	4 KB	4 KB	4 KB	up to 45	4x32-bit	3	2	2	1	2	1	1	2	4	8x12-bit	v	v	v	LQFP64
NUC140RD2CN	64 KB	8 KB	4 KB	4 KB	up to 45	4x32-bit	3	2	2	1	2	1	1	2	4	8x12-bit	v	v	v	LQFP64
NUC140RE3CN	128 KB	16 KB	Definable	4 KB	up to 45	4x32-bit	3	2	2	1	2	1	1	2	4	8x12-bit	v	v	v	LQFP64
NUC140VE3CN	128 KB	16 KB	Definable	4 KB	up to 76	4x32-bit	3	4	2	1	2	1	1	2	8	8x12-bit	v	v	v	LQFP100

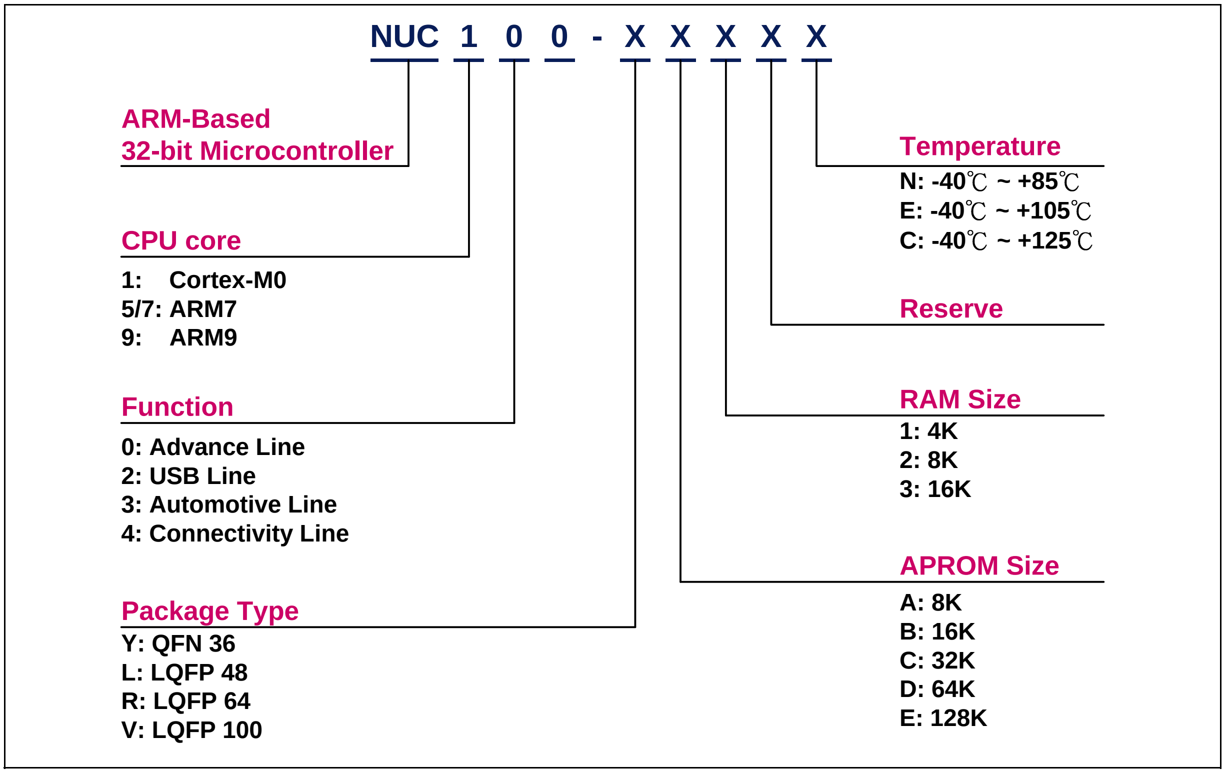


Figure 3-1 NuMicro™ NUC100 Series selection code

3.3 Pin Configuration

3.3.1 NuMicro™ NUC130/NUC140 Pin Diagram

3.3.1.1 NuMicro™ NUC130 LQFP 100 pin

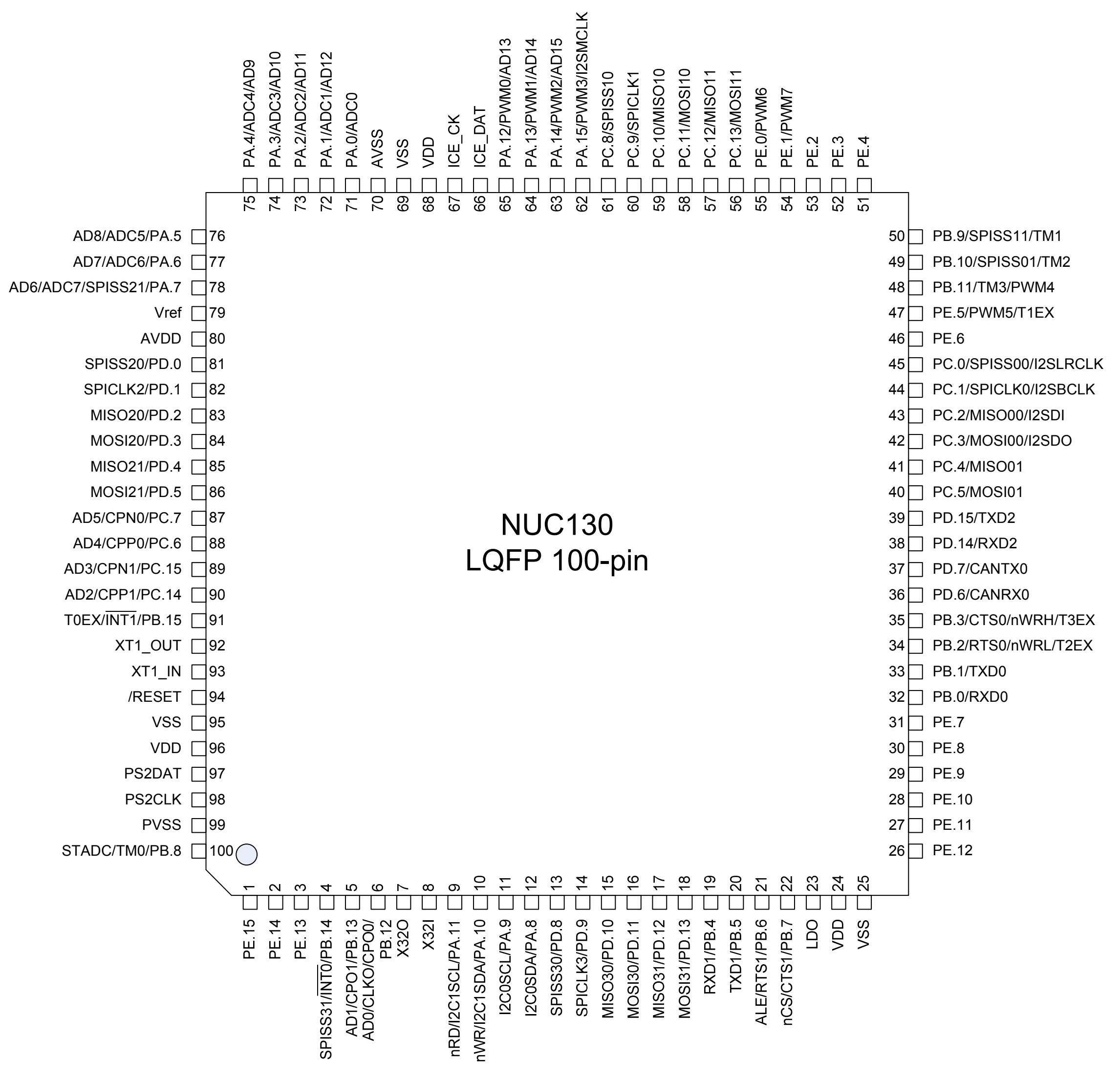


Figure 3-2 NuMicro™ NUC130 LQFP 100-pin Pin Diagram

3.3.1.2 NuMicro™ NUC130 LQFP 64 pin

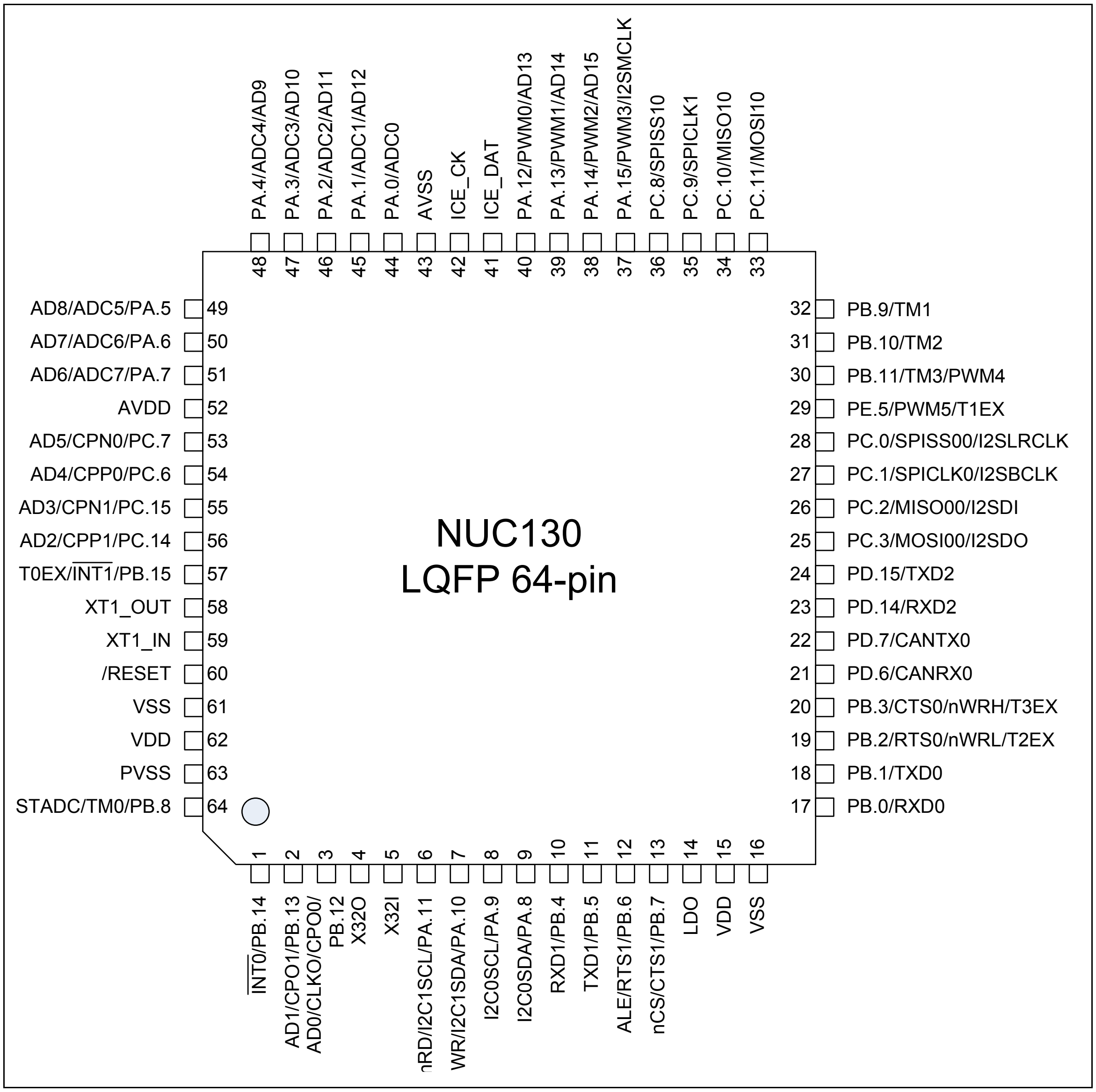


Figure 3-3 NuMicro™ NUC130 LQFP 64-pin Pin Diagram

3.3.1.3 NuMicro™ NUC130 LQFP 48 pin

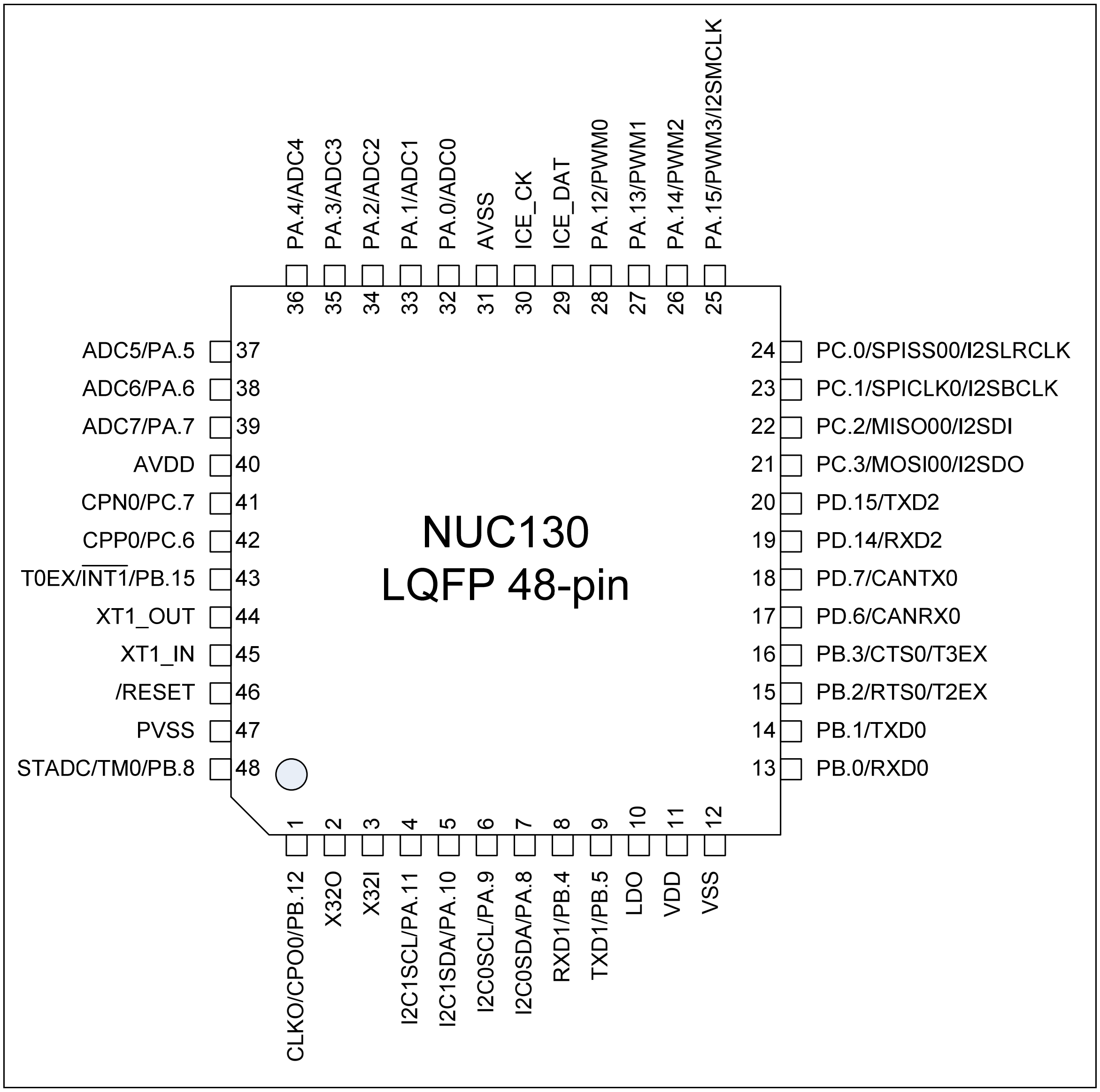


Figure 3-4 NuMicro™ NUC130 LQFP 48-pin Pin Diagram

3.3.1.4 NuMicro™ NUC140 LQFP 100 pin

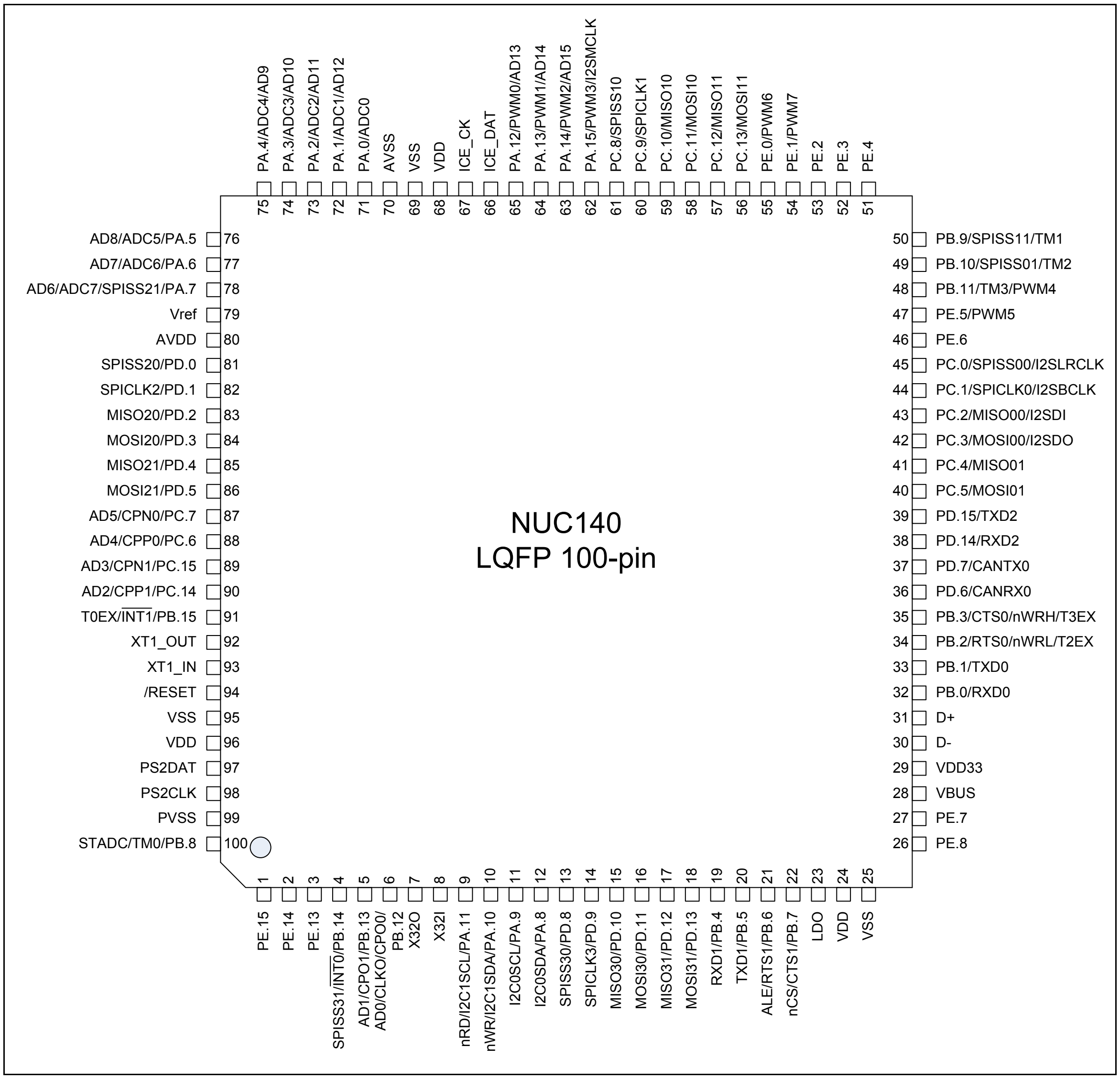


Figure 3-5 NuMicro™ NUC140 LQFP 100-pin Pin Diagram

3.3.1.5 NuMicro™ NUC140 LQFP 64 pin

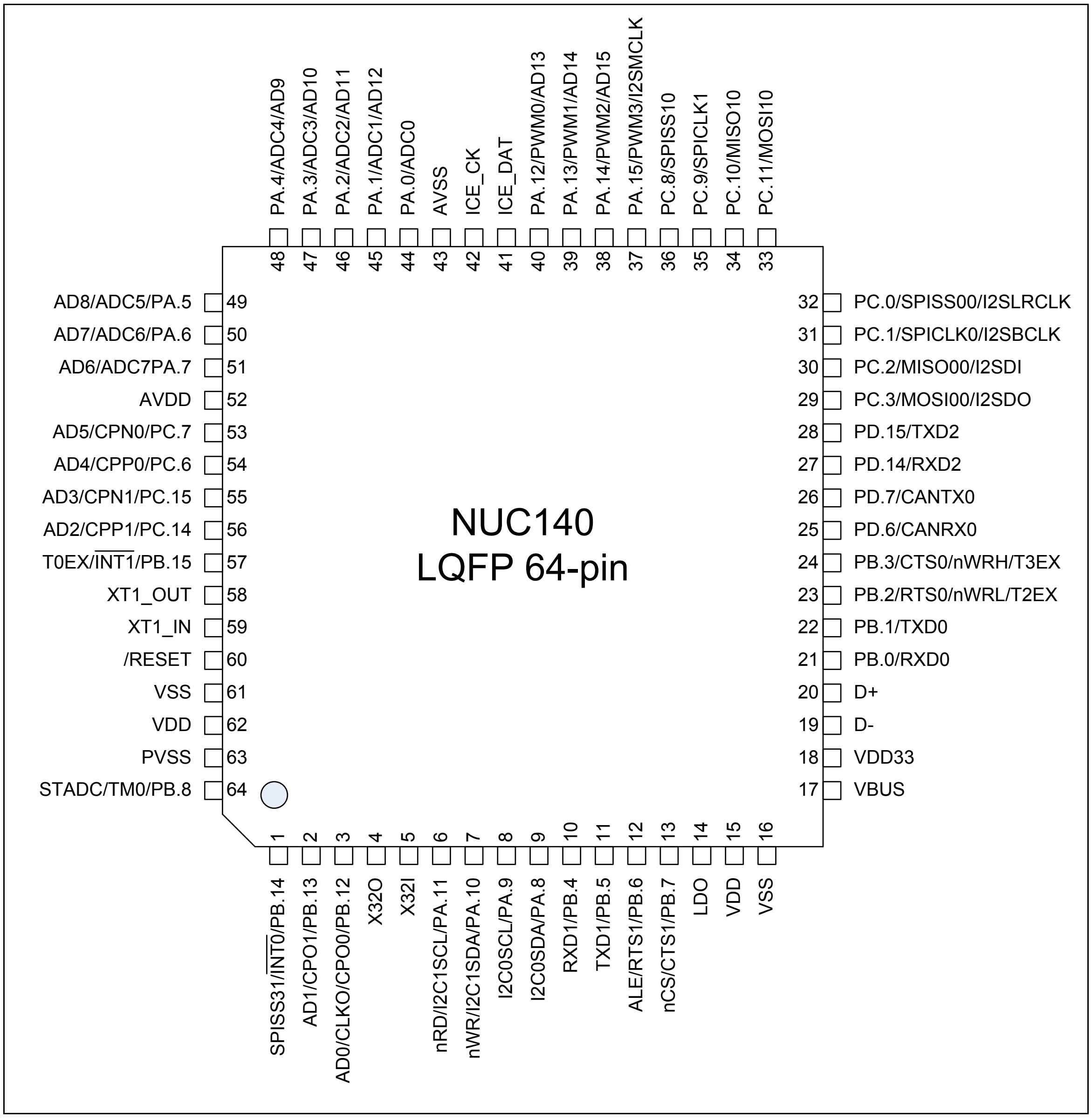


Figure 3-6 NuMicro™ NUC140 LQFP 64-pin Pin Diagram

3.3.1.6 NuMicro™ NUC140 LQFP 48 pin

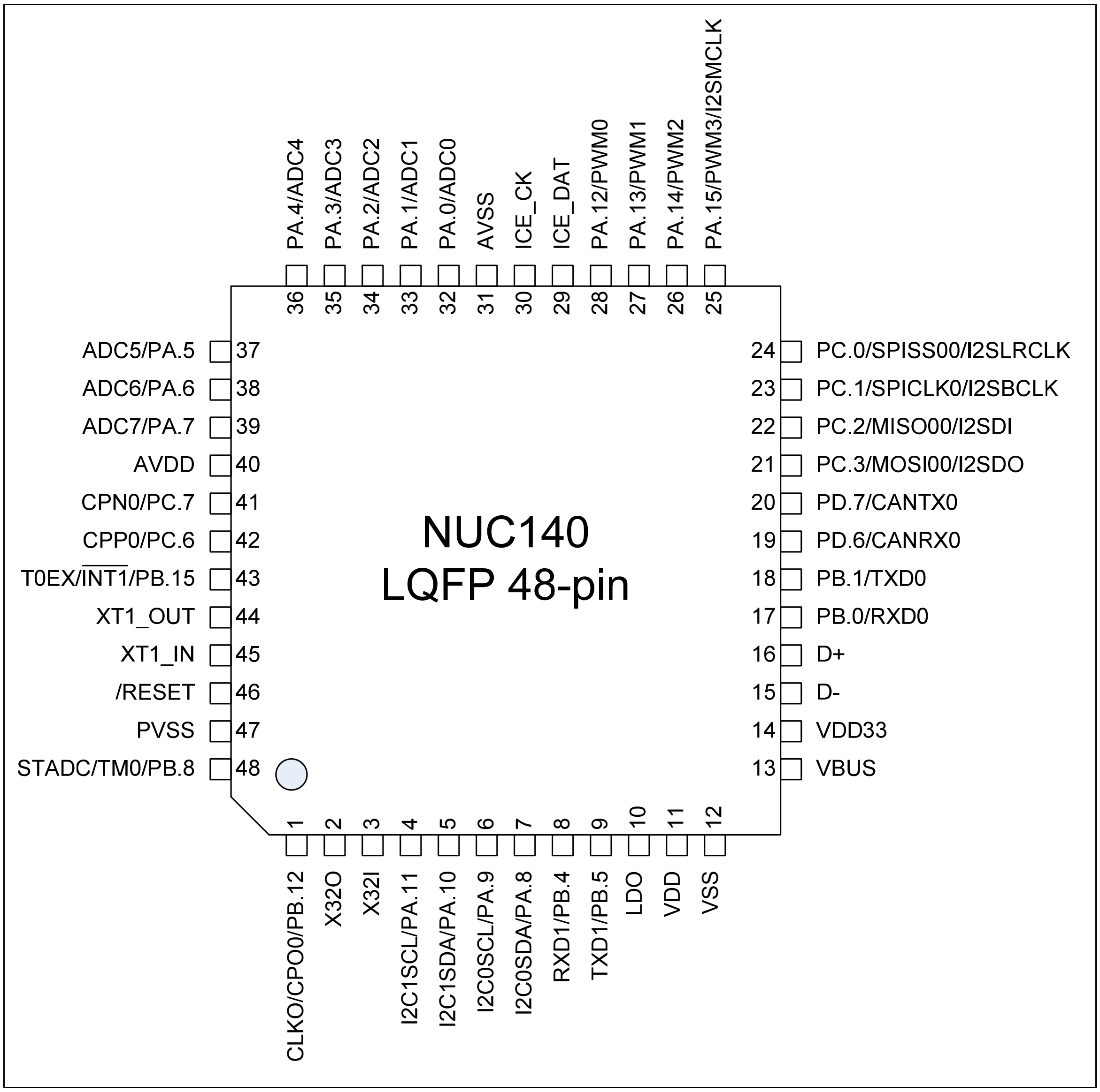


Figure 3-7 NuMicro™ NUC140 LQFP 48-pin Pin Diagram

3.4 Pin Description

3.4.1 NuMicro™ NUC130/NUC140 Pin Description

3.4.1.1 NuMicro™ NUC130 Pin Description

Pin No.			**Pin Name**	**Pin Type**		**Description**
LQFP 100	**LQFP 64**	**LQFP 48**				
1			PE.15	**I/O**		General purpose input/output digital pin
2			PE.14	**I/O**		General purpose input/output digital pin
3			PE.13	**I/O**		General purpose input/output digital pin
4	1		PB.14	**I/O**		General purpose input/output digital pin
			/INT0	**I**		/INT0: External interrupt1 input pin
			SPISS31	**I/O**		SPISS31: SPI3 2^{nd} slave select pin
5	2		PB.13	**I/O**		General purpose input/output digital pin
			CPO1	**O**		Comparator1 output pin
			AD1	**I/O**		EBI Address/Data bus bit1
6	3	1	PB.12	**I/O**		General purpose input/output digital pin
			CPO0	**O**		Comparator0 output pin
			CLKO	**O**		Frequency Divider output pin
			AD0	**I/O**		EBI Address/Data bus bit0
7	4	2	X32O	**O**		External 32.768 kHz low speed crystal output pin
8	5	3	X32I	**I**		External 32.768 kHz low speed crystal input pin
9	6	4	PA.11	**I/O**		General purpose input/output digital pin
			I2C1SCL	**I/O**		I2C1SCL: I^{2}C1 clock pin
			nRD	**O**		EBI read enable output pin
10	7	5	PA.10	**I/O**		General purpose input/output digital pin
			I2C1SDA	**I/O**		I2C1SDA: I^{2}C1 data input/output pin
			nWR	**O**		EBI write enable output pin
11	8	6	PA.9	**I/O**		General purpose input/output digital pin
			I2C0SCL	**I/O**		I2C0SCL: I^{2}C0 clock pin
12	9	7	PA.8	**I/O**		General purpose input/output digital pin
			I2C0SDA	**I/O**		I2C0SDA: I^{2}C0 data input/output pin

Pin No.			Pin Name	Pin Type		Description
LQFP 100	LQFP 64	LQFP 48				
13			PD.8	**I/O**		General purpose input/output digital pin
			SPISS30	**I/O**		SPISS30: SPI3 slave select pin
14			PD.9	**I/O**		General purpose input/output digital pin
			SPICLK3	**I/O**		SPICLK3: SPI3 serial clock pin
15			PD.10	**I/O**		General purpose input/output digital pin
			MISO30	**I/O**		MISO30: SPI3 MISO (Master In, Slave Out) pin
16			PD.11	**I/O**		General purpose input/output digital pin
			MOSI30	**I/O**		MOSI30: SPI3 MOSI (Master Out, Slave In) pin
17			PD.12	**I/O**		General purpose input/output digital pin
			MISO31	**I/O**		MISO31: SPI3 2nd MISO (Master In, Slave Out) pin
18			PD.13	**I/O**		General purpose input/output digital pin
			MOSI31	**I/O**		MOSI31: SPI3 2nd MOSI (Master Out, Slave In) pin
19	10	8	PB.4	**I/O**		General purpose input/output digital pin
			RXD1	**I**		RXD1: Data receiver input pin for UART1
20	11	9	PB.5	**I/O**		General purpose input/output digital pin
			TXD1	**O**		TXD1: Data transmitter output pin for UART1
21	12		PB.6	**I/O**		General purpose input/output digital pin
			RTS1	**O**		RTS1: Request to Send output pin for UART1
			ALE	**O**		EBI address latch enable output pin
22	13		PB.7	**I/O**		General purpose input/output digital pin
			CTS1	**I**		CTS1: Clear to Send input pin for UART1
			nCS	**O**		EBI chip select enable output pin
23	14	10	LDO	**P**		LDO output pin
24	15	11	VDD	**P**		Power supply for I/O ports and LDO source for internal PLL and digital function
25	16	12	VSS	**P**		Ground
26			PE.12	**I/O**		General purpose input/output digital pin
27			PE.11	**I/O**		General purpose input/output digital pin
28			PE.10	**I/O**		General purpose input/output digital pin
29			PE.9	**I/O**		General purpose input/output digital pin

Pin No.			Pin Name	Pin Type		Description
LQFP 100	LQFP 64	LQFP 48				
30			PE.8	I/O		General purpose input/output digital pin
31			PE.7	I/O		General purpose input/output digital pin
32	17	13	PB.0	I/O		General purpose input/output digital pin
			RXD0	I		RXD0: Data receiver input pin for UART0
33	18	14	PB.1	I/O		General purpose input/output digital pin
			TXD0	O		TXD0: Data transmitter output pin for UART0
34	19	15	PB.2	I/O		General purpose input/output digital pin
			RTS0	O		RTS0: Request to Send output pin for UART0
			nWRL	O		EBI low byte write enable output pin
			T2EX	I		Timer2 external capture input pin
35	20	16	PB.3	I/O		General purpose input/output digital pin
			CTS0	I		CTS0: Clear to Send input pin for UART0
			nWRH	O		EBI high byte write enable output pin
			T3EX	I		Timer3 external capture input pin
36	21	17	PD.6	I/O		General purpose input/output digital pin
			CANRX0	I		CAN Bus0 RX Input
37	22	18	PD.7	I/O		General purpose input/output digital pin
			CANTX0	O		CAN Bus0 TX Output
38	23	19	PD.14	I/O		General purpose input/output digital pin
			RXD2	I		RXD2: Data receiver input pin for UART2
39	24	20	PD.15	I/O		General purpose input/output digital pin
			TXD2	O		TXD2: Data transmitter output pin for UART2
40			PC.5	I/O		General purpose input/output digital pin
			MOSI01	I/O		MOSI01: SPI0 2nd MOSI (Master Out, Slave In) pin
41			PC.4	I/O		General purpose input/output digital pin
			MISO01	I/O		MISO01: SPI0 2nd MISO (Master In, Slave Out) pin
42	25	21	PC.3	I/O		General purpose input/output digital pin
			MOSI00	I/O		MOSI00: SPI0 MOSI (Master Out, Slave In) pin
			I2SDO	O		I2SDO: I^{2}S data output

Pin No.			Pin Name	Pin Type		Description
LQFP 100	LQFP 64	LQFP 48				
43	26	22	PC.2	**I/O**		General purpose input/output digital pin
			MISO00	**I/O**		MISO00: SPI0 MISO (Master In, Slave Out) pin
			I2SDI	**I**		I2SDI: I^2S data input
44	27	23	PC.1	**I/O**		General purpose input/output digital pin
			SPICLK0	**I/O**		SPICLK0: SPI0 serial clock pin
			I2SBCLK	**I/O**		I2SBCLK: I^2S bit clock pin
45	28	24	PC.0	**I/O**		General purpose input/output digital pin
			SPISS00	**I/O**		SPISS00: SPI0 slave select pin
			I2SLRCLK	**I/O**		I2SLRCLK: I^2S left right channel clock
46			PE.6	**I/O**		General purpose input/output digital pin
47	29		PE.5	**I/O**		General purpose input/output digital pin
			PWM5	**I/O**		PWM5: PWM output/Capture input
			T1EX	**I**		Timer1 external capture input
48	30		PB.11	**I/O**		General purpose input/output digital pin
			TM3	**I/O**		TM3: Timer3 event counter input / toggle output
			PWM4	**I/O**		PWM4: PWM output/Capture input
49	31		PB.10	**I/O**		General purpose input/output digital pin
			TM2	**I/O**		TM2: Timer2 event counter input / toggle output
			SPISS01	**I/O**		SPISS01: SPI0 2nd slave select pin
50	32		PB.9	**I/O**		General purpose input/output digital pin
			TM1	**I/O**		TM1: Timer1 event counter input / toggle output
			SPISS11	**I/O**		SPISS11: SPI1 2nd slave select pin
51			PE.4	**I/O**		General purpose input/output digital pin
52			PE.3	**I/O**		General purpose input/output digital pin
53			PE.2	**I/O**		General purpose input/output digital pin
54			PE.1	**I/O**		General purpose input/output digital pin
			PWM7	**I/O**		PWM7: PWM output/Capture input
55			PE.0	**I/O**		General purpose input/output digital pin
			PWM6	**I/O**		PWM6: PWM output/Capture input

Pin No.			Pin Name	Pin Type		Description
LQFP 100	LQFP 64	LQFP 48				
56			PC.13	**I/O**		General purpose input/output digital pin
			MOSI11	**I/O**		MOSI11: SPI1 2nd MOSI (Master Out, Slave In) pin
57			PC.12	**I/O**		General purpose input/output digital pin
			MISO11	**I/O**		MISO11: SPI1 2nd MISO (Master In, Slave Out) pin
58	33		PC.11	**I/O**		General purpose input/output digital pin
			MOSI10	**I/O**		MOSI10: SPI1 MOSI (Master Out, Slave In) pin
59	34		PC.10	**I/O**		General purpose input/output digital pin
			MISO10	**I/O**		MISO10: SPI1 MISO (Master In, Slave Out) pin
60	35		PC.9	**I/O**		General purpose input/output digital pin
			SPICLK1	**I/O**		SPICLK1: SPI1 serial clock pin
61	36		PC.8	**I/O**		General purpose input/output digital pin
			SPISS10	**I/O**		SPISS10: SPI1 slave select pin
			MCLK	**O**		EBI clock output
62	37	25	PA.15	**I/O**		General purpose input/output digital pin
			PWM3	**I/O**		PWM3: PWM output/Capture input
			I2SMCLK	**O**		I2SMCLK: I^{2}S master clock output pin
63	38	26	PA.14	**I/O**		General purpose input/output digital pin
			PWM2	**I/O**		PWM2: PWM output/Capture input
			AD15	**I/O**		EBI Address/Data bus bit15
64	39	27	PA.13	**I/O**		General purpose input/output digital pin
			PWM1	**I/O**		PWM1: PWM output/Capture input
			AD14	**I/O**		EBI Address/Data bus bit14
65	40	28	PA.12	**I/O**		General purpose input/output digital pin
			PWM0	**I/O**		PWM0: PWM output/Capture input
			AD13	**I/O**		EBI Address/Data bus bit13
66	41	29	ICE_DAT	**I/O**		Serial Wired Debugger Data pin
67	42	30	ICE_CK	**I**		Serial Wired Debugger Clock pin
68			VDD	**P**		Power supply for I/O ports and LDO source for internal PLL and digital circuit
69			VSS	**P**		Ground

Pin No.			Pin Name	Pin Type		Description
LQFP 100	LQFP 64	LQFP 48				
70	43	31	AVSS	**AP**		Ground Pin for analog circuit
71	44	32	PA.0	**I/O**		General purpose input/output digital pin
			ADC0	**AI**		ADC0: ADC analog input
72	45	33	PA.1	**I/O**		General purpose input/output digital pin
			ADC1	**AI**		ADC1: ADC analog input
			AD12	**I/O**		EBI Address/Data bus bit12
73	46	34	PA.2	**I/O**		General purpose input/output digital pin
			ADC2	**AI**		ADC2: ADC analog input
			AD11	**I/O**		EBI Address/Data bus bit11
74	47	35	PA.3	**I/O**		General purpose input/output digital pin
			ADC3	**AI**		ADC3: ADC analog input
			AD10	**I/O**		EBI Address/Data bus bit10
75	48	36	PA.4	**I/O**		General purpose input/output digital pin
			ADC4	**AI**		ADC4: ADC analog input
			AD9	**I/O**		EBI Address/Data bus bit9
76	49	37	PA.5	**I/O**		General purpose input/output digital pin
			ADC5	**AI**		ADC5: ADC analog input
			AD8	**I/O**		EBI Address/Data bus bit8
77	50	38	PA.6	**I/O**		General purpose input/output digital pin
			ADC6	**AI**		ADC6: ADC analog input
			AD7	**I/O**		EBI Address/Data bus bit7
78	51	39	PA.7	**I/O**		General purpose input/output digital pin
			ADC7	**AI**		ADC7: ADC analog input
			SPISS21	**I/O**		SPISS21: SPI2 2nd slave select pin
			AD6	**I/O**		EBI Address/Data bus bit6
79			VREF	**AP**		Voltage reference input for ADC
80	52	40	AVDD	**AP**		Power supply for internal analog circuit
81			PD.0	**I/O**		General purpose input/output digital pin
			SPISS20	**I/O**		SPISS20: SPI2 slave select pin
82			PD.1	**I/O**		General purpose input/output digital pin

Pin No.			Pin Name	Pin Type		Description
LQFP 100	**LQFP 64**	**LQFP 48**				
			SPICLK2	**I/O**		SPICLK2: SPI2 serial clock pin
83			PD.2	**I/O**		General purpose input/output digital pin
			MISO20	**I/O**		MISO20: SPI2 MISO (Master In, Slave Out) pin
84			PD.3	**I/O**		General purpose input/output digital pin
			MOSI20	**I/O**		MOSI20: SPI2 MOSI (Master Out, Slave In) pin
85			PD.4	**I/O**		General purpose input/output digital pin
			MISO21	**I/O**		MISO21: SPI2 2nd MISO (Master In, Slave Out) pin
86			PD.5	**I/O**		General purpose input/output digital pin
			MOSI21	**I/O**		MOSI21: SPI2 2nd MOSI (Master Out, Slave In) pin
87	53	41	PC.7	**I/O**		General purpose input/output digital pin
			CPN0	**AI**		CPN0: Comparator0 Negative input pin
			AD5	**I/O**		EBI Address/Data bus bit5
88	54	42	PC.6	**I/O**		General purpose input/output digital pin
			CPP0	**AI**		CPP0: Comparator0 Positive input pin
			AD4	**I/O**		EBI Address/Data bus bit4
89	55		PC.15	**I/O**		General purpose input/output digital pin
			CPN1	**AI**		CPN1: Comparator1 Negative input pin
			AD3	**I/O**		EBI Address/Data bus bit3
90	56		PC.14	**I/O**		General purpose input/output digital pin
			CPP1	**AI**		CPP1: Comparator1 Positive input pin
			AD2	**I/O**		EBI Address/Data bus bit2
91	57	43	PB.15	**I/O**		General purpose input/output digital pin
			/INT1	**I**		/INT1: External interrupt0 input pin
			T0EX	**I**		Timer0 external capture input
92	58	44	XT1_OUT	**O**		External 4~24 MHz high speed crystal output pin
93	59	45	XT1_IN	**I**		External 4~24 MHz high speed crystal input pin
94	60	46	/RESET	**I**		External reset input: Low active, set this pin low reset chip to initial state. With internal pull-up.
95	61		VSS	**P**		Ground
96	62		VDD	**P**		Power supply for I/O ports and LDO source for internal PLL and digital circuit

Pin No.			**Pin Name**	**Pin Type**		**Description**
LQFP 100	**LQFP 64**	**LQFP 48**				
97			PS2DAT	**I/O**		PS/2 Data pin
98			PS2CLK	**I/O**		PS/2 clock pin
99	63	47	PVSS	**P**		PLL Ground
100	64	48	PB.8	**I/O**		General purpose input/output digital pin
			STADC	**I**		STADC: ADC external trigger input.
			TM0	**I/O**		TM0: Timer0 event counter input / toggle output

Note: Pin Type I=Digital Input, O=Digital Output; AI=Analog Input; P=Power Pin; AP=Analog Power

3.4.1.2 NuMicro™ NUC140 Pin Description

Pin No.			**Pin Name**	**Pin Type**		**Description**
LQFP 100	**LQFP 64**	**LQFP 48**				
1			PE.15	**I/O**		General purpose input/output digital pin
2			PE.14	**I/O**		General purpose input/output digital pin
3			PE.13	**I/O**		General purpose input/output digital pin
4	1		PB.14	**I/O**		General purpose input/output digital pin
			/INT0	**I**		/INT0: External interrupt1 input pin
			SPISS31	**I/O**		SPISS31: SPI3 2^{nd} slave select pin
5	2		PB.13	**I/O**		General purpose input/output digital pin
			CPO1	**O**		Comparator1 output pin
			AD1	**IO**		EBI Address/Data bus bit1
6	3	1	PB.12	**I/O**		General purpose input/output digital pin
			CPO0	**O**		Comparator0 output pin
			CLKO	**O**		Frequency Divider output pin
			AD0	**I/O**		EBI Address/Data bus bit0
7	4	2	X32O	**O**		External 32.768 kHz low speed crystal output pin
8	5	3	X32I	**I**		External 32.768 kHz low speed crystal input pin
9	6	4	PA.11	**I/O**		General purpose input/output digital pin
			I2C1SCL	**I/O**		I2C1SCL: I^{2}C1 clock pin
			nRD	**O**		EBI read enable output pin
10	7	5	PA.10	**I/O**		General purpose input/output digital pin
			I2C1SDA	**I/O**		I2C1SDA: I^{2}C1 data input/output pin
			nWR	**O**		EBI write enable output pin
11	8	6	PA.9	**I/O**		General purpose input/output digital pin
			I2C0SCL	**I/O**		I2C0SCL: I^{2}C0 clock pin
12	9	7	PA.8	**I/O**		General purpose input/output digital pin
			I2C0SDA	**I/O**		I2C0SDA: I^{2}C0 data input/output pin
13			PD.8	**I/O**		General purpose input/output digital pin
			SPISS30	**I/O**		SPISS30: SPI3 slave select pin
14			PD.9	**I/O**		General purpose input/output digital pin

Pin No.			Pin Name	Pin Type		Description
LQFP 100	LQFP 64	LQFP 48				
			SPICLK3	**I/O**		SPICLK3: SPI3 serial clock pin
15			PD.10	**I/O**		General purpose input/output digital pin
			MISO30	**I/O**		MISO30: SPI3 MISO (Master In, Slave Out) pin
16			PD.11	**I/O**		General purpose input/output digital pin
			MOSI30	**I/O**		MOSI30: SPI3 MOSI (Master Out, Slave In) pin
17			PD.12	**I/O**		General purpose input/output digital pin
			MISO31	**I/O**		MISO31: SPI3 2nd MISO (Master In, Slave Out) pin
18			PD.13	**I/O**		General purpose input/output digital pin
			MOSI31	**I/O**		MOSI31: SPI3 2nd MOSI (Master Out, Slave In) pin
19	10	8	PB.4	**I/O**		General purpose input/output digital pin
			RXD1	**I**		RXD1: Data receiver input pin for UART1
20	11	9	PB.5	**I/O**		General purpose input/output digital pin
			TXD1	**O**		TXD1: Data transmitter output pin for UART1
21	12		PB.6	**I/O**		General purpose input/output digital pin
			RTS1	**O**		RTS1: Request to Send output pin for UART1
			ALE	**O**		EBI address latch enable output pin
22	13		PB.7	**I/O**		General purpose input/output digital pin
			CTS1	**I**		CTS1: Clear to Send input pin for UART1
			nCS	**O**		EBI chip select enable output pin
23	14	10	LDO	**P**		LDO output pin
24	15	11	VDD	**P**		Power supply for I/O ports and LDO source for internal PLL and digital function
25	16	12	VSS	**P**		Ground
26			PE.8	**I/O**		General purpose input/output digital pin
27			PE.7	**I/O**		General purpose input/output digital pin
28	17	13	VBUS	**USB**		POWER SUPPLY: From USB Host or HUB.
29	18	14	VDD33	**USB**		Internal Power Regulator Output 3.3 V Decoupling Pin
30	19	15	D-	**USB**		USB Differential Signal D-
31	20	16	D+	**USB**		USB Differential Signal D+
32	21	17	PB.0	**I/O**		General purpose input/output digital pin

Pin No.			Pin Name	Pin Type		Description
LQFP 100	LQFP 64	LQFP 48				
			RXD0	I		RXD0: Data receiver input pin for UART0
33	22	18	PB.1	I/O		General purpose input/output digital pin
			TXD0	O		TXD0: Data transmitter output pin for UART0
34	23		PB.2	I/O		General purpose input/output digital pin
			RTS0	O		RTS0: Request to Send output pin for UART0
			nWRL	O		EBI low byte write enable output pin
			T2EX	I		Timer2 external capture input pin
35	24		PB.3	I/O		General purpose input/output digital pin
			CTS0	I		CTS0: Clear to Send input pin for UART0
			nWRH	O		EBI high byte write enable output pin
			T3EX	I		Timer3 external capture input pin
36	25	19	PD.6	I/O		General purpose input/output digital pin
			CANRX0	I		CAN Bus0 RX Input
37	26	20	PD.7	I/O		General purpose input/output digital pin
			CANTX0	O		CAN Bus0 TX Output
38	27		PD.14	I/O		General purpose input/output digital pin
			RXD2	I		RXD2: Data receiver input pin for UART2
39	28		PD.15	I/O		General purpose input/output digital pin
			TXD2	O		TXD2: Data transmitter output pin for UART2
40			PC.5	I/O		General purpose input/output digital pin
			MOSI01	I/O		MOSI01: SPI0 2nd MOSI (Master Out, Slave In) pin
41			PC.4	I/O		General purpose input/output digital pin
			MISO01	I/O		MISO01: SPI0 2nd MISO (Master In, Slave Out) pin
42	29	21	PC.3	I/O		General purpose input/output digital pin
			MOSI00	I/O		MOSI00: SPI0 MOSI (Master Out, Slave In) pin
			I2SDO	O		I2SDO: I^{2}S data output
43	30	22	PC.2	I/O		General purpose input/output digital pin
			MISO00	I/O		MISO00: SPI0 MISO (Master In, Slave Out) pin
			I2SDI	I		I2SDI: I^{2}S data input

Pin No.			Pin Name	Pin Type		Description
LQFP 100	LQFP 64	LQFP 48				
44	31	23	PC.1	**I/O**		General purpose input/output digital pin
			SPICLK0	**I/O**		SPICLK0: SPI0 serial clock pin
			I2SBCLK	**I/O**		I2SBCLK: I^{2}S bit clock pin
45	32	24	PC.0	**I/O**		General purpose input/output digital pin
			SPISS00	**I/O**		SPISS00: SPI0 slave select pin
			I2SLRCLK	**I/O**		I2SLRCLK: I^{2}S left right channel clock
46			PE.6	**I/O**		General purpose input/output digital pin
47			PE.5	**I/O**		General purpose input/output digital pin
			PWM5	**I/O**		PWM5: PWM output/Capture input
			T1EX	**I**		Timer1 external capture input pin
48			PB.11	**I/O**		General purpose input/output digital pin
			TM3	**I/O**		TM3: Timer3 event counter input / toggle output
			PWM4	**I/O**		PWM4: PWM output/Capture input
49			PB.10	**I/O**		General purpose input/output digital pin
			TM2	**I/O**		TM2: Timer2 event counter input / toggle output
			SPISS01	**I/O**		SPISS01: SPI0 2nd slave select pin
50			PB.9	**I/O**		General purpose input/output digital pin
			TM1	**I/O**		TM1: Timer1 event counter input / toggle output
			SPISS11	**I/O**		SPISS11: SPI1 2nd slave select pin
51			PE.4	**I/O**		General purpose input/output digital pin
52			PE.3	**I/O**		General purpose input/output digital pin
53			PE.2	**I/O**		General purpose input/output digital pin
54			PE.1	**I/O**		General purpose input/output digital pin
			PWM7	**I/O**		PWM7: PWM output/Capture input
55			PE.0	**I/O**		General purpose input/output digital pin
			PWM6	**I/O**		PWM6: PWM output/Capture input
56			PC.13	**I/O**		General purpose input/output digital pin
			MOSI11	**I/O**		MOSI11: SPI1 2nd MOSI (Master Out, Slave In) pin
57			PC.12	**I/O**		General purpose input/output digital pin

Pin No.			Pin Name	Pin Type		Description
LQFP 100	LQFP 64	LQFP 48				
			MISO11	**I/O**		MISO11: SPI1 2nd MISO (Master In, Slave Out) pin
58	33		PC.11	**I/O**		General purpose input/output digital pin
			MOSI10	**I/O**		MOSI10: SPI1 MOSI (Master Out, Slave In) pin
59	34		PC.10	**I/O**		General purpose input/output digital pin
			MISO10	**I/O**		MISO10: SPI1 MISO (Master In, Slave Out) pin
60	35		PC.9	**I/O**		General purpose input/output digital pin
			SPICLK1	**I/O**		SPICLK1: SPI1 serial clock pin
61	36		PC.8	**I/O**		General purpose input/output digital pin
			SPISS10	**I/O**		SPISS10: SPI1 slave select pin
			MCLK	**O**		EBI clock output
62	37	25	PA.15	**I/O**		General purpose input/output digital pin
			PWM3	**I/O**		PWM3: PWM output/Capture input
			I2SMCLK	**O**		I2SMCLK: I^{2}S master clock output pin
63	38	26	PA.14	**I/O**		General purpose input/output digital pin
			PWM2	**I/O**		PWM2: PWM output/Capture input
			AD15	**I/O**		EBI Address/Data bus bit15
64	39	27	PA.13	**I/O**		General purpose input/output digital pin
			PWM1	**I/O**		PWM1: PWM output/Capture input
			AD14	**I/O**		EBI Address/Data bus bit14
65	40	28	PA.12	**I/O**		General purpose input/output digital pin
			PWM0	**I/O**		PWM0: PWM output/Capture input
			AD13	**I/O**		EBI Address/Data bus bit13
66	41	29	ICE_DAT	**I/O**		Serial Wired Debugger Data pin
67	42	30	ICE_CK	**I**		Serial Wired Debugger Clock pin
68			VDD	**P**		Power supply for I/O ports and LDO source for internal PLL and digital circuit
69			VSS	**P**		Ground
70	43	31	AVSS	**AP**		Ground Pin for analog circuit
71	44	32	PA.0	**I/O**		General purpose input/output digital pin
			ADC0	**AI**		ADC0: ADC analog input

Pin No.			Pin Name	Pin Type		Description
LQFP 100	LQFP 64	LQFP 48				
72	45	33	PA.1	**I/O**		General purpose input/output digital pin
			ADC1	**AI**		ADC1: ADC analog input
			AD12	**I/O**		EBI Address/Data bus bit12
73	46	34	PA.2	**I/O**		General purpose input/output digital pin
			ADC2	**AI**		ADC2: ADC analog input
			AD11	**I/O**		EBI Address/Data bus bit11
74	47	35	PA.3	**I/O**		General purpose input/output digital pin
			ADC3	**AI**		ADC3: ADC analog input
			AD10	**I/O**		EBI Address/Data bus bit10
75	48	36	PA.4	**I/O**		General purpose input/output digital pin
			ADC4	**AI**		ADC4: ADC analog input
			AD9	**I/O**		EBI Address/Data bus bit9
76	49	37	PA.5	**I/O**		General purpose input/output digital pin
			ADC5	**AI**		ADC5: ADC analog input
			AD8	**I/O**		EBI Address/Data bus bit8
77	50	38	PA.6	**I/O**		General purpose input/output digital pin
			ADC6	**AI**		ADC6: ADC analog input
			AD7	**I/O**		EBI Address/Data bus bit7
78	51	39	PA.7	**I/O**		General purpose input/output digital pin
			ADC7	**AI**		ADC7: ADC analog input
			SPISS21	**I/O**		SPISS21: SPI2 2nd slave select pin
			AD6	**I/O**		EBI Address/Data bus bit6
79			VREF	**AP**		Voltage reference input for ADC
80	52	40	AVDD	**AP**		Power supply for internal analog circuit
81			PD.0	**I/O**		General purpose input/output digital pin
			SPISS20	**I/O**		SPISS20: SPI2 slave select pin
82			PD.1	**I/O**		General purpose input/output digital pin
			SPICLK2	**I/O**		SPICLK2: SPI2 serial clock pin
83			PD.2	**I/O**		General purpose input/output digital pin
			MISO20	**I/O**		MISO20: SPI2 MISO (Master In, Slave Out) pin

Pin No.			Pin Name	Pin Type		Description
LQFP 100	LQFP 64	LQFP 48				
84			PD.3	**I/O**		General purpose input/output digital pin
			MOSI20	**I/O**		MOSI20: SPI2 MOSI (Master Out, Slave In) pin
85			PD.4	**I/O**		General purpose input/output digital pin
			MISO21	**I/O**		MISO21: SPI2 2nd MISO (Master In, Slave Out) pin
86			PD.5	**I/O**		General purpose input/output digital pin
			MOSI21	**I/O**		MOSI21: SPI2 2nd MOSI (Master Out, Slave In) pin
87	53	41	PC.7	**I/O**		General purpose input/output digital pin
			CPN0	**AI**		CPN0: Comparator0 Negative input pin
			AD5	**I/O**		EBI Address/Data bus bit 5
88	54	42	PC.6	**I/O**		General purpose input/output digital pin
			CPP0	**AI**		CPP0: Comparator0 Positive input pin
			AD4	**I/O**		EBI Address/Data bus bit 4
89	55		PC.15	**I/O**		General purpose input/output digital pin
			CPN1	**AI**		CPN1: Comparator1 Negative input pin
			AD3	**I/O**		EBI Address/Data bus bit 3
90	56		PC.14	**I/O**		General purpose input/output digital pin
			CPP1	**AI**		CPP1: Comparator1 Positive input pin
			AD2	**I/O**		EBI Address/Data bus bit 2
91	57	43	PB.15	**I/O**		General purpose input/output digital pin
			/INT1	**I**		/INT1: External interrupt0 input pin
			T0EX	**I**		Timer 0 external capture input pin
92	58	44	XT1_OUT	**O**		External 4~24 MHz high speed crystal output pin
93	59	45	XT1_IN	**I**		External 4~24 MHz high speed crystal input pin
94	60	46	/RESET	**I**		External reset input: Low active, set this pin low reset chip to initial state. With internal pull-up.
95	61		VSS	**P**		Ground
96	62		VDD	**P**		Power supply for I/O ports and LDO source for internal PLL and digital circuit
97			PS2DAT	**I/O**		PS/2 Data pin
98			PS2CLK	**I/O**		PS/2 clock pin
99	63	47	PVSS	**P**		PLL Ground

Pin No.			**Pin Name**	**Pin Type**		**Description**
LQFP 100	**LQFP 64**	**LQFP 48**				
100	64	48	PB.8	**I/O**		General purpose input/output digital pin
			STADC	**I**		STADC: ADC external trigger input.
			TM0	**I/O**		TM0: Timer0 event counter input / toggle output

Note: Pin Type I=Digital Input, O=Digital Output; AI=Analog Input; P=Power Pin; AP=Analog Power

4 BLOCK DIAGRAM

4.1 NuMicro™NUC130/NUC140 Block Diagram

4.1.1 NuMicro™ NUC130 Block Diagram

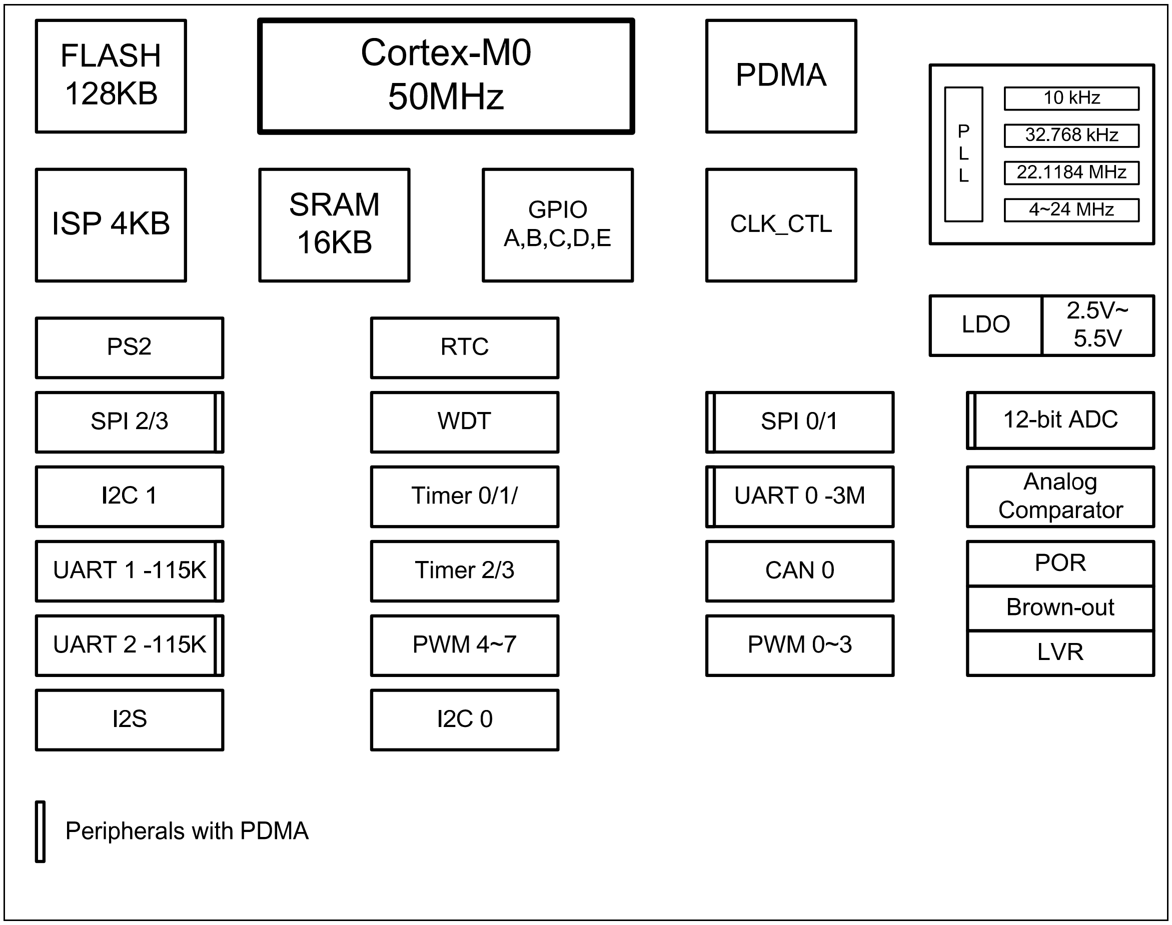


Figure 4-1 NuMicro™ NUC130 Block Diagram

4.1.2 NuMicro™ NUC140 Block Diagram

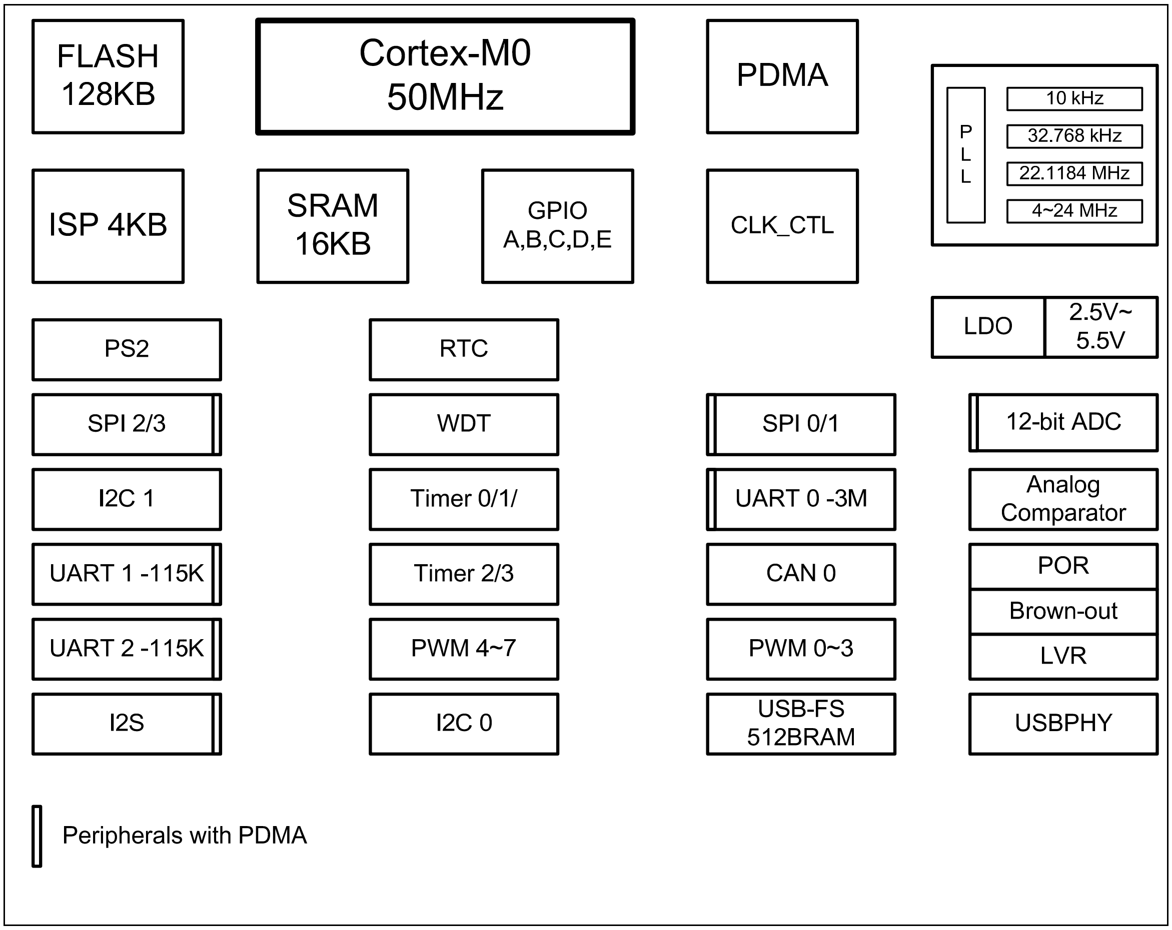


Figure 4-2 NuMicro™ NUC140 Block Diagram

5 FUNCTIONAL DESCRIPTION

5.1 ARM® Cortex™-M0 Core

The Cortex™-M0 processor is a configurable, multistage, 32-bit RISC processor. It has an AMBA AHB-Lite interface and includes an NVIC component. It also has optional hardware debug functionality. The processor can execute Thumb code and is compatible with other Cortex-M profile processor. 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. Figure 5-1 shows the functional controller of processor.

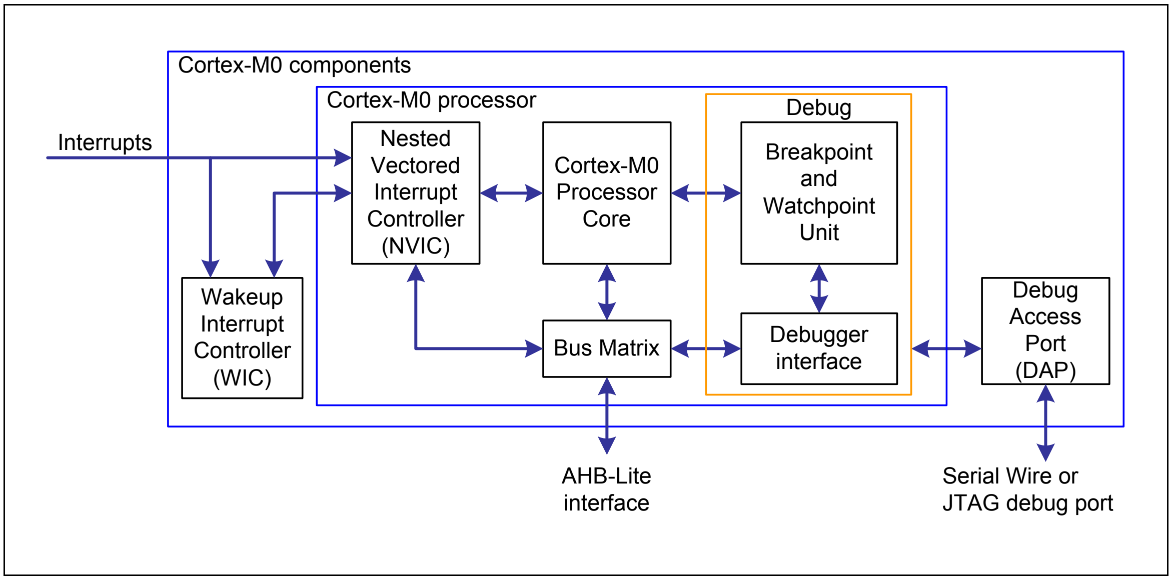


Figure 5-1 Functional Controller Diagram

The implemented device provides:

- A low gate count processor that features:
  - ◆ The ARMv6-M Thumb® instruction set
  - ◆ Thumb-2 technology
  - ◆ ARMv6-M compliant 24-bit SysTick timer
  - ◆ A 32-bit hardware multiplier
  - ◆ The system interface supports little-endian data accesses
  - ◆ The ability to have deterministic, fixed-latency, interrupt handling
  - ◆ Load/store-multiples and multicycle-multiplies that can be abandoned and restarted to facilitate rapid interrupt handling
  - ◆ C Application Binary Interface compliant exception model. This is the ARMv6-M, C Application Binary Interface (C-ABI) compliant exception model that enables the use of pure C functions as interrupt handlers
  - ◆ Low power sleep mode entry using Wait For Interrupt (WFI), Wait For Event

(WFE) instructions, or the return from interrupt sleep-on-exit feature

- NVIC that features:
  - 32 external interrupt inputs, each with four levels of priority
  - Dedicated Non-Maskable Interrupt (NMI) input.
  - Support for both level-sensitive and pulse-sensitive interrupt lines
  - Wake-up Interrupt Controller (WIC), providing ultra-low power sleep mode support.
- Debug support
  - Four hardware breakpoints.
  - Two watchpoints.
  - Program Counter Sampling Register (PCSR) for non-intrusive code profiling.
  - Single step and vector catch capabilities.
- Bus interfaces:
  - Single 32-bit AMBA-3 AHB-Lite system interface that provides simple integration to all system peripherals and memory.
  - Single 32-bit slave port that supports the DAP (Debug Access Port).

5.2 System Manager

5.2.1 Overview

System management includes these following sections:

- System Resets
- System Memory Map
- System management registers for Part Number ID, chip reset and on-chip controllers reset , multi-functional pin control
- System Timer (SysTick)
- Nested Vectored Interrupt Controller (NVIC)
- System Control registers

5.2.2 System Reset

The system reset can be issued by one of the below listed events. For these reset event flags can be read by RSTSRC register.

- The Power-On Reset
- The low level on the /RESET pin
- Watchdog Time Out Reset
- Low Voltage Reset
- Brown-Out Detector Reset
- CPU Reset
- System Reset

System Reset and Power-On Reset all reset the whole chip including all peripherals. The difference between System Reset and Power-On Reset is external crystal circuit and ISPCON.BS bit. System Reset doesn't reset external crystal circuit and ISPCON.BS bit, but Power-On Reset does.

5.2.3 System Power Distribution

In this chip, the power distribution is divided into three segments.

- Analog power from AVDD and AVSS provides the power for analog components operation.
- Digital power from VDD and VSS supplies the power to the internal regulator which provides a fixed 2.5 V power for digital operation and I/O pins.
- USB transceiver power from VBUS offers the power for operating the USB transceiver. (For NuMicro™ NUC140 only)

The outputs of internal voltage regulators, LDO and VDD33, require an external capacitor which should be located close to the corresponding pin. Analog power (AVDD) should be the same voltage level of the digital power (VDD). Figure 5-2 shows the power distribution of NuMicro™ NUC140 and Figure 5-3 shows the power distribution of NuMicro™ NUC130.

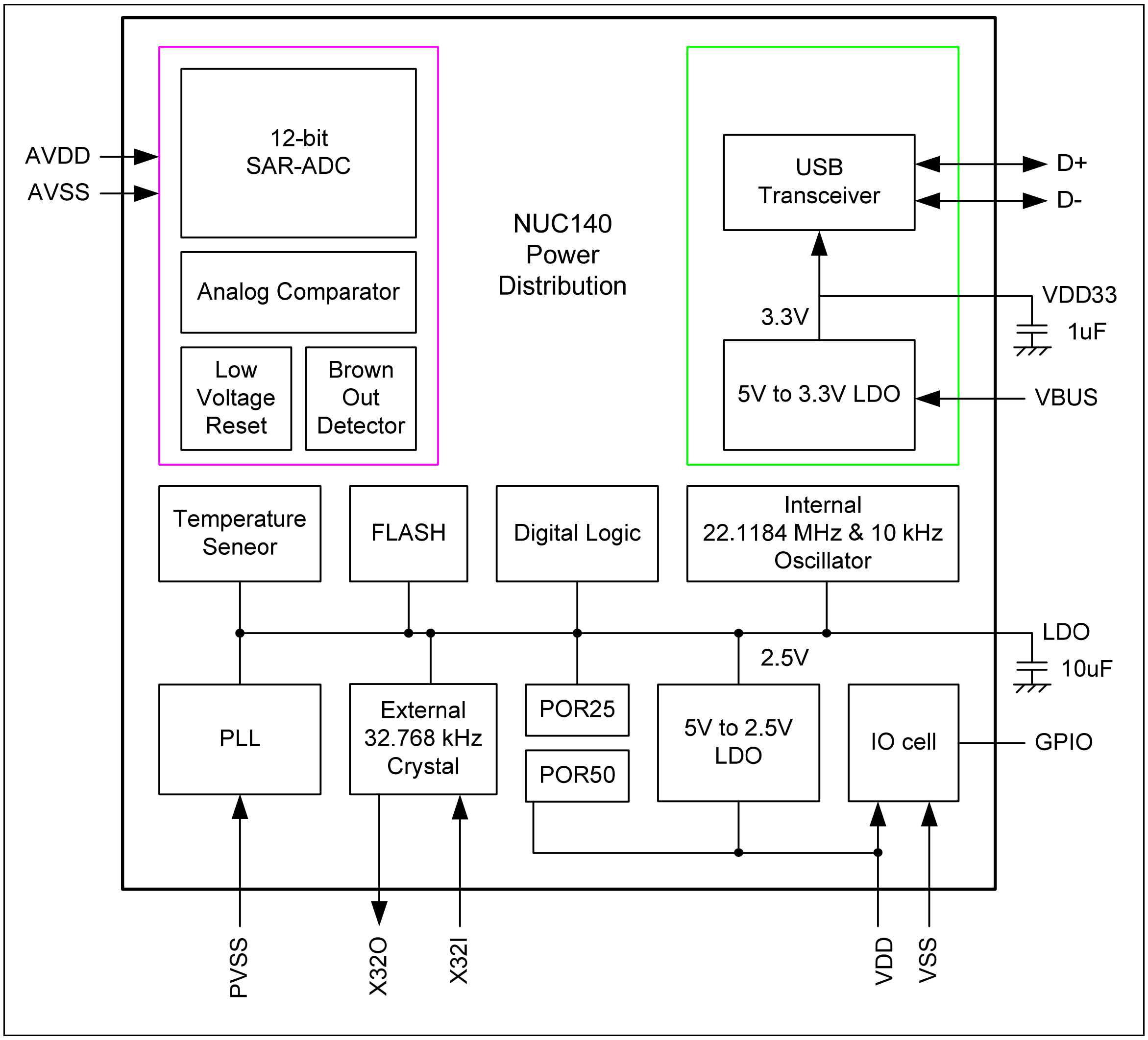


Figure 5-2 NuMicro™ NUC140 Power Distribution Diagram

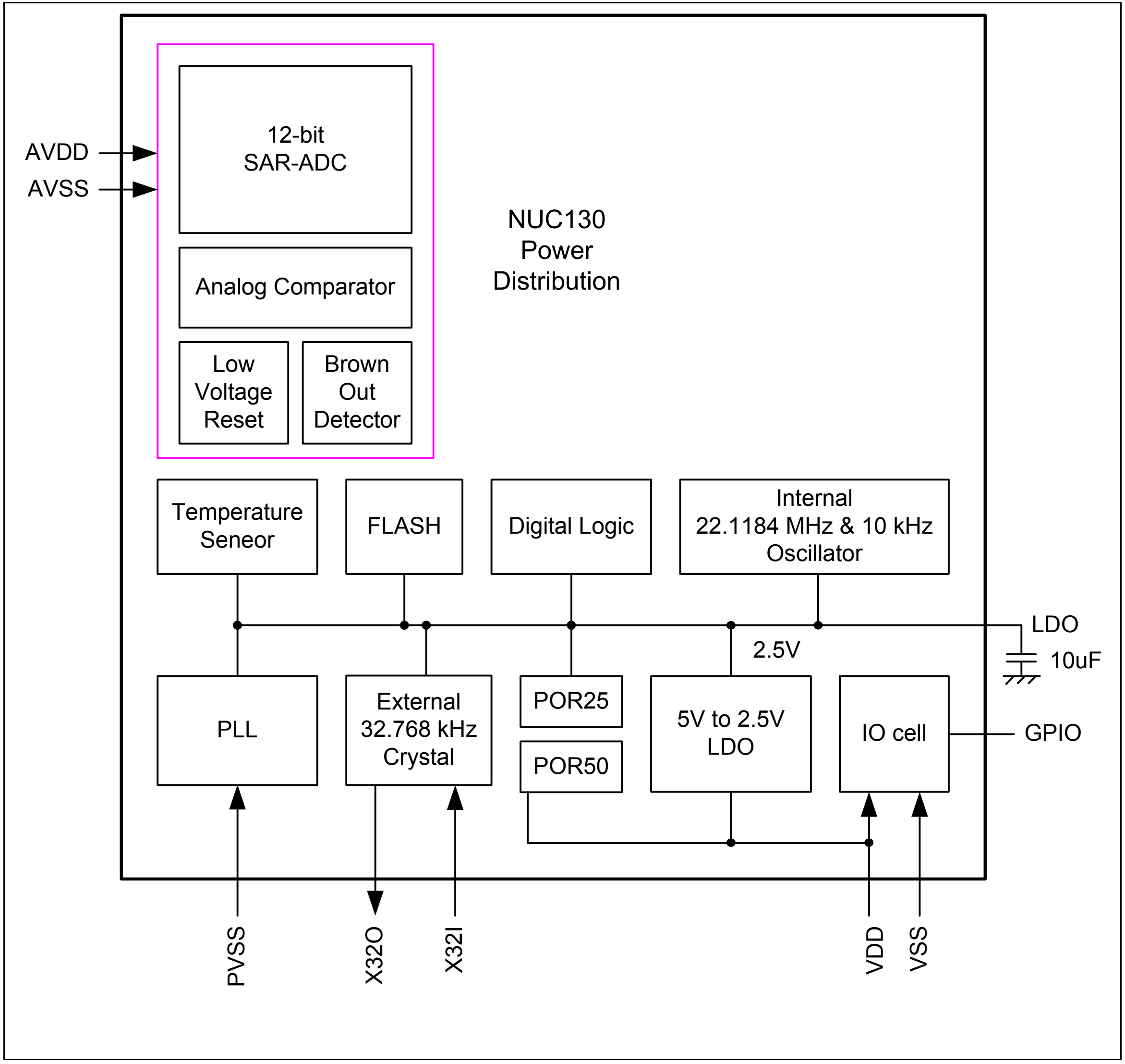


Figure 5-3 NuMicro™ NUC130 Power Distribution Diagram

5.2.4 System Memory Map

NuMicro™ NUC100 Series provides 4G-byte addressing space. The memory locations assigned to each on-chip controllers are shown in the following table. The detailed register definition, memory space, and programming detailed will be described in the following sections for each on-chip peripherals. NuMicro™ NUC100 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)
0x2000_0000 – 0x2000_3FFF	SRAM_BA	SRAM Memory Space (16KB)
0x6000_0000 – 0x6001_FFFF	EXTMEM_BA	External Memory Space (128KB)
AHB Controllers Space (0x5000_0000 – 0x501F_FFFF)		
0x5000_0000 – 0x5000_01FF	GCR_BA	System Global Control Registers
0x5000_0200 – 0x5000_02FF	CLK_BA	Clock Control Registers
0x5000_0300 – 0x5000_03FF	INT_BA	Interrupt Multiplexer Control Registers
0x5000_4000 – 0x5000_7FFF	GPIO_BA	GPIO Control Registers
0x5000_8000 – 0x5000_BFFF	PDMA_BA	Peripheral DMA Control Registers
0x5000_C000 – 0x5000_FFFF	FMC_BA	Flash Memory Control Registers
0x5001_0000 – 0x5001_03FF	EBI_BA	External Bus Interface Control Registers
APB1 Controllers Space (0x4000_0000 ~ 0x400F_FFFF)		
0x4000_4000 – 0x4000_7FFF	WDT_BA	Watchdog Timer Control Registers
0x4000_8000 – 0x4000_BFFF	RTC_BA	Real Time Clock (RTC) Control Register
0x4001_0000 – 0x4001_3FFF	TMR01_BA	Timer0/Timer1 Control Registers
0x4002_0000 – 0x4002_3FFF	I2C0_BA	I^2C0 Interface Control Registers
0x4003_0000 – 0x4003_3FFF	SPI0_BA	SPI0 with master/slave function Control Registers
0x4003_4000 – 0x4003_7FFF	SPI1_BA	SPI1 with master/slave function Control Registers
0x4004_0000 – 0x4004_3FFF	PWMA_BA	PWM0/1/2/3 Control Registers
0x4005_0000 – 0x4005_3FFF	UART0_BA	UART0 Control Registers
0x4006_0000 – 0x4006_3FFF	USBD_BA	USB 2.0 FS device Controller Registers

0x400D_0000 – 0x400D_3FFF	ACMP_BA	Analog Comparator Control Registers
0x400E_0000 – 0x400E_FFFF	ADC_BA	Analog-Digital-Converter (ADC) Control Registers
APB2 Controllers Space (0x4010_0000 ~ 0x401F_FFFF)		
0x4010_0000 – 0x4010_3FFF	PS2_BA	PS/2 Interface Control Registers
0x4011_0000 – 0x4011_3FFF	TMR23_BA	Timer2/Timer3 Control Registers
0x4012_0000 – 0x4012_3FFF	I2C1_BA	I^2C1 Interface Control Registers
0x4013_0000 – 0x4013_3FFF	SPI2_BA	SPI2 with master/slave function Control Registers
0x4013_4000 – 0x4013_7FFF	SPI3_BA	SPI3 with master/slave function Control Registers
0x4014_0000 – 0x4014_3FFF	PWMB_BA	PWM4/5/6/7 Control Registers
0x4015_0000 – 0x4015_3FFF	UART1_BA	UART1 Control Registers
0x4015_4000 – 0x4015_7FFF	UART2_BA	UART2 Control Registers
0x4018_0000 – 0x4018_3FFF	CAN0_BA	CAN0 Bus Control Registers
0x401A_0000 – 0x401A_3FFF	I2S_BA	I^2S Interface Control Registers
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 5-1 Address Space Assignments for On-Chip Controllers

5.2.5 System Manager Control Registers

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

Register	Offset	R/W	Description	Reset Value
GCR_BA = 0x5000_0000				
PDID	GCR_BA+0x00	R	Part Device Identification Number Register	0x0014_0018[1]
RSTSRC	GCR_BA+0x04	R/W	System Reset Source Register	0x0000_00XX
IPRSTC1	GCR_BA+0x08	R/W	IP Reset Control Register1	0x0000_0000
IPRSTC2	GCR_BA+0x0C	R/W	IP Reset Control Register2	0x0000_0000
BODCR	GCR_BA+0x18	R/W	Brown-Out Detector Control Register	0x0000_008X
TEMPCR	GCR_BA+0x1C	R/W	Temperature Sensor Control Register	0x0000_0000
PORCR	GCR_BA+0x24	R/W	Power-On-Reset Controller Register	0x0000_00XX
GPA_MFP	GCR_BA+0x30	R/W	GPIOA Multiple Function and Input Type Control Register	0x0000_0000
GPB_MFP	GCR_BA+0x34	R/W	GPIOB Multiple Function and Input Type Control Register	0x0000_0000
GPC_MFP	GCR_BA+0x38	R/W	GPIOC Multiple Function and Input Type Control Register	0x0000_0000
GPD_MFP	GCR_BA+0x3C	R/W	GPIOD Multiple Function and Input Type Control Register	0x0000_0000
GPE_MFP	GCR_BA+0x40	R/W	GPIOE Multiple Function and Input Type Control Register	0x0000_0000
ALT_MFP	GCR_BA+0x50	R/W	Alternative Multiple Function Pin Control Register	0x0000_0000
REGWRPROT	GCR_BA+0x100	R/W	Register Write Protect register	0x0000_0000

Note: [1] Dependents on part number.

Part Device ID Code Register (PDID)

Register	Offset	R/W	Description	Reset Value
PDID	GCR_BA+0x00	R	Part Device Identification Number Register	0x0014_0018[1]

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

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

Bits	Descriptions	
[31:0]	**PDID**	**Part Device Identification Number** This register reflects device part number code. S/W can read this register to identify which device is used.

System Reset Source Register (RSTSRC)

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

Register	Offset	R/W	Description	Reset Value
RSTSRC	GCR_BA+0x04	R/W	System Reset Source 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**
RSTS_CPU	**Reserved**	**RSTS_SYS**	**RSTS_BOD**	**RSTS_LVR**	**RSTS_WDT**	**RSTS_RESET**	**RSTS_POR**

Bits	Descriptions	
[31:8]	**Reserved**	Reserved
[7]	**RSTS_CPU**	The RSTS_CPU flag is set by hardware if software writes CPU_RST (IPRSTC1[1]) 1 to reset Cortex-M0 CPU kernel and Flash memory controller (FMC). 1 = The Cortex-M0 CPU kernel and FMC are reset by software setting CPU_RST to 1. 0 = No reset from CPU Software can write 1 to clear this bit to zero.
[6]	**Reserved**	Reserved
[5]	**RSTS_SYS**	The RSTS_SYS flag is set by the "reset signal" from the Cortex_M0 kernel to indicate the previous reset source. 1 = The Cortex_M0 had issued the reset signal to reset the system by software writing 1 to bit SYSRESETREQ(AIRCR[2], Application Interrupt and Reset Control Register, address = 0xE000ED0C) in system control registers of Cortex_M0 kernel. 0 = No reset from Cortex_M0 Software can write 1 to clear this bit to zero.
[4]	**RSTS_BOD**	The RSTS_BOD flag is set by the "reset signal" from the Brown-Out-Detector to indicate the previous reset source. 1 = The BOD had issued the reset signal to reset the system 0 = No reset from BOD Software can write 1 to clear this bit to zero.
[3]	**RSTS_LVR**	The RSTS_LVR flag is set by the "reset signal" from the Low-Voltage-Reset controller to indicate the previous reset source. 1 = The LVR controller had issued the reset signal to reset the system.

		0 = No reset from LVR Software can write 1 to clear this bit to zero.
[2]	**RSTS_WDT**	The RSTS_WDT flag is set by the “reset signal” from the watchdog timer to indicate the previous reset source. 1 = The watchdog timer had issued the reset signal to reset the system. 0 = No reset from watchdog timer Software can write 1 to clear this bit to zero.
[1]	**RSTS_RESET**	The RSTS_RESET flag is set by the “reset signal” from the /RESET pin to indicate the previous reset source. 1 = The Pin /RESET had issued the reset signal to reset the system. 0 = No reset from /RESET pin Software can write 1 to clear this bit to zero.
[0]	**RSTS_POR**	The RSTS_POR flag is set by the “reset signal” from the Power-On Reset (POR) controller or bit CHIP_RST (IPRSTC1[0]) to indicate the previous reset source. 1 = The Power-On Reset (POR) or CHIP_RST had issued the reset signal to reset the system. 0 = No reset from POR or CHIP_RST Software can write 1 to clear this bit to zero.

Peripheral Reset Control Register1 (IPRSTC1)

Register	Offset	R/W	Description	Reset Value
IPRSTC1	GCR_BA+0x08	R/W	IP Reset Control 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**
Reserved				**EBI_RST**	**PDMA_RST**	**CPU_RST**	**CHIP_RST**

Bits	Descriptions	
[31:4]	**Reserved**	Reserved
[3]	**EBI_RST**	**EBI Controller Reset** (write-protection bit in NUC130/NUC140 100-pin and 64-pin package) Set this bit to 1 will generate a reset signal to the EBI. User need to set this bit to 0 to release from the reset state. This bit is the protected bit, It means programming this bit needs to write "59h", "16h", "88h" to address 0x5000_0100 to disable register protection. Reference the register REGWRPROT at address GCR_BA+0x100 1 = EBI controller reset 0 = EBI controller normal operation
[2]	**PDMA_RST**	**PDMA Controller Reset** (write-protection bit in NUC130/NUC140) Setting this bit to 1 will generate a reset signal to the PDMA. User need to set this bit to 0 to release from reset state. This bit is the protected bit, It means programming this bit needs to write "59h", "16h", "88h" to address 0x5000_0100 to disable register protection. Reference the register REGWRPROT at address GCR_BA+0x100. 1 = PDMA controller reset 0 = PDMA controller normal operation
[1]	**CPU_RST**	**CPU kernel one shot reset** (write-protection bit) Setting this bit will only reset the CPU kernel and Flash Memory Controller(FMC), and this bit will automatically return to 0 after the 2 clock cycles This bit is the protected bit, It means programming this bit needs to write "59h", "16h", "88h" to address 0x5000_0100 to disable register protection. Reference the register REGWRPROT at address GCR_BA+0x100 1 = CPU one shot reset 0 = CPU normal operation

[0]	**CHIP_RST**	**CHIP one shot reset** (write-protection bit) Setting this bit will reset the whole chip, including CPU kernel and all peripherals, and this bit will automatically return to 0 after the 2 clock cycles. The CHIP_RST 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 CHIP_RST and SYSRESETREQ, please refer to section 5.2.2 This bit is the protected bit. It means programming this bit needs to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Reference the register REGWRPROT at address GCR_BA+0x100 1 = CHIP one shot reset 0 = CHIP normal operation

Peripheral Reset Control Register2 (IPRSTC2)

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

Register	Offset	R/W	Description	Reset Value
IPRSTC2	GCR_BA+0x0C	R/W	Peripheral Controller Reset Control Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		I2S_RST	ADC_RST	USBD_RST	Reserved		CAN0_RST
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PS2_RST	ACMP_RST	PWM47_RST	PWM03_RST	Reserved	UART2_RST	UART1_RST	UART0_RST
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SPI3_RST	SPI2_RST	SPI1_RST	SPI0_RST	Reserved		I2C1_RST	I2C0_RST
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		TMR3_RST	TMR2_RST	TMR1_RST	TMR0_RST	GPIO_RST	Reserved

Bits	Descriptions	
[31:30]	**Reserved**	Reserved
[29]	**I2S_RST**	**I^2S Controller Reset** 1 = I^2S controller reset 0 = I^2S controller normal operation
[28]	**ADC_RST**	**ADC Controller Reset** 1 = ADC controller reset 0 = ADC controller normal operation
[27]	**USBD_RST**	**USB Device Controller Reset** 1 = USB device controller reset 0 = USB device controller normal operation
[26:25]	**Reserved**	Reserved
[24]	**CAN0_RST**	**CAN0 Controller Reset** 1 = CAN0 controller reset 0 = CAN0 controller normal operation
[23]	**PS2_RST**	**PS/2 Controller Reset** 1 = PS/2 controller reset 0 = PS/2 controller normal operation
[22]	**ACMP_RST**	**Analog Comparator Controller Reset** 1 = Analog Comparator controller reset 0 = Analog Comparator controller normal operation

[21]	**PWM47_RST**	**PWM47 controller Reset** 1 = PWM47 controller reset 0 = PWM47 controller normal operation
[20]	**PWM03_RST**	**PWM03 controller Reset** 1 = PWM03 controller reset 0 = PWM03 controller normal operation
[19]	**Reserved**	Reserved
[18]	**UART2_RST**	**UART2 controller Reset** 1 = UART2 controller reset 0 = UART2 controller normal operation
[17]	**UART1_RST**	**UART1 controller Reset** 1 = UART1 controller reset 0 = UART1 controller normal operation
[16]	**UART0_RST**	**UART0 controller Reset** 1 = UART0 controller reset 0 = UART0 controller normal operation
[15]	**SPI3_ RST**	**SPI3 controller Reset** 1 = SPI3 controller reset 0 = SPI3 controller normal operation
[14]	**SPI2_ RST**	**SPI2 controller Reset** 1 = SPI2 controller reset 0 = SPI2 controller normal operation
[13]	**SPI1_ RST**	**SPI1 controller Reset** 1 = SPI1 controller reset 0 = SPI1 controller normal operation
[12]	**SPI0_ RST**	**SPI0 controller Reset** 1 = SPI0 controller reset 0 = SPI0 controller normal operation
[11:10]	**Reserved**	Reserved
[9]	**I2C1_RST**	**I^{2}C1 controller Reset** 1 = I^{2}C1 controller reset 0 = I^{2}C1 controller normal operation
[8]	**I2C0_RST**	**I^{2}C0 controller Reset** 1 = I^{2}C0 controller reset 0 = I^{2}C0 controller normal operation
[7:6]	**Reserved**	Reserved
[5]	**TMR3_RST**	**Timer3 controller Reset** 1 = Timer3 controller reset

		0 = Timer3 controller normal operation
[4]	**TMR2_RST**	**Timer2 controller Reset** 1 = Timer2 controller reset 0 = Timer2 controller normal operation
[3]	**TMR1_RST**	**Timer1 controller Reset** 1 = Timer1 controller reset 0 = Timer1 controller normal operation
[2]	**TMR0_RST**	**Timer0 controller Reset** 1 = Timer0 controller reset 0 = Timer0 controller normal operation
[1]	**GPIO_RST**	**GPIO controller Reset** 1 = GPIO controller reset 0 = GPIO controller normal operation
[0]	**Reserved**	Reserved

Brown-Out Detector Control Register (BODCR)

Partial of the BODCR control registers values are initiated by the flash configuration and partial bits are write-protected bit. Programming write-protected bits needs to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Reference the register REGWRPROT at address GCR_BA+0x100

Register	Offset	R/W	Description	Reset Value
BODCR	GCR_BA+0x18	R/W	Brown-Out Detector Control Register	0x0000_008X

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**
LVR_EN	**BOD_OUT**	**BOD_LPM**	**BOD_INTF**	**BOD_RSTEN**	**BOD_VL**		**BOD_EN**

Bits	Descriptions	
[31:8]	**Reserved**	Reserved
[7]	**LVR_EN**	**Low Voltage Reset Enable** (write-protection bit) The LVR function reset the chip when the input power voltage is lower than LVR circuit setting. LVR function is enabled in default. 1 = Enabled Low Voltage Reset function – After enabling the bit, the LVR function will be active with 100uS delay for LVR output stable. (Default). 0 = Disabled Low Voltage Reset function This bit is the protected bit. It means programming this needs to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Reference the register REGWRPROT at address GCR_BA+0x100
[6]	**BOD_OUT**	**Brown-Out Detector output status** 1 = Brown-Out Detector output status is 1. It means the detected voltage is lower than BOD_VL setting. If the BOD_EN is 0, BOD function disabled , this bit always responds 0 0 = Brown-Out Detector output status is 0. It means the detected voltage is higher than BOD_VL setting or BOD_EN is 0
[5]	**BOD_LPM**	**Brown-Out Detector Low power Mode** (write-protection bit) 1 = Enable the BOD low power mode 0 = BOD operate in normal mode (default) The BOD consumes about 100 uA in normal mode, the low power mode can reduce the current to about 1/10 but slow the BOD response. This bit is the protected bit. It means programming this needs to write “59h”, “16h”,

		"88h" to address 0x5000_0100 to disable register protection. Reference the register REGWRPROT at address GCR_BA+0x100.
[4]	**BOD_INTF**	**Brown-Out Detector Interrupt Flag** 1 = When Brown-Out Detector detects the VDD is dropped down through the voltage of BOD_VL setting or the VDD is raised up through the voltage of BOD_VL setting, this bit is set to 1 and the Brown-Out interrupt is requested if Brown-Out interrupt is enabled. 0 = Brown-Out Detector does not detect any voltage draft at VDD down through or up through the voltage of BOD_VL setting. Software can write 1 to clear this bit to zero.
[3]	**BOD_RSTEN**	**Brown-Out Reset Enable** (write-protection bit) 1 = Enable the Brown-Out "RESET" function While the Brown-Out Detector function is enabled (BOD_EN high) and BOD reset function is enabled (BOD_RSTEN high), BOD will assert a signal to reset chip when the detected voltage is lower than the threshold (BOD_OUT high). 0 = Enable the Brown-Out "INTERRUPT" function While the BOD function is enabled (BOD_EN high) and BOD interrupt function is enabled (BOD_RSTEN low), BOD will assert an interrupt if BOD_OUT is high. BOD interrupt will keep till to the BOD_EN set to 0. BOD interrupt can be blocked by disabling the NVIC BOD interrupt or disabling BOD function (set BOD_EN low). The default value is set by flash controller user configuration register config0 bit[20]. This bit is the protected bit. It means programming this needs to write "59h", "16h", "88h" to address 0x5000_0100 to disable register protection. Reference the register REGWRPROT at address GCR_BA+0x100.
[2:1]	**BOD_VL**	**Brown-Out Detector Threshold Voltage Selection** (write-protection bits) The default value is set by flash controller user configuration register config0 bit[22:21] This bit is the protected bit. It means programming this needs to write "59h", "16h", "88h" to address 0x5000_0100 to disable register protection. Reference the register REGWRPROT at address GCR_BA+0x100. <table><tr><th>BOV_VL[1]</th><th>BOV_VL[0]</th><th>Brown-Out voltage</th></tr><tr><td>1</td><td>1</td><td>4.5 V</td></tr><tr><td>1</td><td>0</td><td>3.8 V</td></tr><tr><td>0</td><td>1</td><td>2.7 V</td></tr><tr><td>0</td><td>0</td><td>2.2 V</td></tr></table>
[0]	**BOD_EN**	**Brown-Out Detector Enable** (write-protection bit) The default value is set by flash controller user configuration register config0 bit[23] 1 = Brown-Out Detector function is enabled 0 = Brown-Out Detector function is disabled This bit is the protected bit. It means programming this needs to write "59h", "16h", "88h" to address 0x5000_0100 to disable register protection. Reference the register REGWRPROT at address GCR_BA+0x100.

Temperature Sensor Control Register (TEMPCR)

Register	Offset	R/W	Description	Reset Value
TEMPCR	GCR_BA+0x1C	R/W	Temperature Sensor 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							**VTEMP_EN**

Bits	Descriptions	
[31:1]	**Reserved**	Reserved
[0]	**VTEMP_EN**	**Temperature sensor Enable** This bit is used to enable/disable temperature sensor function. 1 = Enabled temperature sensor function 0 = Disabled temperature sensor function (default) After this bit is set to 1, the value of temperature can get from ADC conversion result by ADC channel selecting channel 7 and alternative multiplexer channel selecting temperature sensor. Detail ADC conversion function please reference ADC function chapter.

Power-On-Reset Control Register (PORCR)

Register	Offset	R/W	Description	Reset Value
PORCR	GCR_BA+0x24	R/W	Power-On-Reset Controller 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**
POR_DIS_CODE[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
POR_DIS_CODE[7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	Reserved
[15:0]	**POR_DIS_CODE**	**The register is used for the Power-On-Reset enable control** (write-protection bits) When power 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: /RESET, Watchdog, LVR reset, BOD reset, ICE reset command and the software-chip reset function This bit is the protected bit. It means programming this needs to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Reference the register REGWRPROT at address GCR_BA+0x100.

Multiple Function Pin GPIOA Control Register (GPA_MFP)

Register	Offset	R/W	Description	Reset Value
GPA_MFP	GCR_BA+0x30	R/W	GPIOA Multiple Function and Input Type Control Register	0x0000_0000

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th></tr>
<tr><td colspan="8">GPA_TYPE[15:8]</td></tr>
<tr><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="8">GPA_TYPE[7:0]</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th></tr>
<tr><td colspan="8">GPA_MFP[15:8]</td></tr>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="8">GPA_MFP[7:0]</td></tr>
</table>

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[31:16]</td><td>GPA_TYPEn</td><td>1 = Enable GPIOA[15:0] I/O input Schmitt Trigger function
0 = Disable GPIOA[15:0] I/O input Schmitt Trigger function</td></tr>
<tr><td>[15]</td><td>GPA_MFP15</td><td>PA.15 Pin Function Selection
The pin function depends on GPA_MFP15 and PA15_I2SMCLK (ALT_MFP[9]).
<table>
<tr><th>PA15_I2SMCLK</th><th>GPA_MFP[15]</th><th>PA.15 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>PWM3 (PWM)</td></tr>
<tr><td>1</td><td>1</td><td>I2SMCLK (I^2S)</td></tr>
</table></td></tr>
<tr><td>[14]</td><td>GPA_MFP14</td><td>PA.14 Pin Function Selection
The pin function depends on GPA_MFP14 and EBI_HB_EN[7] (ALT_MFP[23]) and EBI_EN (ALT_MFP[11]).
<table>
<tr><th>EBI_HB_EN[7]</th><th>EBI_EN</th><th>GPA_MFP[14]</th><th>PA.14 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>PWM2 (PWM)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>AD15 (EBI AD bus bit 15)</td></tr>
</table></td></tr>
<tr><td>[13]</td><td>GPA_MFP13</td><td>PA.13 Pin Function Selection
The pin function depends on GPA_MFP13 and EBI_HB_EN[6] (ALT_MFP[22]) and EBI_EN (ALT_MFP[11]).
<table>
<tr><th>EBI_HB_EN[6]</th><th>EBI_EN</th><th>GPA_MFP[13]</th><th>PA.13 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>PWM1 (PWM)</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td>1</td><td>1</td><td>1</td><td>AD14 (EBI AD bus bit 14)</td></tr></table></td></tr>
<tr><td>[12]</td><td>GPA_MFP12</td><td>PA.12 Pin Function Selection
The pin function depends on GPA_MFP12 and EBI_HB_EN[5] (ALT_MFP[21]) and EBI_EN (ALT_MFP[11]).<table><tr><th>EBI_HB_EN[5]</th><th>EBI_EN</th><th>GPA_MFP[12]</th><th>PA.12 function</th></tr><tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>0</td><td>1</td><td>PWM0 (PWM)</td></tr><tr><td>1</td><td>1</td><td>1</td><td>AD13 (EBI AD bus bit 13)</td></tr></table></td></tr>
<tr><td>[11]</td><td>GPA_MFP11</td><td>PA.11 Pin Function Selection
The pin function depends on GPA_MFP11 and EBI_EN (ALT_MFP[11]).<table><tr><th>EBI_EN</th><th>GPA_MFP[11]</th><th>PA.11 function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>SCL1 (I^{2}C)</td></tr><tr><td>1</td><td>1</td><td>nRD (EBI)</td></tr></table></td></tr>
<tr><td>[10]</td><td>GPA_MFP10</td><td>PA.10 Pin Function Selection
The pin function depends on GPA_MFP10 and EBI_EN (ALT_MFP[11]).<table><tr><th>EBI_EN</th><th>GPA_MFP[10]</th><th>PA.10 function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>SDA1 (I^{2}C)</td></tr><tr><td>1</td><td>1</td><td>nWR (EBI)</td></tr></table></td></tr>
<tr><td>[9]</td><td>GPA_MFP9</td><td>PA.9 Pin Function Selection
1 = The I^{2}C0 SCL function is selected to the pin PA.9
0 = The GPIOA[9] is selected to the pin PA.9</td></tr>
<tr><td>[8]</td><td>GPA_MFP8</td><td>PA.8 Pin Function Selection
1 = The I^{2}C0 SDA function is selected to the pin PA.8
0 = The GPIOA[8] is selected to the pin PA.8</td></tr>
<tr><td>[7]</td><td>GPA_MFP7</td><td>PA.7 Pin Function Selection
The pin function depends on GPA_MFP7 and PA7_S21 (ALT_MFP[2]) and EBI_EN (ALT_MFP[11]).<table><tr><th>EBI_EN</th><th>PA7_S21</th><th>GPA_MFP[7]</th><th>PA.7 function</th></tr><tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>0</td><td>1</td><td>ADC7 (ADC)</td></tr><tr><td>0</td><td>1</td><td>1</td><td>SPISS21 (SPI2)</td></tr><tr><td>1</td><td>0</td><td>1</td><td>AD6 (EBI AD bus bit 6)</td></tr></table></td></tr>
<tr><td>[6]</td><td>GPA_MFP6</td><td>PA.6 Pin Function Selection
The pin function depends on GPA_MFP6 and EBI_EN (ALT_MFP[11]).<table><tr><th>EBI_EN</th><th>GPA_MFP[6]</th><th>PA.6 function</th></tr></table></td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>ADC6 (ADC)</td></tr>
<tr><td>1</td><td>1</td><td>AD7 (EBI AD bus bit 7)</td></tr>
</table>
</td></tr>
<tr><td>[5]</td><td>GPA_MFP5</td><td>
PA.5 Pin Function Selection
The pin function depends on GPA_MFP5 and EBI_HB_EN[0] (ALT_MFP[16]) and EBI_EN (ALT_MFP[11]).
<table>
<tr><th>EBI_HB_EN[0]</th><th>EBI_EN</th><th>GPA_MFP[5]</th><th>PA.5 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>ADC5 (ADC)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>AD8 (EBI AD bus bit 8)</td></tr>
</table>
</td></tr>
<tr><td>[4]</td><td>GPA_MFP4</td><td>
PA.4 Pin Function Selection
The pin function depends on GPA_MFP4 and EBI_HB_EN[1] (ALT_MFP[17]) and EBI_EN (ALT_MFP[11]).
<table>
<tr><th>EBI_HB_EN[1]</th><th>EBI_EN</th><th>GPA_MFP[4]</th><th>PA.4 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>ADC4 (ADC)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>AD9 (EBI AD bus bit 9)</td></tr>
</table>
</td></tr>
<tr><td>[3]</td><td>GPA_MFP3</td><td>
PA.3 Pin Function Selection
The pin function depends on GPA_MFP3 and EBI_HB_EN[2] (ALT_MFP[18]) and EBI_EN (ALT_MFP[11]).
<table>
<tr><th>EBI_HB_EN[2]</th><th>EBI_EN</th><th>GPA_MFP[3]</th><th>PA.3 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>ADC3 (ADC)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>AD10 (EBI AD bus bit 10)</td></tr>
</table>
</td></tr>
<tr><td>[2]</td><td>GPA_MFP2</td><td>
PA.2 Pin Function Selection
The pin function depends on GPA_MFP2 and EBI_HB_EN[3] (ALT_MFP[19]) and EBI_EN (ALT_MFP[11]).
<table>
<tr><th>EBI_HB_EN[3]</th><th>EBI_EN</th><th>GPA_MFP[2]</th><th>PA.2 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>ADC2 (ADC)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>AD11 (EBI AD bus bit 11)</td></tr>
</table>
</td></tr>
<tr><td>[1]</td><td>GPA_MFP1</td><td>
PA.1 Pin Function Selection
The pin function depends on GPA_MFP1 and EBI_HB_EN[4] (ALT_MFP[20]) and EBI_EN (ALT_MFP[11]).
<table>
<tr><th>EBI_HB_EN[4]</th><th>EBI_EN</th><th>GPA_MFP[1]</th><th>PA.1 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>ADC1 (ADC)</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><td>1</td><td>1</td><td>1</td><td>AD12 (EBI AD bus bit 12)</td></tr></table></td></tr>
<tr><td>[0]</td><td>GPA_MFP0</td><td>PA.0 Pin Function Selection
1 = The ADC0 (Analog-to-Digital converter channel 0) function is selected to the pin PA.0
0 = The GPIOA[0] is selected to the pin PA.0</td></tr>
</table>

Multiple Function Pin GPIOB Control Register (GPB_MFP)

Register	Offset	R/W	Description	Reset Value
GPB_MFP	GCR_BA+0x34	R/W	GPIOB Multiple Function and Input Type Control Register	0x0000_0000

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

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[31:16]</td><td>GPB_TYPEn</td><td>1 = Enable GPIOB[15:0] I/O input Schmitt Trigger function
0 = Disable GPIOB[15:0] I/O input Schmitt Trigger function</td></tr>
<tr><td>[15]</td><td>GPB_MFP15</td><td>PB.15 Pin Function Selection
The pin function depends on GPB_MFP15 and PB15_T0EX (ALT_MFP[24])
<table>
<tr><th>PB15_T0EX</th><th>GPB_MFP[15]</th><th>PB.15 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>/INT1</td></tr>
<tr><td>1</td><td>1</td><td>T0EX (TMR0)</td></tr>
</table></td></tr>
<tr><td>[14]</td><td>GPB_MFP14</td><td>PB.14 Pin Function Selection
The pin function depends on GPB_MFP14 and PB14_S31 (ALT_MFP[3])
<table>
<tr><th>PB14_S31</th><th>GPB_MFP[14]</th><th>PB.14 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>/INT0</td></tr>
<tr><td>1</td><td>1</td><td>SPISS31 (SPI3)</td></tr>
</table></td></tr>
<tr><td>[13]</td><td>GPB_MFP13</td><td>PB.13 Pin Function Selection
The pin function depends on GPB_MFP13 and EBI_EN (ALT_MFP[11]).
<table>
<tr><th>EBI_EN</th><th>GPB_MFP[13]</th><th>PB.13 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>CPO1 (CMP)</td></tr>
<tr><td>1</td><td>1</td><td>AD1 (EBI AD bus bit 1)</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>[12]</td><td>GPB_MFP12</td><td>
PB.12 Pin Function Selection
The pin function depends on GPB_MFP12 and PB12_CLKO (ALT_MFP[10]) and EBI_EN (ALT_MFP[11]).
<table>
<tr><th>EBI_EN</th><th>PB12_CLKO</th><th>GPB_MFP[12]</th><th>PB.12 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>CPO0 (CMP)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>CLKO (Clock Driver output)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>AD0 (EBI AD bus bit 0)</td></tr>
</table>
</td></tr>
<tr><td>[11]</td><td>GPB_MFP11</td><td>
PB.11 Pin Function Selection
The pin function depends on GPB_MFP11 and PB11_PWM4 (ALT_MFP[4]).
<table>
<tr><th>PB11_PWM4</th><th>GPB_MFP[11]</th><th>PB.11 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>TM3</td></tr>
<tr><td>1</td><td>1</td><td>PWM4 (PWM)</td></tr>
</table>
</td></tr>
<tr><td>[10]</td><td>GPB_MFP10</td><td>
PB.10 Pin Function Selection
The pin function depends on GPB_MFP10 and PB10_S01 (ALT_MFP[0]).
<table>
<tr><th>PB10_S01</th><th>GPB_MFP[10]</th><th>PB.10 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>TM2</td></tr>
<tr><td>1</td><td>1</td><td>SPISS01 (SPI0)</td></tr>
</table>
</td></tr>
<tr><td>[9]</td><td>GPB_MFP9</td><td>
PB.9 Pin Function Selection
The pin function depends on GPB_MFP9 and PB9_S11 (ALT_MFP[1]).
<table>
<tr><th>PB9_S11</th><th>GPB_MFP[9]</th><th>PB.9 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>TM1</td></tr>
<tr><td>1</td><td>1</td><td>SPISS11 (SPI1)</td></tr>
</table>
</td></tr>
<tr><td>[8]</td><td>GPB_MFP8</td><td>
PB.8 Pin Function Selection
1 = The TM0 (Timer/Counter external trigger clock input) function is selected to the pin PB.8
0 = The GPIOB[8] is selected to the pin PB.8
</td></tr>
<tr><td>[7]</td><td>GPB_MFP7</td><td>
PB.7 Pin Function Selection
The pin function depends on GPB_MFP7 and EBI_EN (ALT_MFP[11]).
<table>
<tr><th>EBI_EN</th><th>GPB_MFP[7]</th><th>PB.7 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>CTS1 (UART1)</td></tr>
<tr><td>1</td><td>1</td><td>nCS (EBI)</td></tr>
</table>
</td></tr>
<tr><td>[6]</td><td>GPB_MFP6</td><td>PB.6 Pin Function Selection</td></tr>
</table>

<table>
<tr><td></td><td></td><td>The pin function depends on GPB_MFP6 and EBI_EN (ALT_MFP[11]).
<table>
<tr><th>EBI_EN</th><th>GPB_MFP[6]</th><th>PB.6 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>RTS1 (UART1)</td></tr>
<tr><td>1</td><td>1</td><td>ALE (EBI)</td></tr>
</table></td></tr>
<tr><td>[5]</td><td>GPB_MFP5</td><td>PB. 5 Pin Function Selection
1 = The UART1 TXD function is selected to the pin PB.5
0 = The GPIOB[5] is selected to the pin PB.5</td></tr>
<tr><td>[4]</td><td>GPB_MFP4</td><td>PB.4 Pin Function Selection
1 = The UART1 RXD function is selected to the pin PB.4
0 = The GPIOB[4] is selected to the pin PB.4</td></tr>
<tr><td>[3]</td><td>GPB_MFP3</td><td>PB.3 Pin Function Selection
The pin function depends on GPB_MFP3 and EBI_nWRH_EN (ALT_MFP[14]) and EBI_EN (ALT_MFP[11]) and PB3_T3EX (ALT_MFP[27]).
<table>
<tr><th>EBI_nWRH_EN</th><th>EBI_EN</th><th>GPB_MFP[3]</th><th>PB3_T3EX</th><th>PB.3 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>CTS0 (UART0)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>nWRH (EBI write high byte enable)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>T3EX (TMR3)</td></tr>
</table></td></tr>
<tr><td>[2]</td><td>GPB_MFP2</td><td>PB.2 Pin Function Selection
The pin function depends on GPB_MFP2 and EBI_nWRL_EN (ALT_MFP[13]) and EBI_EN (ALT_MFP[11]) and PB2_T2EX (ALT_MFP[26]).
<table>
<tr><th>EBI_nWRL_EN</th><th>EBI_EN</th><th>GPB_MFP[2]</th><th>PB2_T2EX</th><th>PB.2 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>RTS0 (UART0)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>nWRL (EBI write low byte enable)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>T2EX (TMR2)</td></tr>
</table></td></tr>
<tr><td>[1]</td><td>GPB_MFP1</td><td>PB.1 Pin Function Selection
1 = The UART0 TXD function is selected to the pin PB.1
0 = The GPIOB[1] is selected to the pin PB.1</td></tr>
<tr><td>[0]</td><td>GPB_MFP0</td><td>PB.0 Pin Function Selection
1 = The UART0 RXD function is selected to the pin PB.0
0 = The GPIOB[0] is selected to the pin PB.0</td></tr>
</table>

Multiple Function Pin GPIOC Control Register (GPC_MFP)

Register	Offset	R/W	Description	Reset Value
GPC_MFP	GCR_BA+0x38	R/W	GPIOC Multiple Function and Input Type Control Register	0x0000_0000

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

Bits	Descriptions	
[31:16]	**GPC_TYPEn**	1 = Enable GPIOC[15:0] I/O input Schmitt Trigger function 0 = Disable GPIOC[15:0] I/O input Schmitt Trigger function
[15]	**GPC_MFP15**	**PC.15 Pin Function Selection** The pin function depends on GPC_MFP15 and EBI_EN (ALT_MFP[11]). EBI_EN / GPC_MFP[15] / PC.15 function 0 / 0 / GPIO 0 / 1 / CPN1 (CMP) 1 / 1 / AD3 (EBI AD bus bit 3)
[14]	**GPC_MFP14**	**PC.14 Pin Function Selection** The pin function depends on GPC_MFP14 and EBI_EN (ALT_MFP[11]). EBI_EN / GPC_MFP[14] / PC.14 function 0 / 0 / GPIO 0 / 1 / CPP1 (CMP) 1 / 1 / AD2 (EBI AD bus bit 2)
[13]	**GPC_MFP13**	**PC.13 Pin Function Selection** 1 = The SPI1 MOSI1 (master output, slave input pin-1) function is selected to the pin PC.13 0 = The GPIOC[13] is selected to the pin PC.13
[12]	**GPC_MFP12**	**PC.12 Pin Function Selection** 1 = The SPI1 MISO1 (master input, slave output pin-1) function is selected to the pin PC.12 0 = The GPIOC[12] is selected to the pin PC.12

<table>
<tr><td>[11]</td><td>GPC_MFP11</td><td>PC.11 Pin Function Selection
1 = The SPI1 MOSI0 (master output, slave input pin-0) function is selected to the pin PC.11
0 = The GPIOC[11] is selected to the pin PC.11</td></tr>
<tr><td>[10]</td><td>GPC_MFP10</td><td>PC.10 Pin Function Selection
1 = The SPI1 MISO0 (master input, slave output pin-0) function is selected to the pin PC.10
0 = The GPIOC[10] is selected to the pin PC.10</td></tr>
<tr><td>[9]</td><td>GPC_MFP9</td><td>PC.9 Pin Function Selection
1 = The SPI1 SPICLK function is selected to the pin PC.9
0 = The GPIOC[9] is selected to the pin PC.9</td></tr>
<tr><td>[8]</td><td>GPC_MFP8</td><td>PC.8 Pin Function Selection
The pin function depends on GPC_MFP8 and EBI_MCLK_EN (ALT_MFP[12]) and EBI_EN (ALT_MFP[11]).
<table>
<tr><th>EBI_MCLK_EN</th><th>EBI_EN</th><th>GPC_MFP[8]</th><th>PC.8 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>SPISS10 (SPI1)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>MCLK (EBI Clock output)</td></tr>
</table></td></tr>
<tr><td>[7]</td><td>GPC_MFP7</td><td>PC.7 Pin Function Selection
The pin function depends on GPC_MFP7 and EBI_EN (ALT_MFP[11]).
<table>
<tr><th>EBI_EN</th><th>GPC_MFP[7]</th><th>PC.7 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>CPN0 (CMP)</td></tr>
<tr><td>1</td><td>1</td><td>AD5 (EBI AD bus bit 5)</td></tr>
</table></td></tr>
<tr><td>[6]</td><td>GPC_MFP6</td><td>PC.6 Pin Function Selection
The pin function depends on GPC_MFP6 and EBI_EN (ALT_MFP[11]).
<table>
<tr><th>EBI_EN</th><th>GPC_MFP[6]</th><th>PC.6 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>CPP0 (CMP)</td></tr>
<tr><td>1</td><td>1</td><td>AD4 (EBI AD bus bit 4)</td></tr>
</table></td></tr>
<tr><td>[5]</td><td>GPC_MFP5</td><td>PC.5 Pin Function Selection
1 = The SPI0 MOSI1 (master output, slave input pin-1) function is selected to the pin PC.5
0 = The GPIOC[5] is selected to the pin PC.5</td></tr>
<tr><td>[4]</td><td>GPC_MFP4</td><td>PC.4 Pin Function Selection
1 = The SPI0 MISO1 (master input, slave output pin-1) function is selected to the pin PC.4
0 = The GPIOC[4] is selected to the pin PC.4</td></tr>
<tr><td>[3]</td><td>GPC_MFP3</td><td>PC.3 Pin Function Selection</td></tr>
</table>

<table>
<tr>
<td></td>
<td></td>
<td>Bits PC3_I2SDO (ALT_MFP[8]) and GPC_MFP[3] determine the PC.3 function.
<table>
<tr><th>PC3_I2SDO</th><th>GPC_MFP[3]</th><th>PC.3 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MOSI00 (SPI0)</td></tr>
<tr><td>1</td><td>1</td><td>I2SDO (I^{2}S)</td></tr>
</table>
</td>
</tr>
<tr>
<td>[2]</td>
<td>GPC_MFP2</td>
<td>PC.2 Pin Function Selection
Bits PC2_I2SDI (ALT_MFP[7]) and GPC_MFP[2] determine the PC.2 function.
<table>
<tr><th>PC2_I2SDI</th><th>GPC_MFP[2]</th><th>PC.2 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MISO00 (SPI0)</td></tr>
<tr><td>1</td><td>1</td><td>I2SDI (I^{2}S)</td></tr>
</table>
</td>
</tr>
<tr>
<td>[1]</td>
<td>GPC_MFP1</td>
<td>PC.1 Pin Function Selection
Bits PC1_I2SBCLK (ALT_MFP[6]) and GPC_MFP[1] determine the PC.1 function.
<table>
<tr><th>PC1_I2SBCLK</th><th>GPC_MFP[1]</th><th>PC.1 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>SPICLK0 (SPI0)</td></tr>
<tr><td>1</td><td>1</td><td>I2SBCLK (I^{2}S)</td></tr>
</table>
</td>
</tr>
<tr>
<td>[0]</td>
<td>GPC_MFP0</td>
<td>PC.0 Pin Function Selection
Bits PC0_I2SLRCLK (ALT_MFP[5]) and GPC_MFP[0] determine the PC.0 function.
<table>
<tr><th>PC0_I2SLRCLK</th><th>GPC_MFP[0]</th><th>PC.0 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>SPISS00 (SPI0)</td></tr>
<tr><td>1</td><td>1</td><td>I2SLRCLK (I^{2}S)</td></tr>
</table>
</td>
</tr>
</table>

Multiple Function Pin GPIOD Control Register (GPD_MFP)

Register	Offset	R/W	Description	Reset Value
GPD_MFP	GCR_BA+0x3C	R/W	GPIOD Multiple Function and Input Type Control Register	0x0000_0000

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

Bits	Descriptions	
[31:16]	**GPD_TYPEn**	1 = Enable GPIOD[15:0] I/O input Schmitt Trigger function 0 = Disable GPIOD[15:0] I/O input Schmitt Trigger function
[15]	**GPD_MFP15**	**PD.15 Pin Function Selection** 1 = The UART2 TXD function is selected to the pin PD.15 0 = The GPIOD[15] selected to the pin PD.15
[14]	**GPD_MFP14**	**PD.14 Pin Function Selection** 1 = The UART2 RXD function is selected to the pin PD.14 0 = The GPIOD[14] selected to the pin PD.14
[13]	**GPD_MFP13**	**PD.13 Pin Function Selection** 1 = The SPI3 MOSI1 (master output, slave input pin-1) function is selected to the pin PD.13 0 = The GPIOD[13] is selected to the pin PD.13
[12]	**GPD_MFP12**	**PD.12 Pin Function Selection** 1 = The SPI3 MISO1 (master input, slave output pin-1) function is selected to the pin PD.12 0 = The GPIOD[12] is selected to the pin PD.12
[11]	**GPD_MFP11**	**PD.11 Pin Function Selection** 1 = The SPI3 MOSI0 (master output, slave input pin-0) function is selected to the pin PD.11 0 = The GPIOD[11] is selected to the pin PD.11
[10]	**GPD_MFP10**	**PD.10 Pin Function Selection** 1 = The SPI3 MISO0 (master input, slave output pin-0) function is selected to the pin

		PD.10 0 = The GPIOD[10] is selected to the pin PD.10
[9]	**GPD_MFP9**	**PD.9 Pin Function Selection** 1 = The SPI3 SPICLK function is selected to the pin PD.9 0 = The GPIOD[9] is selected to the pin PD.9
[8]	**GPD_MFP8**	**PD.8 Pin Function Selection** 1 = The SPI3 SS30 function is selected to the pin PD8 0 = The GPIOD[8] is selected to the pin PD8
[7]	**GPD_MFP7**	**PD.7 Pin Function Selection** 1 = The CAN0 TX function is selected to the pin PD.7 0 = The GPIOD[7] is selected to the pin PD.7
[6]	**GPD_MFP6**	**PD.6 Pin Function Selection** 1 = The CAN0 RX function is selected to the pin PD.6 0 = The GPIOD[6] is selected to the pin PD.6
[5]	**GPD_MFP5**	**PD.5 Pin Function Selection** 1 = The SPI2 MOSI1 (master output, slave input pin-1) function is selected to the pin PD.5 0 = The GPIOD[5] is selected to the pin PD.5
[4]	**GPD_MFP4**	**PD.4 Pin Function Selection** 1 = The SPI2 MISO1 (master input, slave output pin-1) function is selected to the pin PD.4 0 = The GPIOD[4]is selected to the pin PD.4
[3]	**GPD_MFP3**	**PD.3 Pin Function Selection** 1 = The SPI2 MOSI0 (master output, slave input pin-0) function is selected to the pin PD.3 0 = The GPIOD[3] is selected to the pin PD.3
[2]	**GPD_MFP2**	**PD.2 Pin Function Selection** 1 = The SPI2 MISO0 (master input, slave output pin-0) function is selected to the pin PD.2 0 = The GPIOD[2] is selected to the pin PD.2
[1]	**GPD_MFP1**	**PD.1 Pin Function Selection** 1 = The SPI2 SPICLK function is selected to the pin PD.1 0 = The GPIOD[1] is selected to the pin PD.1
[0]	**GPD_MFP0**	**PD.0 Pin Function Selection** 1 = The SPI2 SS20 function is selected to the pin PD.0 0 = The GPIOD[0] is selected to the pin PD.0

Multiple Function Pin GPIOE Control Register (GPE_MFP)

Register	Offset	R/W	Description	Reset Value
GPE_MFP	GCR_BA+0x40	R/W	GPIOE Multiple Function and Input Type Control Register	0x0000_0000

31	30	29	28	27	26	25	24
GPE_TYPE[15:8]							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
GPE_TYPE[7:0]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**GPE_MFP5**	**Reserved**			**GPE_MFP1**	**GPE_MFP0**

Bits	Descriptions	
[31:16]	**GPE_TYPEn**	1 = Enable GPIOE[15:0] I/O input Schmitt Trigger function 0 = Disable GPIOE[15:0] I/O input Schmitt Trigger function
[15:6]	**Reserved**	Reserved
[5]	**GPE_MFP5**	**PE.5 Pin Function Selection** The pin function depends on GPE_MFP5 and PE5_T1EX (ALT_MFP[25]) <table><tr><th>PE5_T1EX</th><th>GPE_MFP[5]</th><th>PE.5 function</th></tr><tr><td>0</td><td>0</td><td>GPIO</td></tr><tr><td>0</td><td>1</td><td>PWM5 (PWM)</td></tr><tr><td>1</td><td>1</td><td>T1EX (TMR1)</td></tr></table>
[4:2]	**Reserved**	Reserved
[1]	**GPE_MFP1**	**PE.1 Pin Function Selection** 1 = The PWM7 function is selected to the pin PE.1 0 = The GPIOE[1] is selected to the pin PE.1
[0]	**GPE_MFP0**	**PE.0 Pin Function Selection** 1 = The PWM6 function is selected to the pin PE.0 0 = The GPIOE[0] is selected to the pin PE.0

Alternative Multiple Function Pin Control Register (ALT_MFP)

Register	Offset	R/W	Description	Reset Value
ALT_MFP	GCR_BA+0x50	R/W	Alternative Multiple Function Pin Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				PB3_T3EX	PB2_T2EX	PE5_T1EX	PB15_T0EX
23	22	21	20	19	18	17	16
EBI_HB_EN							
15	14	13	12	11	10	9	8
Reserved	EBI_nWRH_EN	EBI_nWRL_EN	EBI_MCLK_EN	EBI_EN	PB12_CLKO	PA15_I2SMCLK	PC3_I2SDO
7	6	5	4	3	2	1	0
PC2_I2SDI	PC1_I2SBCLK	PC0_I2SLRCLK	PB11_PWM4	PB14_S31	PA7_S21	PB9_S11	PB10_S01

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[31:24]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[27]</td><td>PB3_T3EX</td><td>Bits GPB_MFP3 and EBI_nWRH_EN (ALT_MFP[14]) and EBI_EN (ALT_MFP[11]) and PB3_T3EX (ALT_MFP[27]) determine the PB.3 function.
<table>
<tr><th>EBI_nWRH_EN</th><th>EBI_EN</th><th>GPB_MFP[3]</th><th>PB3_T3EX</th><th>PB.3 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>CTS0 (UART0)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>nWRH (EBI write high byte enable)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>T3EX (TMR3)</td></tr>
</table></td></tr>
<tr><td>[26]</td><td>PB2_T2EX</td><td>Bits GPB_MFP2 and EBI_nWRL_EN (ALT_MFP[13]) and EBI_EN (ALT_MFP[11]) and PB2_T2EX (ALT_MFP[26]) determine the PB.2 function.
<table>
<tr><th>EBI_nWRL_EN</th><th>EBI_EN</th><th>GPB_MFP[2]</th><th>PB2_T2EX</th><th>PB.2 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>RTS0 (UART0)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>nWRL (EBI write low byte enable)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>T2EX (TMR2)</td></tr>
</table></td></tr>
<tr><td>[25]</td><td>PE5_T1EX</td><td>Bits GPE_MFP5 and PE5_T1EX (ALT_MFP[25]) determine the PE.5 function.
<table>
<tr><th>PE5_T1EX</th><th>GPE_MFP[5]</th><th>PE.5 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>0</td><td>1</td><td>PWM5 (PWM)</td></tr>
<tr><td>1</td><td>1</td><td>T1EX (TMR1)</td></tr>
</table>
</td></tr>
<tr><td>[24]</td><td>PB15_T0EX</td><td>
Bits GPB_MFP15 and PB15_T0EX (ALT_MFP[24]) determine the PB.15 function.
<table>
<tr><th>PB15_T0EX</th><th>GPB_MFP[15]</th><th>PB.15 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>/INT1</td></tr>
<tr><td>1</td><td>1</td><td>T0EX (TMR0)</td></tr>
</table>
</td></tr>
<tr><td>[23]</td><td>EBI_HB_EN[7]</td><td>
EBI_HB_EN is use to switch GPIO function to EBI address/data bus high byte (AD[15:8]), EBI_HB_EN, EBI_EN and corresponding GPx_MFP[y] determine the Px.y function.

Bits EBI_HB_EN[7], EBI_EN and GPA_MFP[14] determine the PA.14 function.
<table>
<tr><th>EBI_HB_EN[7]</th><th>EBI_EN</th><th>GPA_MFP[14]</th><th>PA.14 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>PWM2 (PWM)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>AD15 (EBI AD bus bit 15)</td></tr>
</table>
</td></tr>
<tr><td>[22]</td><td>EBI_HB_EN[6]</td><td>
Bits EBI_HB_EN[6], EBI_EN and GPA_MFP[13] determine the PA.13 function.
<table>
<tr><th>EBI_HB_EN[6]</th><th>EBI_EN</th><th>GPA_MFP[13]</th><th>PA.13 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>PWM1 (PWM)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>AD14 (EBI AD bus bit 14)</td></tr>
</table>
</td></tr>
<tr><td>[21]</td><td>EBI_HB_EN[5]</td><td>
Bits EBI_HB_EN[5], EBI_EN and GPA_MFP[12] determine the PA.12 function.
<table>
<tr><th>EBI_HB_EN[5]</th><th>EBI_EN</th><th>GPA_MFP[12]</th><th>PA.12 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>PWM0 (PWM)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>AD13 (EBI AD bus bit 13)</td></tr>
</table>
</td></tr>
<tr><td>[20]</td><td>EBI_HB_EN[4]</td><td>
Bits EBI_HB_EN[4], EBI_EN and GPA_MFP[1] determine the PA.1 function.
<table>
<tr><th>EBI_HB_EN[4]</th><th>EBI_EN</th><th>GPA_MFP[1]</th><th>PA.1 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>ADC1 (ADC)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>AD12 (EBI AD bus bit 12)</td></tr>
</table>
</td></tr>
<tr><td>[19]</td><td>EBI_HB_EN[3]</td><td>
Bits EBI_HB_EN[3], EBI_EN and GPA_MFP[2] determine the PA.2 function.
<table>
<tr><th>EBI_HB_EN[3]</th><th>EBI_EN</th><th>GPA_MFP[2]</th><th>PA.2 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>ADC2 (ADC)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>AD11 (EBI AD bus bit 11)</td></tr>
</table>
</td></tr>
<tr><td>[18]</td><td>EBI_HB_EN[2]</td><td>Bits EBI_HB_EN[2], EBI_EN and GPA_MFP[3] determine the PA.3 function.</td></tr>
</table>

<table>
<tr><td></td><td></td><td>
<table>
<tr><th>EBI_HB_EN[2]</th><th>EBI_EN</th><th>GPA_MFP[3]</th><th>PA.3 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>ADC3 (ADC)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>AD10 (EBI AD bus bit 10)</td></tr>
</table>
</td></tr>
<tr><td>[17]</td><td>EBI_HB_EN[1]</td><td>Bits EBI_HB_EN[1], EBI_EN and GPA_MFP[4] determine the PA.4 function.
<table>
<tr><th>EBI_HB_EN[1]</th><th>EBI_EN</th><th>GPA_MFP[4]</th><th>PA.4 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>ADC4 (ADC)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>AD9 (EBI AD bus bit 9)</td></tr>
</table>
</td></tr>
<tr><td>[16]</td><td>EBI_HB_EN[0]</td><td>Bits EBI_HB_EN[0], EBI_EN and GPA_MFP[5] determine the PA.5 function.
<table>
<tr><th>EBI_HB_EN[0]</th><th>EBI_EN</th><th>GPA_MFP[5]</th><th>PA.5 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>ADC5 (ADC)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>AD8 (EBI AD bus bit 8)</td></tr>
</table>
</td></tr>
<tr><td>[15]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[14]</td><td>EBI_nWRH_EN</td><td>Bits EBI_nWRH_EN, EBI_EN and GPB_MFP[3] determine the PB.3 function.
<table>
<tr><th>EBI_nWRH_EN</th><th>EBI_EN</th><th>GPB_MFP[3]</th><th>PB.3 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>CTS0 (UART0)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>nWRH (EBI write high byte enable)</td></tr>
</table>
</td></tr>
<tr><td>[13]</td><td>EBI_nWRL_EN</td><td>Bits EBI_nWRL_EN, EBI_EN and GPB_MFP[2] determine the PB.2 function.
<table>
<tr><th>EBI_nWRL_EN</th><th>EBI_EN</th><th>GPB_MFP[2]</th><th>PB.2 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>RTS0 (UART0)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>nWRL (EBI write low byte enable)</td></tr>
</table>
</td></tr>
<tr><td>[12]</td><td>EBI_MCLK_EN</td><td>Bits EBI_MCLK_EN, EBI_EN and GPC_MFP[8] determine the PC.8 function.
<table>
<tr><th>EBI_MCLK_EN</th><th>EBI_EN</th><th>GPC_MFP[8]</th><th>PC.8 function</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>SPISS10 (SPI1)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>MCLK (EBI Clock output)</td></tr>
</table>
</td></tr>
<tr><td>[11]</td><td>EBI_EN</td><td>EBI_EN is use to switch GPIO function to EBI function (AD[15:0], ALE, RE, WE, CS, MCLK), it need additional registers EBI_EN[7:0] and EBI_MCLK_EN for some GPIO to switch to EBI function(AD[15:8], MCLK)
<table>
<tr><th>EBI_EN</th><th>GPA_MFP[6]</th><th>PA.6 function</th></tr>
</table>
</td></tr>
</table>

0	0	GPIO
0	1	ADC5 (ADC)
1	1	AD7 (EBI AD bus bit 7)

EBI_EN	GPA_MFP[7]	PA.7 function
0	0	GPIO
0	1	ADC7 (ADC)
1	1	AD6 (EBI AD bus bit 6)

EBI_EN	GPC_MFP[7]	PC.7 function
0	0	GPIO
0	1	CPN0 (CMP)
1	1	AD5 (EBI AD bus bit 5)

EBI_EN	GPC_MFP[6]	PC.6 function
0	0	GPIO
0	1	CPP0 (CMP)
1	1	AD4 (EBI AD bus bit 4)

EBI_EN	GPC_MFP[15]	PC.15 function
0	0	GPIO
0	1	CPN1 (CMP)
1	1	AD3 (EBI AD bus bit 3)

EBI_EN	GPC_MFP[14]	PC.14 function
0	0	GPIO
0	1	CPP1 (CMP)
1	1	AD2 (EBI AD bus bit 2)

EBI_EN	GPB_MFP[13]	PB.13 function
0	0	GPIO
0	1	CPO1 (CMP)
1	1	AD1 (EBI AD bus bit 1)

EBI_EN	PB12_CLKO	GPB_MFP[12]	PB.12 function
0	0	0	GPIO
0	0	1	CPO0 (CMP)
0	1	1	CLKO (Clock Driver output)
1	1	1	AD0 (EBI AD bus bit 0)

EBI_EN	GPA_MFP[11]	PA.11 function
0	0	GPIO
0	1	SCL1 (I^2C)
1	1	nRD (EBI)

EBI_EN	GPA_MFP[10]	PA.10 function
0	0	GPIO
0	1	SDA1 (I^2C)
1	1	nWR (EBI)

EBI_EN	GPB_MFP[6]	PB.6 function
0	0	GPIO
0	1	RTS1 (UART1)
1	1	ALE (EBI)

EBI_EN	GPB_MFP[7]	PB.7 function
0	0	GPIO
0	1	CTS1 (UART1)
1	1	nCS (EBI)

[10] **PB12_CLKO**

Bits PB12_CLKO, GPB_MFP[12] and EBI_EN (ALT_MFP[11]) determine the PB.12 function.

EBI_EN	PB12_CLKO	GPB_MFP[12]	PB.12 function
0	0	0	GPIO
0	0	1	CPO0 (CMP)
0	1	1	CLKO (Clock Driver output)
1	1	1	AD0 (EBI AD bus bit 0)

[9] **PA15_I2SMCLK**

Bits PA15_I2SMCLK and GPA_MFP[15] determine the PA.15 function.

PA15_I2SMCLK	GPA_MFP[15]	PA.15 function
0	0	GPIO

<table>
<tr><td></td><td></td><td><table>
<tr><td>0</td><td>1</td><td>PWM3 (PWM)</td></tr>
<tr><td>1</td><td>1</td><td>I2SMCLK (I^{2}S)</td></tr>
</table></td></tr>
<tr><td>[8]</td><td>PC3_I2SDO</td><td>Bits PC3_I2SDO and GPC_MFP[3] determine the PC.3 function.<table>
<tr><th>PC3_I2SDO</th><th>GPC_MFP[3]</th><th>PC.3 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MOSI00 (SPI0)</td></tr>
<tr><td>1</td><td>1</td><td>I2SDO (I^{2}S)</td></tr>
</table></td></tr>
<tr><td>[7]</td><td>PC2_I2SDI</td><td>Bits PC2_I2SDI and GPC_MFP[2] determine the PC.2 function.<table>
<tr><th>PC2_I2SDI</th><th>GPC_MFP[2]</th><th>PC.2 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MISO00 (SPI0)</td></tr>
<tr><td>1</td><td>1</td><td>I2SDI (I^{2}S)</td></tr>
</table></td></tr>
<tr><td>[6]</td><td>PC1_I2SBCLK</td><td>Bits PC1_I2SBCLK and GPC_MFP[1] determine the PC.1 function.<table>
<tr><th>PC1_I2SBCLK</th><th>GPC_MFP[1]</th><th>PC.1 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>SPICLK0 (SPI0)</td></tr>
<tr><td>1</td><td>1</td><td>I2SBCLK (I^{2}S)</td></tr>
</table></td></tr>
<tr><td>[5]</td><td>PC0_I2SLRCLK</td><td>Bits PC0_I2SLRCLK and GPC_MFP[0] determine the PC.0 function.<table>
<tr><th>PC0_I2SLRCLK</th><th>GPC_MFP[0]</th><th>PC.0 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>SPISS00 (SPI0)</td></tr>
<tr><td>1</td><td>1</td><td>I2SLRCLK (I^{2}S)</td></tr>
</table></td></tr>
<tr><td>[4]</td><td>PB11_PWM4</td><td>Bits PB11_PWM4 and GPB_MFP[11] determine the PB.11 function.<table>
<tr><th>PB11_PWM4</th><th>GPB_MFP[11]</th><th>PB.11 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>TM3</td></tr>
<tr><td>1</td><td>1</td><td>PWM4 (PWM)</td></tr>
</table></td></tr>
<tr><td>[3]</td><td>PB14_S31</td><td>Bits PB14_S31 and GPB_MFP[14] determine the PB.14 function.<table>
<tr><th>PB14_S31</th><th>GPB_MFP[14]</th><th>PB.14 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>/INT0</td></tr>
<tr><td>1</td><td>1</td><td>SPISS31 (SPI3)</td></tr>
</table></td></tr>
<tr><td>[2]</td><td>PA7_S21</td><td>Bits PA7_S21, GPA_MFP[7] and EBI_EN (ALT_MFP[11]).determine the PA.7 function.<table>
<tr><th>EBI_EN</th><th>PA7_S21</th><th>GPA_MFP[7]</th><th>PA.7 function</th></tr>
</table></td></tr>
</table>

<table>
<tr><td></td><td></td><td><table>
<tr><td>0</td><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>ADC7 (ADC)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>SPISS21 (SPI2)</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>AD6 (EBI AD bus bit 6)</td></tr>
</table></td></tr>
<tr><td>[1]</td><td>PB9_S11</td><td>Bits PB9_S11 and GPB_MFP[9] determine the PB.9 function.<table>
<tr><th>PB9_S11</th><th>GPB_MFP[9]</th><th>PB.9 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>TM1</td></tr>
<tr><td>1</td><td>1</td><td>SPISS11 (SPI1)</td></tr>
</table></td></tr>
<tr><td>[0]</td><td>PB10_S01</td><td>Bits PB10_S01 and GPB_MFP[10] determine the PB.10 function.<table>
<tr><th>PB10_S01</th><th>GPB_MFP[10]</th><th>PB.10 function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>TM2</td></tr>
<tr><td>1</td><td>1</td><td>SPISS01 (SPI0)</td></tr>
</table></td></tr>
</table>

Register Write-Protection Control Register (REGWRPROT)

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 REGWRPROT address at 0x5000_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 0x5000_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 “0x5000_0100” to enable register protection.

This register is write for disable/enable register protection and read for the REGPROTDIS status

Register	**Offset**	**R/W**	**Description**	**Reset Value**
REGWRPROT	GCR_BA+0x100	R/W	Register Write-Protection 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**
REGWRPROT[7:1]							**REGWRPROT [0]** **REGPROTDIS**

Bits	**Descriptions**	
[31:16]	**Reserved**	Reserved
[7:0]	**REGWRPROT**	**Register Write-Protection Code** (Write Only) Some registers have write-protection function. Writing these registers have to disable the protected function by writing the sequence value “59h”, “16h”, “88h” to this field. After this sequence is completed, the REGPROTDIS bit will be set to 1 and write-protection registers can be normal write.
[0]	**REGPROTDIS**	**Register Write-Protection Disable index** (Read only) 1 = Write-protection is disabled for writing protected registers 0 = Write-protection is enabled for writing protected registers. Any write to the protected register is ignored. The Protected registers are: **IPRSTC1**: address 0x5000_0008

		BODCR: address 0x5000_0018 **PORCR**: address 0x5000_0024 **PWRCON**: address 0x5000_0200 (bit[6] is not protected for power wake-up interrupt clear) **APBCLK bit[0]**: address 0x5000_0208 (bit[0] is watchdog clock enable) **CLKSEL0**: address 0x5000_0210 (for HCLK and CPU STCLK clock source select) **CLKSEL1 bit[1:0]**: address 0x5000_0214 (for watchdog clock source select) **ISPCON**: address 0x5000_C000 (Flash ISP Control register) **WTCR**: address 0x4000_4000 **FATCON**: address 0x5000_C018

5.2.6 System Timer (SysTick)

The Cortex-M0 includes an integrated system timer, SysTick. SysTick 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_CVR) to zero, and reload (wrap) to the value in the SysTick Reload Value Register (SYST_RVR) on the next clock cycle, 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_CVR 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_RVR value rather than an arbitrary value when it is enabled.

If the SYST_RVR 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 documents “ARM® Cortex™-M0 Technical Reference Manual” and “ARM® v6-M Architecture Reference Manual”.

5.2.6.1 System Timer Control Register Map

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

Register	**Offset**	**R/W**	**Description**	**Reset Value**
SCS_BA = 0xE000_E000				
SYST_CSR	SCS_BA+0x10	R/W	SysTick Control and Status Register	0x0000_0000
SYST_RVR	SCS_BA+0x14	R/W	SysTick Reload value Register	0xXXXX_XXXX
SYST_CVR	SCS_BA+0x18	R/W	SysTick Current value Register	0xXXXX_XXXX

5.2.6.2 System Timer Control Register Description

SysTick Control and Status (SYST_CSR)

Register	Offset	R/W	Description	Reset Value
SYST_CSR	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	Descriptions	
[31:17]	**Reserved**	Reserved
[16]	**COUNTFLAG**	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**	1 = Core clock used for SysTick. 0 = Clock source is (optional) external reference clock
[1]	**TICKINT**	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 = 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.
[0]	**ENABLE**	1 = The counter will operate in a multi-shot manner 0 = The counter is disabled

SysTick Reload Value Register (SYST_RVR)

Register	Offset	R/W	Description	Reset Value
SYST_RVR	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[23:16]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RELOAD[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RELOAD[7:0]							

Bits	Descriptions	
[31:24]	**Reserved**	Reserved
[23:0]	**RELOAD**	Value to load into the Current Value register when the counter reaches 0.

SysTick Current Value Register (SYST_CVR)

Register	Offset	R/W	Description	Reset Value
SYST_CVR	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 [23:16]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CURRENT [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CURRENT[7:0]							

Bits	Descriptions	
[31:24]	**Reserved**	Reserved
[23:0]	**CURRENT**	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.

5.2.7 Nested Vectored Interrupt Controller (NVIC)

Cortex-M0 provides an interrupt controller as an integral part of the exception mode, named as "Nested Vectored Interrupt Controller (NVIC)". It is closely coupled to the processor kernel and provides following features:

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

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

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

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

For more detailed information, please refer to the documents "ARM® Cortex™-M0 Technical Reference Manual" and "ARM® v6-M Architecture Reference Manual".

5.2.7.1 Exception Model and System Interrupt Map

Table 5-2 lists the exception model supported by NuMicro™ NUC100 Series. Software can set four levels of priority on some of these exceptions as well as on all interrupts. The highest user-configurable priority is denoted as “0” and the lowest priority is denoted as “3”. The default priority of all the user-configurable interrupts is “0”. Note that priority “0” is treated as the fourth priority on the system, after three system exceptions “Reset”, “NMI” and “Hard Fault”.

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

Table 5-2 Exception Model

Vector Number	**Interrupt Number (Bit in Interrupt Registers)**	**Interrupt Name**	**Source IP**	**Interrupt description**
0 ~ 15	-	**-**	-	System exceptions
16	0	**BOD_OUT**	Brown-Out	Brown-Out low voltage detected interrupt
17	1	**WDT_INT**	WDT	Watchdog Timer interrupt
18	2	**EINT0**	GPIO	External signal interrupt from PB.14 pin
19	3	**EINT1**	GPIO	External signal interrupt from PB.15 pin
20	4	**GPAB_INT**	GPIO	External signal interrupt from PA[15:0]/PB[13:0]
21	5	**GPCDE_INT**	GPIO	External interrupt from PC[15:0]/PD[15:0]/PE[15:0]
22	6	**PWMA_INT**	PWM0~3	PWM0, PWM1, PWM2 and PWM3 interrupt
23	7	**PWMB_INT**	PWM4~7	PWM4, PWM5, PWM6 and PWM7 interrupt
24	8	**TMR0_INT**	TMR0	Timer 0 interrupt
25	9	**TMR1_INT**	TMR1	Timer 1 interrupt

26	10	**TMR2_INT**	TMR2	Timer 2 interrupt
27	11	**TMR3_INT**	TMR3	Timer 3 interrupt
28	12	**UART02_INT**	UART0/2	UART0 and UART2 interrupt
29	13	**UART1_INT**	UART1	UART1 interrupt
30	14	**SPI0_INT**	SPI0	SPI0 interrupt
31	15	**SPI1_INT**	SPI1	SPI1 interrupt
32	16	**SPI2_INT**	SPI2	SPI2 interrupt
33	17	**SPI3_INT**	SPI3	SPI3 interrupt
34	18	**I2C0_INT**	I^2C0	I^2C0 interrupt
35	19	**I2C1_INT**	I^2C1	I^2C1 interrupt
36	20	**CAN0_INT**	CAN0	CAN0 interrupt
37	21	**Reserved**	Reserved	Reserved
38	22	**Reserved**	Reserved	Reserved
39	23	**USB_INT**	USBD	USB 2.0 FS Device interrupt
40	24	**PS2_INT**	PS/2	PS/2 interrupt
41	25	**ACMP_INT**	ACMP	Analog Comparator-0 or Comaprator-1 interrupt
42	26	**PDMA_INT**	PDMA	PDMA interrupt
43	27	**I2S_INT**	I^2S	I^2S interrupt
44	28	**PWRWU_INT**	CLKC	Clock controller interrupt for chip wake-up from power down state
45	29	**ADC_INT**	ADC	ADC interrupt
46	30	**Reserved**	Reserved	Reserved
47	31	**RTC_INT**	RTC	Real time clock interrupt

Table 5-3 System Interrupt Map

5.2.7.2 Vector Table

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. For ARMv6-M, the vector table base address is fixed at 0x00000000. The vector table contains the initialization value for the stack pointer on reset, and the entry point addresses for all exception handlers. The vector number on previous page defines the order of entries in the vector table associated with exception handler entry as illustrated in previous section.

Vector Table Word Offset	Description
0	SP_main – The Main stack pointer
Vector Number	Exception Entry Pointer using that Vector Number

Table 5-4 Vector Table Format

5.2.7.3 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.

5.2.7.4 NVIC Control Registers

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

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

IRQ0 ~ IRQ31 Set-Enable Control Register (NVIC_ISER)

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

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

Bits	Descriptions	
[31:0]	**SETENA**	Enable one or more interrupts within a group of 32. Each bit represents an interrupt number from IRQ0 ~ IRQ31 (Vector number from 16 ~ 47). Writing 1 will enable the associated interrupt. Writing 0 has no effect. The register reads back with the current enable state.

IRQ0 ~ IRQ31 Clear-Enable Control Register (NVIC_ICER)

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

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

Bits	Descriptions	
[31:0]	**CLRENA**	Disable one or more interrupts within a group of 32. Each bit represents an interrupt number from IRQ0 ~ IRQ31 (Vector number from 16 ~ 47). Writing 1 will disable the associated interrupt. Writing 0 has no effect. The register reads back with the current enable state.

IRQ0 ~ IRQ31 Set-Pending Control Register (NVIC_ISPR)

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

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

Bits	Descriptions	
[31:0]	**SETPEND**	Writing 1 to a bit to set pending state of the associated interrupt under software control. Each bit represents an interrupt number from IRQ0 ~ IRQ31 (Vector number from 16 ~ 47). Writing 0 has no effect. The register reads back with the current pending state.

IRQ0 ~ IRQ31 Clear-Pending Control Register (NVIC_ICPR)

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

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

Bits	Descriptions	
[31:0]	**CLRPEND**	Writing 1 to a bit to remove the pending state of associated interrupt under software control. Each bit represents an interrupt number from IRQ0 ~ IRQ31 (Vector number from 16 ~ 47). Writing 0 has no effect. The register reads back with the current pending state.

IRQ0 ~ IRQ3 Interrupt Priority Register (NVIC_IPR0)

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

31	30	29	28	27	26	25	24
PRI_3		**Reserved**					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PRI_2		**Reserved**					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PRI_1		**Reserved**					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRI_0		**Reserved**					

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

IRQ4 ~ IRQ7 Interrupt Priority Register (NVIC_IPR1)

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

31	30	29	28	27	26	25	24
PRI_7		**Reserved**					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PRI_6		**Reserved**					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PRI_5		**Reserved**					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRI_4		**Reserved**					

Bits	Descriptions	
[31:30]	**PRI_7**	**Priority of IRQ7** “0” denotes the highest priority and “3” denotes lowest priority
[29:24]	**Reserved**	Reserved
[23:22]	**PRI_6**	**Priority of IRQ6** “0” denotes the highest priority and “3” denotes lowest priority
[21:16]	**Reserved**	Reserved
[15:14]	**PRI_5**	**Priority of IRQ5** “0” denotes the highest priority and “3” denotes lowest priority
[13:8]	**Reserved**	Reserved
[7:6]	**PRI_4**	**Priority of IRQ4** “0” denotes the highest priority and “3” denotes lowest priority
[5:0]	**Reserved**	Reserved

IRQ8 ~ IRQ11 Interrupt Priority Register (NVIC_IPR2)

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

31	30	29	28	27	26	25	24
PRI_11		Reserved					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PRI_10		Reserved					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PRI_9		Reserved					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRI_8		Reserved					

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

IRQ12 ~ IRQ15 Interrupt Priority Register (NVIC_IPR3)

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

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**
PRI_13		**Reserved**					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRI_12		**Reserved**					

Bits	Descriptions	
[31:30]	**PRI_15**	**Priority of IRQ15** “0” denotes the highest priority and “3” denotes lowest priority
[29:24]	**Reserved**	Reserved
[23:22]	**PRI_14**	**Priority of IRQ14** “0” denotes the highest priority and “3” denotes lowest priority
[21:16]	**Reserved**	Reserved
[15:14]	**PRI_13**	**Priority of IRQ13** “0” denotes the highest priority and “3” denotes lowest priority
[13:8]	**Reserved**	Reserved
[7:6]	**PRI_12**	**Priority of IRQ12** “0” denotes the highest priority and “3” denotes lowest priority
[5:0]	**Reserved**	Reserved

IRQ16 ~ IRQ19 Interrupt Priority Register (NVIC_IPR4)

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

31	30	29	28	27	26	25	24
PRI_19		**Reserved**					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PRI_18		**Reserved**					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PRI_17		**Reserved**					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRI_16		**Reserved**					

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

IRQ20 ~ IRQ23 Interrupt Priority Register (NVIC_IPR5)

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

31	30	29	28	27	26	25	24
PRI_23		**Reserved**					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PRI_22		**Reserved**					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PRI_21		**Reserved**					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRI_20		**Reserved**					

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

IRQ24 ~ IRQ27 Interrupt Priority Register (NVIC_IPR6)

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

31	30	29	28	27	26	25	24
PRI_27		**Reserved**					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PRI_26		**Reserved**					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PRI_25		**Reserved**					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRI_24		**Reserved**					

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

IRQ28 ~ IRQ31 Interrupt Priority Register (NVIC_IPR7)

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

31	30	29	28	27	26	25	24
PRI_31		**Reserved**					
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PRI_30		**Reserved**					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PRI_29		**Reserved**					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRI_28		**Reserved**					

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

5.2.7.5 *Interrupt Source Control Registers*

Besides the interrupt control registers associated with the NVIC, NuMicro™ NUC100 Series also implement some specific control registers to facilitate the interrupt functions, including "interrupt source identification", "NMI source selection" and "interrupt test mode". They are described as below.

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

Register	Offset	R/W	Description	Reset Value
INT_BA = 0x5000_0300				
IRQ0_SRC	INT_BA+0x00	R	IRQ0 (BOD) interrupt source identity	0xXXXX_XXXX
IRQ1_SRC	INT_BA+0x04	R	IRQ1 (WDT) interrupt source identity	0xXXXX_XXXX
IRQ2_SRC	INT_BA+0x08	R	IRQ2 (EINT0) interrupt source identity	0xXXXX_XXXX
IRQ3_SRC	INT_BA+0x0C	R	IRQ3 (EINT1) interrupt source identity	0xXXXX_XXXX
IRQ4_SRC	INT_BA+0x10	R	IRQ4 (GPA/B) interrupt source identity	0xXXXX_XXXX
IRQ5_SRC	INT_BA+0x14	R	IRQ5 (GPC/D/E) interrupt source identity	0xXXXX_XXXX
IRQ6_SRC	INT_BA+0x18	R	IRQ6 (PWMA) interrupt source identity	0xXXXX_XXXX
IRQ7_SRC	INT_BA+0x1C	R	IRQ7 (PWMB) interrupt source identity	0xXXXX_XXXX
IRQ8_SRC	INT_BA+0x20	R	IRQ8 (TMR0) interrupt source identity	0xXXXX_XXXX
IRQ9_SRC	INT_BA+0x24	R	IRQ9 (TMR1) interrupt source identity	0xXXXX_XXXX
IRQ10_SRC	INT_BA+0x28	R	IRQ10 (TMR2) interrupt source identity	0xXXXX_XXXX
IRQ11_SRC	INT_BA+0x2C	R	IRQ11 (TMR3) interrupt source identity	0xXXXX_XXXX
IRQ12_SRC	INT_BA+0x30	R	IRQ12 (URT0) interrupt source identity	0xXXXX_XXXX
IRQ13_SRC	INT_BA+0x34	R	IRQ13 (URT1) interrupt source identity	0xXXXX_XXXX
IRQ14_SRC	INT_BA+0x38	R	IRQ14 (SPI0) interrupt source identity	0xXXXX_XXXX
IRQ15_SRC	INT_BA+0x3C	R	IRQ15 (SPI1) interrupt source identity	0xXXXX_XXXX
IRQ16_SRC	INT_BA+0x40	R	IRQ16 (SPI2) interrupt source identity	0xXXXX_XXXX
IRQ17_SRC	INT_BA+0x44	R	IRQ17 (SPI3)) interrupt source identity	0xXXXX_XXXX
IRQ18_SRC	INT_BA+0x48	R	IRQ18 (I^2C0) interrupt source identity	0xXXXX_XXXX
IRQ19_SRC	INT_BA+0x4C	R	IRQ19 (I^2C1) interrupt source identity	0xXXXX_XXXX
IRQ20_SRC	INT_BA+0x50	R	IRQ20 (CAN0) interrupt source identity	0xXXXX_XXXX
IRQ21_SRC	INT_BA+0x54	R	IRQ21 (Reserved) interrupt source identity	0xXXXX_XXXX

IRQ22_SRC	INT_BA+0x58	R	IRQ22 (Reserved) interrupt source identity	0xXXXX_XXXX
IRQ23_SRC	INT_BA+0x5C	R	IRQ23 (USBD) interrupt source identity	0xXXXX_XXXX
IRQ24_SRC	INT_BA+0x60	R	IRQ24 (PS/2) interrupt source identity	0xXXXX_XXXX
IRQ25_SRC	INT_BA+0x64	R	IRQ25 (ACMP) interrupt source identity	0xXXXX_XXXX
IRQ26_SRC	INT_BA+0x68	R	IRQ26 (PDMA) interrupt source identity	0xXXXX_XXXX
IRQ27_SRC	INT_BA+0x6C	R	IRQ27 (I^2S) interrupt source identity	0xXXXX_XXXX
IRQ28_SRC	INT_BA+0x70	R	IRQ28 (PWRWU) interrupt source identity	0xXXXX_XXXX
IRQ29_SRC	INT_BA+0x74	R	IRQ29 (ADC) interrupt source identity	0xXXXX_XXXX
IRQ30_SRC	INT_BA+0x78	R	IRQ30 (Reserved) interrupt source identity	0xXXXX_XXXX
IRQ31_SRC	INT_BA+0x7C	R	IRQ31 (RTC) interrupt source identity	0xXXXX_XXXX
NMI_SEL	INT_BA+0x80	R/W	NMI source interrupt select control register	0x0000_0000
MCU_IRQ	INT_BA+0x84	R/W	MCU IRQ Number identity register	0x0000_0000

Interrupt Source Identity Register (IRQn_SRC)

Register	Offset	R/W	Description	Reset Value
IRQn_SRC	INT_BA+0x00 …….. INT_BA+0x7C	R	IRQ0 (BOD) interrupt source identity : IRQ31 (RTC) interrupt source identity	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**
Reserved				**INT_SRC[3]**	**INT_SRC[2:0]**		

Bits	Address	INT-Num	Descriptions
[2:0]	INT_BA+0x00	0	Bit2: 0 Bit1: 0 Bit0: BOD_INT
[2:0]	INT_BA+0x04	1	Bit2: 0 Bit1: 0 Bit0: WDT_INT
[2:0]	INT_BA+0x08	2	Bit2: 0 Bit1: 0 Bit0: EINT0 – external interrupt 0 from PB.14
[2:0]	INT_BA+0x0C	3	Bit2: 0 Bit1: 0 Bit0: EINT1 – external interrupt 1 from PB.15
[2:0]	INT_BA+0x10	4	Bit2: 0 Bit1: GPB_INT Bit0: GPA_INT
[2:0]	INT_BA+0x14	5	Bit2: GPE_INT Bit1: GPD_INT Bit0: GPC_INT
[3:0]	INT_BA+0x18	6	Bit3: PWM3_INT Bit2: PWM2_INT

			Bit1: PWM1_INT Bit0: PWM0_INT
[3:0]	INT_BA+0x1C	7	Bit3: PWM7_INT Bit2: PWM6_INT Bit1: PWM5_INT Bit0: PWM4_INT
[2:0]	INT_BA+0x20	8	Bit2: 0 Bit1: 0 Bit0: TMR0_INT
[2:0]	INT_BA+0x24	9	Bit2: 0 Bit1: 0 Bit0: TMR1_INT
[2:0]	INT_BA+0x28	10	Bit2: 0 Bit1: 0 Bit0: TMR2_INT
[2:0]	INT_BA+0x2C	11	Bit2: 0 Bit1: 0 Bit0: TMR3_INT
[2:0]	INT_BA+0x30	12	Bit2: 0 Bit1: 0 Bit0: URT0_INT
[2:0]	INT_BA+0x34	13	Bit2: 0 Bit1: 0 Bit0: URT1_INT
[2:0]	INT_BA+0x38	14	Bit2: 0 Bit1: 0 Bit0: SPI0_INT
[2:0]	INT_BA+0x3C	15	Bit2: 0 Bit1: 0 Bit0: SPI1_INT
[2:0]	INT_BA+0x40	16	Bit2: 0 Bit1: 0 Bit0: SPI2_INT
[2:0]	INT_BA+0x44	17	Bit2: 0 Bit1: 0 Bit0: SPI3_INT
[2:0]	INT_BA+0x48	18	Bit2: 0 Bit1: 0 Bit0: I2C0_INT

[2:0]	INT_BA+0x4C	19	Bit2: 0 Bit1: 0 Bit0: I2C1_INT
[2:0]	INT_BA+0x50	20	Bit2: 0 Bit1: 0 Bit0: CAN0_INT
[2:0]	INT_BA+0x54	21	Reserved
[2:0]	INT_BA+0x58	22	Reserved
[2:0]	INT_BA+0x5C	23	Bit2: 0 Bit1: 0 Bit0: USB_INT
[2:0]	INT_BA+0x60	24	Bit2: 0 Bit1: 0 Bit0: PS2_INT
[2:0]	INT_BA+0x64	25	Bit2: 0 Bit1: 0 Bit0: ACMP_INT
[2:0]	INT_BA+0x68	26	Bit2: 0 Bit1: 0 Bit0: PDMA_INT
[2:0]	INT_BA+0x6C	27	Bit2: 0 Bit1: 0 Bit0: I2S_INT
[2:0]	INT_BA+0x70	28	Bit2: 0 Bit1: 0 Bit0: PWRWU_INT
[2:0]	INT_BA+0x74	29	Bit2: 0 Bit1: 0 Bit0: ADC_INT
[2:0]	INT_BA+0x78	30	Reserved
[2:0]	INT_BA+0x7C	31	Bit2: 0 Bit1: 0 Bit0: RTC_INT

NMI Interrupt Source Select Control Register (NMI_SEL)

Register	Offset	R/W	Description	Reset Value
NMI_SEL	INT_BA+0x80	R/W	NMI source interrupt select 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							**NMI_EN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			**NMI_SEL[4:0]**				

Bits	Descriptions	
[31:8]	**Reserved**	Reserved
[8]	**NMI_EN**	NMI interrupt enable 1 = Enable NMI interrupt 0 = Disable NMI interrupt
[7:5]	**Reserved**	Reserved
[4:0]	**NMI_SEL**	NMI interrupt source select The NMI interrupt to Cortex-M0 can be selected from one of the peripheral interrupt by setting NMI_SEL.

MCU Interrupt Request Source Register (MCU_IRQ)

Register	Offset	R/W	Description	Reset Value
MCU_IRQ	INT_BA+0x84	R/W	MCU Interrupt Request Source Register	0x0000_0000

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

Bits	Descriptions	
[31:0]	**MCU_IRQ**	**MCU IRQ Source Register** The MCU_IRQ collects all the interrupts from the peripherals and generates the synchronous interrupt to Cortex-M0. There are two modes to generate interrupt to Cortex-M0, the normal mode and test mode. The MCU_IRQ collects all interrupts from each peripheral and synchronizes them then interrupts the Cortex-M0. When the MCU_IRQ[n] is 0: Set MCU_IRQ[n] 1 will generate an interrupt to Cortex_M0 NVIC[n]. When the MCU_IRQ[n] is 1 (mean an interrupt is assert), set 1 to the MCU_IRQ[n] will clear the interrupt and set MCU_IRQ[n] 0 : no any effect

5.2.8 System Control Register

Cortex-M0 status and operating mode control are managed by System Control Registers. Including CPUID, Cortex-M0 interrupt priority and Cortex-M0power management can be controlled through these system control register

For more detailed information, please refer to the documents "ARM® Cortex™-M0 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
SCS_BA = 0xE000_E000				
CPUID	SCS_BA+0xD00	R	CPUID Register	0x410C_C200
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
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

CPUID Register (CPUID)

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

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

Bits	Descriptions	
[31:24]	**IMPLEMENTER**	Implementer code assigned by ARM. (ARM = 0x41)
[23:20]	**Reserved**	Reserved
[19:16]	**PART**	Reads as 0xC for ARMv6-M parts
[15:4]	**PARTNO**	Reads as 0xC20.
[3:0]	**REVISION**	Reads as 0x0

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**	**PENDSVCLR**	**PENDSTSET**	**PENDSTCLR**	**Reserved**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ISRPREEMPT	**ISRPENDING**	**Reserved**				**VECTPENDING[5:4]**	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
VECTPENDING[3:0]				**Reserved**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**VECTACTIVE[5:0]**					

Bits	Descriptions	
[31]	**NMIPENDSET**	NMI set-pending bit Write: 0 = no effect 1 = changes NMI exception state to pending. Read: 0 = NMI exception is not pending 1 = NMI exception is pending. 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: 0 = no effect 1 = changes PendSV exception state to pending. Read: 0 = PendSV exception is not pending 1 = PendSV exception is pending. Writing 1 to this bit is the only way to set the PendSV exception state to pending.
[27]	**PENDSVCLR**	PendSV clear-pending bit. Write: 0 = no effect

		1 = removes the pending state from the PendSV exception. This is a write only bit. When you want to clear PENDSV bit, you must "write 0 to PENDSVSET and write 1 to PENDSVCLR" at the same time.
[26]	**PENDSTSET**	SysTick exception set-pending bit. Write: 0 = no effect 1 = changes SysTick exception state to pending. Read: 0 = SysTick exception is not pending 1 = SysTick exception is pending.
[25]	**PENDSTCLR**	SysTick exception clear-pending bit. Write: 0 = no effect 1 = removes the pending state from the SysTick exception. This is a write only bit. When you want to clear PENDST bit, you must "write 0 to PENDSTSET and write 1 to PENDSTCLR" at the same time.
[24]	**Reserved**	Reserved
[23]	**ISRPREEMPT**	If set, a pending exception will be serviced on exit from the debug halt state. This is a read only bit.
[22]	**ISRPENDING**	Interrupt pending flag, excluding NMI and Faults: 0 = interrupt not pending 1 = interrupt pending. This is a read only bit.
[21:18]	**Reserved**	Reserved
[17:12]	**VECTPENDING**	Indicates 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.
[11:6]	**Reserved**	Reserved
[5:0]	**VECTACTIVE**	Contains the active exception number 0 = Thread mode Nonzero = 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[15:8]							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
VECTORKEY[7:0]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**SYSRESETREQ**	**VECTCLKACTIVE**	**Reserved**

Bits	Descriptions	
[31:16]	**VECTORKEY**	When write this register, this field should be 0x05FA, otherwise the write action will be unpredictable.
[15:3]	**Reserved**	Reserved
[2]	**SYSRESETREQ**	Writing this bit 1 will cause a reset signal to be asserted to the chip to indicate a reset is requested. The bit is a write only bit and self-clears as part of the reset sequence.
[1]	**VECTCLRACTIVE**	Set this bit to 1 will clears all active state information for fixed and configurable exceptions. The bit is a write only bit 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

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	Descriptions	
[31:5]	**Reserved**	Reserved
[4]	**SEVONPEND**	Send Event on Pending bit: 0 = only enabled interrupts or events can wakeup the processor, disabled interrupts are excluded 1 = enabled events and all interrupts, including disabled interrupts, can wakeup 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**	Controls whether the processor uses sleep or deep sleep as its low power mode: 0 = sleep 1 = deep sleep
[1]	**SLEEPONEXIT**	Indicates sleep-on-exit when returning from Handler mode to Thread mode: 0 = do not sleep when returning to Thread mode. 1 = enter sleep, or deep sleep, on return from 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 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	Descriptions	
[31:30]	**PRI_11**	**Priority of system handler 11 – SVCall** “0” denotes the highest priority and “3” denotes 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	Descriptions	
[31:30]	**PRI_15**	**Priority of system handler 15 – SysTick** “0” denotes the highest priority and “3” denotes lowest priority
[29:24]	**Reserved**	Reserved
[23:22]	**PRI_14**	**Priority of system handler 14 – PendSV** “0” denotes the highest priority and “3” denotes lowest priority
[21:0]	**Reserved**	Reserved

5.3 Clock Controller

5.3.1 Overview

The clock controller generates the 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 (PWR_DOWN_EN) and Cortex-M0 core executes the WFI instruction. After that, chip enter power down mode and wait for wake-up interrupt source triggered to leave power down mode. In the power down mode, the clock controller turns off the external 4~24 MHz high speed crystal and internal 22.1184 MHz high speed oscillator to reduce the overall system power consumption.

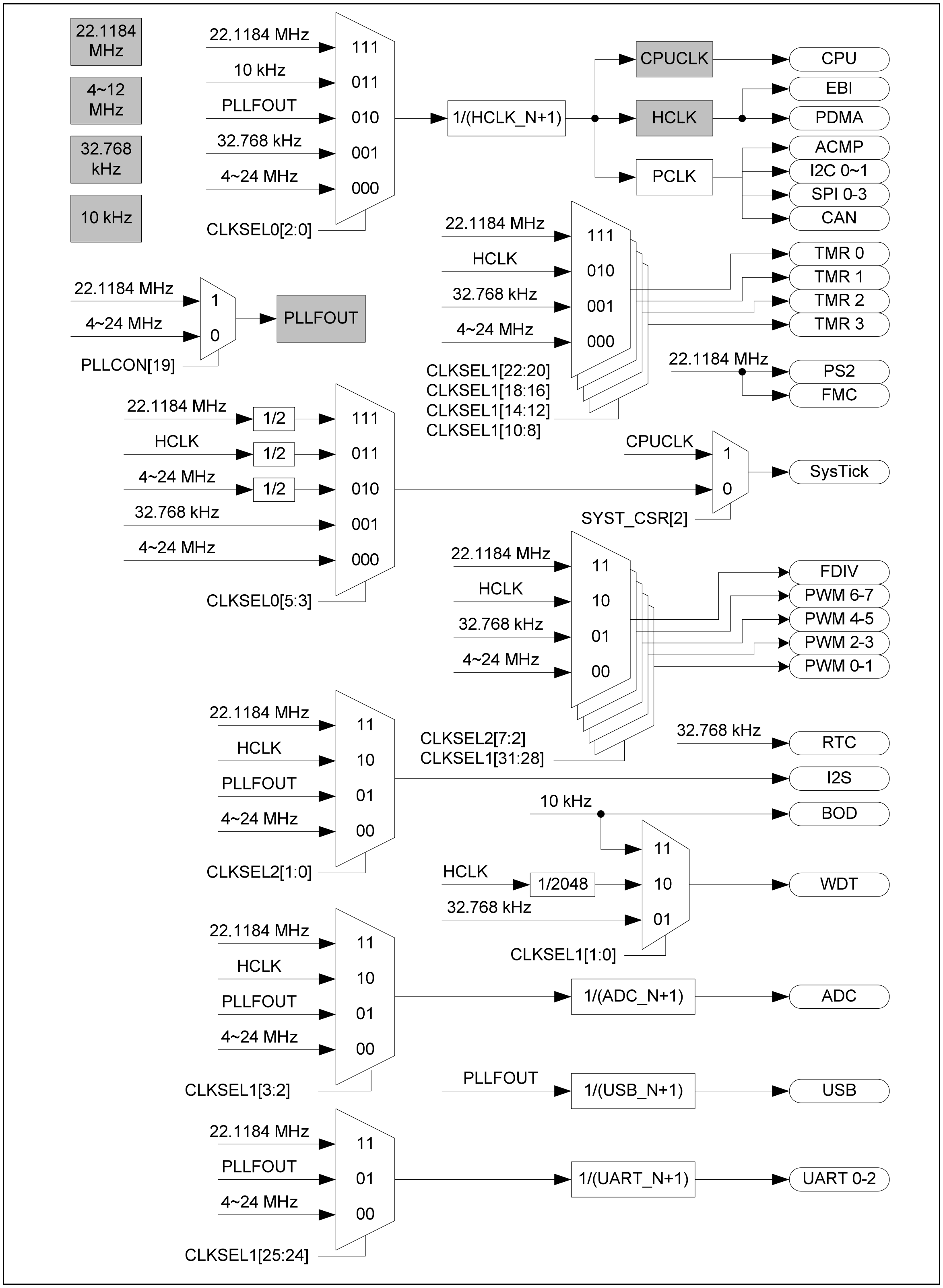


Figure 5-4 Clock generator global view diagram

5.3.2 Clock Generator

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

- One external 32.768 kHz low speed crystal
- One external 4~24 MHz high speed crystal
- One programmable PLL FOUT(PLL source consists of external 4~24 MHz high speed crystal and internal 22.1184 MHz high speed oscillator)
- One internal 22.1184 MHz high speed oscillator
- One internal 10 kHz low speed oscillator

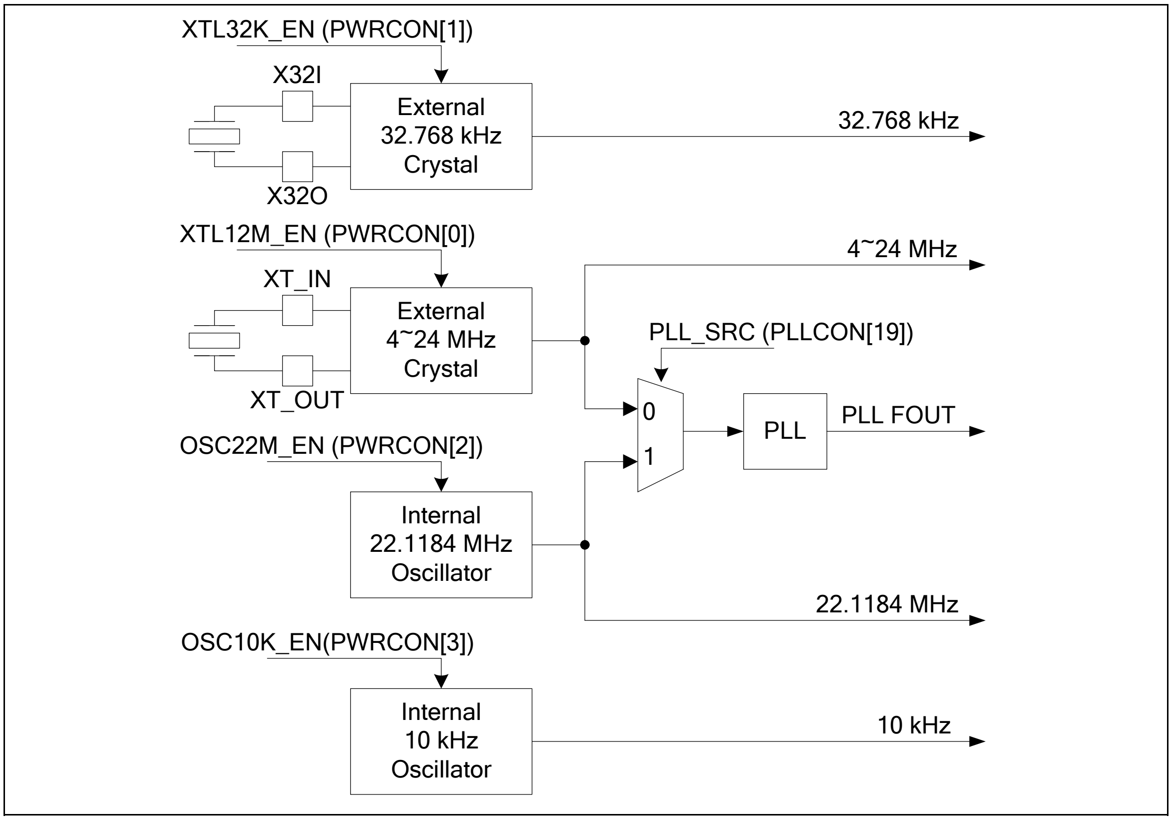


Figure 5-5 Clock generator block diagram

5.3.3 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 HCLK_S (CLKSEL0[2:0]). The block diagram is showed in Figure 5-6.

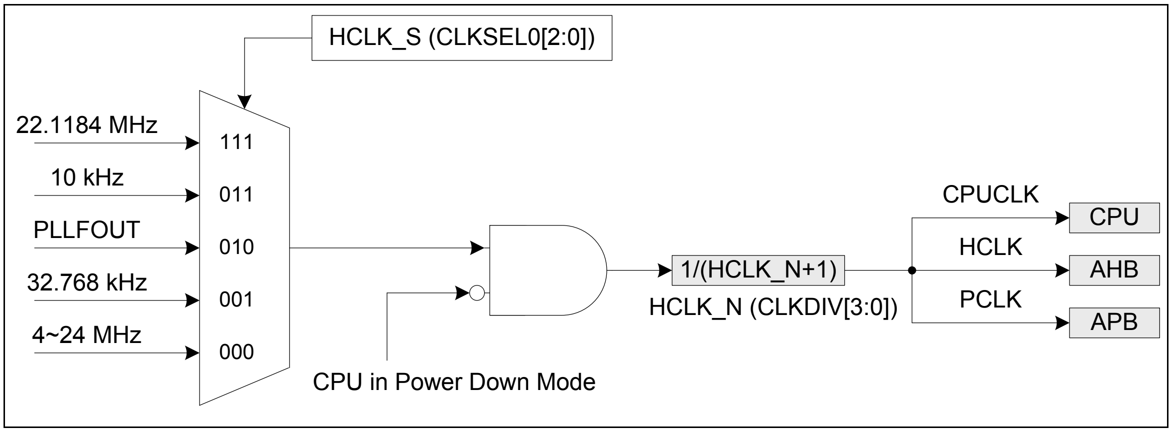


Figure 5-6 System Clock Block Diagram

The clock source of SysTick in Cortex-M0 core can use CPU clock or external clock (SYST_CSR[2]). If using external clock, the SysTick clock (STCLK) has 5 clock sources. The clock source switch depends on the setting of the register STCLK_S (CLKSEL0[5:3]). The block diagram is showed in Figure 5-7.

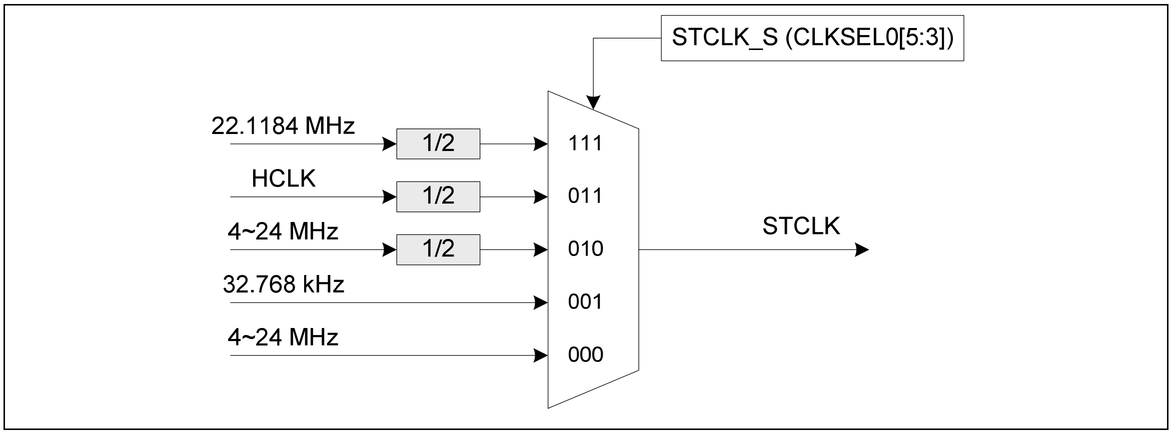


Figure 5-7 SysTick Clock Control Block Diagram

5.3.4 Peripherals Clock

The peripherals clock had different clock source switch setting which depends on the different peripheral. Please refer the CLKSEL1 and CLKSEL2 register description in 5.3.7.

5.3.5 Power Down Mode Clock

When chip enters into power down mode, system clocks, some clock sources, and some peripheral clocks will be disabled. Some clock sources and peripherals clock are still active in power down mode.

For theses clocks which still keep active list below:

- Clock Generator
  - Internal 10 kHz low speed oscillator clock
  - External 32.768 kHz low speed crystal clock
- Peripherals Clock (When these IP adopt external 32.768 kHz low speed crystal or 10 kHz low speed oscillator as clock source)

5.3.6 Frequency Divider Output

This device is equipped 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 Fin 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 FSEL (FRQDIV[3:0]).

When write 1 to DIVIDER_EN (FRQDIV[4]), the chained counter starts to count. When write 0 to DIVIDER_EN (FRQDIV[4]), the chained counter continuously runs till divided clock reaches low state and stay in low state.

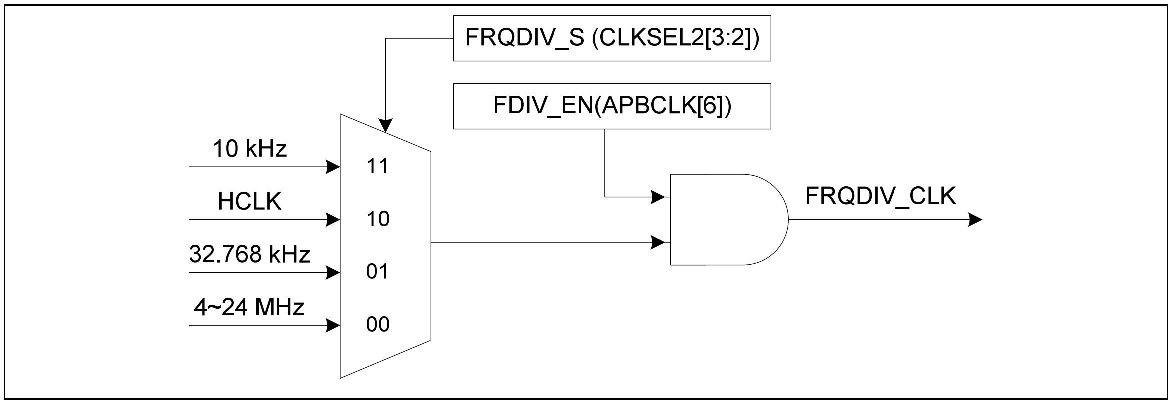


Figure 5-8 Clock Source of Frequency Divider

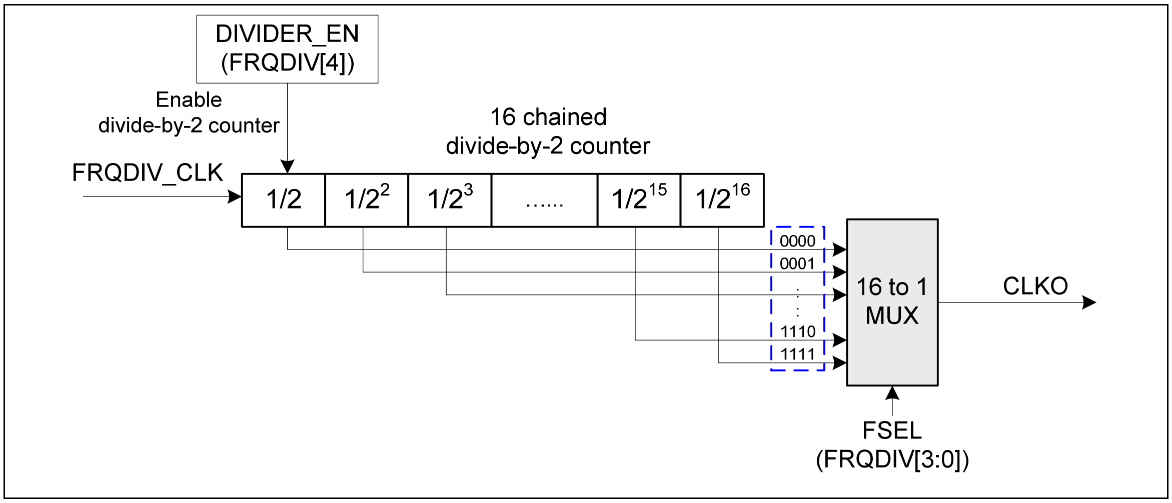


Figure 5-9 Block Diagram of Frequency Divider

5.3.7 Register Map

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

Register	Offset	R/W	Description	Reset Value
CLK_BA = 0x5000_0200				
PWRCON	CLK_BA+0x00	R/W	System Power Down Control Register	0x0000_001X
AHBCLK	CLK_BA+0x04	R/W	AHB Devices Clock Enable Control Register	0x0000_000D
APBCLK	CLK_BA+0x08	R/W	APB Devices Clock Enable Control Register	0x0000_000X
CLKSTATUS	CLK_BA+0x0C	R/W	Clock status monitor Register	0x0000_00XX
CLKSEL0	CLK_BA+0x10	R/W	Clock Source Select Control Register 0	0x0000_003X
CLKSEL1	CLK_BA+0x14	R/W	Clock Source Select Control Register 1	0xFFFF_FFFF
CLKSEL2	CLK_BA+0x1C	R/W	Clock Source Select Control Register 2	0x0000_00FF
CLKDIV	CLK_BA+0x18	R/W	Clock Divider Number Register	0x0000_0000
PLLCON	CLK_BA+0x20	R/W	PLL Control Register	0x0005_C22E
FRQDIV	CLK_BA+0x24	R/W	Frequency Divider Control Register	0x0000_0000

5.3.8 Register Description

Power Down Control Register (PWRCON)

Except the BIT[6], all the other bits are protected, program these bits need to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Reference the register REGWRPROT at address GCR_BA+0x100

Register	**Offset**	**R/W**	**Description**	**Reset Value**
PWRCON	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							**PD_WAIT_CPU**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PWR_DOWN_EN	**PD_WU_STS**	**PD_WU_INT_EN**	**PD_WU_DLY**	**OSC10K_EN**	**OSC22M_EN**	**XTL32K_EN**	**XTL12M_EN**

Bits	**Descriptions**	
[31:9]	**Reserved**	Reserve
[8]	**PD_WAIT_CPU**	**This Bit Control the Power Down Entry Condition** (write-protection bit) 1 = Chip enter power down mode when the both PD_WAIT_CPU and PWR_DOWN_EN bits are set to 1 and CPU run WFI instruction. 0 = Chip entry power down mode when the PWR_DOWN_EN bit is set to 1
[7]	**PWR_DOWN_EN**	**System Power Down Enable Bit** (write-protection bit) When this bit is set to 1, the chip power down mode is enabled and chip power down behavior will depends on the PD_WAIT_CPU bit (a) If the PD_WAIT_CPU is 0, then the chip enters power down mode immediately after the PWR_DOWN_EN bit set. (b) if the PD_WAIT_CPU 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. When in power down mode, external 4~24 MHz high speed crystal and the internal 22.1184 MHz high speed oscillator will be disabled in this mode, but the external 32.768 kHz low speed crystal and internal 10 kHz low speed oscillator are not controlled by power down mode. When 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 external 32.768 kHz low speed crystal or the internal 10 kHz low speed oscillator. 1 = Chip enter the power down mode instant or wait CPU sleep command WFI 0 = Chip operating normally or chip in idle mode because of WFI command
[6]	**PD_WU_STS**	**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 GPIO, USB, UART, WDT, CAN, ACMP, BOD or RTC wake-up occurred Write 1 to clear the bit to zero. Note: This bit is working only if PD_WU_INT_EN (PWRCON[5]) set to 1.
[5]	**PD_WU_INT_EN**	**Power Down Mode Wake-up Interrupt Enable** (write-protection bit) 0 = Disable 1 = Enable The interrupt will occur when both PD_WU_STS and PD_WU_INT_EN are high.
[4]	**PD_WU_DLY**	**Enable the Wake-up Delay Counter** (write-protection bit) 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 work at external 4~24 MHz high speed crystal, and 256 clock cycles when chip work at internal 22.1184 MHz high speed oscillator. 1 = Enable clock cycles delay 0 = Disable clock cycles delay
[3]	**OSC10K_EN**	**Internal 10 kHz Low Speed Oscillator Enable** (write-protection bit) 1 = Enable internal 10 kHz low speed oscillator 0 = Disable internal 10 kHz low speed oscillator
[2]	**OSC22M_EN**	**Internal 22.1184 MHz High Speed Oscillator Enable** (write-protection bit) 1 = Enable internal 22.1184 MHz high speed oscillator 0 = Disable internal 22.1184 MHz high speed oscillator
[1]	**XTL32K_EN**	**External 32.768 kHz Low Speed Crystal Enable** (write-protection bit) 1 = Enable external 32.768 kHz low speed crystal (Normal operation) 0 = Disable external 32.768 kHz low speed crystal
[0]	**XTL12M_EN**	**External 4~24 MHz High Speed Crystal Enable** (write-protection bit) The bit default value is set by flash controller user configuration register config0 [26:24]. When the default clock source is from external 4~24 MHz high speed crystal, this bit is set to 1 automatically 1 = Enable external 4~24 MHz high speed crystal 0 = Disable external 4~24 MHz high speed crystal

Register/Instruction Mode	PWR_DOWN_EN	PD_WAIT_CPU	CPU run WFI instruction	Clock Disable
Normal operation	0	0	NO	All clocks are disabled by control register
Idle mode (CPU entry sleep mode)	0	0	YES	Only CPU clock is disabled
Power down mode	1	0	NO	Most clocks are disabled except 10 kHz/32.768 kHz, only RTC/WDT/Timer/PWM peripheral clock are still enabled.
Power down mode (CPU entry deep sleep mode)	1	1	YES	Most clocks are disabled except 10 kHz/32.768 kHz, only RTC/WDT/Timer/PWM peripheral clock are still enabled.

Table 5-5 Power Down Mode Control Table

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

AHB Devices Clock Enable Control Register (AHBCLK)

These bits for this register are used to enable/disable clock for system clock PDMA clock and EBI clock.

Register	Offset	R/W	Description	Reset Value
AHBCLK	CLK_BA+0x04	R/W	AHB Devices Clock Enable Control Register	0x0000_000D

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				**EBI_EN**	**ISP_EN**	**PDMA_EN**	**Reserved**

Bits	Descriptions	
[31:4]	**Reserved**	Reserved
[3]	**EBI_EN**	**EBI Controller Clock Enable Control** 1 = Enable the EBI engine clock 0 = Disable the EBI engine clock
[2]	**ISP_EN**	**Flash ISP Controller Clock Enable Control** 1 = Enable the Flash ISP engine clock 0 = Disable the Flash ISP engine clock
[1]	**PDMA_EN**	**PDMA Controller Clock Enable Control** 1 = Enable the PDMA engine clock 0 = Disable the PDMA engine clock
[0]	**Reserved**	Reserved

APB Devices Clock Enable Control Register (APBCLK)

These bits of this register are used to enable/disable clock for peripheral controller clocks.

Register	Offset	R/W	Description	Reset Value
APBCLK	CLK_BA+0x08	R/W	APB Devices Clock Enable Control Register	0x0000_000X

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**
PS2_EN	**ACMP_EN**	**I2S_EN**	**ADC_EN**	**USBD_EN**	**Reserved**		**CAN0_EN**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PWM67_EN	**PWM45_EN**	**PWM23_EN**	**PWM01_EN**	**Reserved**	**UART2_EN**	**UART1_EN**	**UART0_EN**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SPI3_EN	**SPI2_EN**	**SPI1_EN**	**SPI0_EN**	**Reserved**		**I2C1_EN**	**I2C0_EN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**FDIV_EN**	**TMR3_EN**	**TMR2_EN**	**TMR1_EN**	**TMR0_EN**	**RTC_EN**	**WDT_EN**

Bits	Descriptions	
[31]	**PS2_EN**	**PS/2 Clock Enable** 1 = Enable PS/2 clock 0 = Disable PS/2 clock
[30]	**ACMP_EN**	**Analog Comparator Clock Enable** 1 = Enable the Analog Comparator Clock 0 = Disable the Analog Comparator Clock
[29]	**I2S_EN**	**I^{2}S Clock Enable** 1 = Enable I^{2}S Clock 0 = Disable I^{2}S Clock
[28]	**ADC_EN**	**Analog-Digital-Converter (ADC) Clock Enable** 1 = Enable ADC clock 0 = Disable ADC clock
[27]	**USBD_EN**	**USB 2.0 FS Device Controller Clock Enable** 1 = Enable USB clock 0 = Disable USB clock
[26:25]	**Reserved**	Reserved
[24]	**CAN0_EN**	**CAN Bus Controller-0 Clock Enable** 1 = Enable CAN0 clock 0 = Disable CAN0 clock
[23]	**PWM67_EN**	**PWM_67 Clock Enable** 1 = Enable PWM67 clock

		0 = Disable PWM67 clock
[22]	**PWM45_EN**	**PWM_45 Clock Enable** 1 = Enable PWM45 clock 0 = Disable PWM45 clock
[21]	**PWM23_EN**	**PWM_23 Clock Enable** 1 = Enable PWM23 clock 0 = Disable PWM23 clock
[20]	**PWM01_EN**	**PWM_01 Clock Enable** 1 = Enable PWM01 clock 0 = Disable PWM01 clock
[19]	**Reserved**	Reserved
[18]	**UART2_EN**	**UART2 Clock Enable** 1 = Enable UART2 clock 0 = Disable UART2 clock
[17]	**UART1_EN**	**UART1 Clock Enable** 1 = Enable UART1 clock 0 = Disable UART1 clock
[16]	**UART0_EN**	**UART0 Clock Enable** 1 = Enable UART0 clock 0 = Disable UART0 clock
[15]	**SPI3_EN**	**SPI3 Clock Enable** 1 = Enable SPI3 Clock 0 = Disable SPI3 Clock
[14]	**SPI2_EN**	**SPI2 Clock Enable** 1 = Enable SPI2 Clock 0 = Disable SPI2 Clock
[13]	**SPI1_EN**	**SPI1 Clock Enable** 1 = Enable SPI1 Clock 0 = Disable SPI1 Clock
[12]	**SPI0_EN**	**SPI0 Clock Enable** 1 = Enable SPI0 Clock 0 = Disable SPI0 Clock
[11:10]	**Reserved**	Reserved
[9]	**I2C1_EN**	**I^2C1 Clock Enable** 1 = Enable I^2C1 Clock 0 = Disable I^2C1 Clock
[8]	**I2C0_EN**	**I^2C0 Clock Enable** 1 = Enable I^2C0 Clock

		0 = Disable I^{2}C0 Clock
[7]	**Reserved**	Reserved
[6]	**FDIV_EN**	**Frequency Divider Output Clock Enable** 1 = Enable FDIV Clock 0 = Disable FDIV Clock
[5]	**TMR3_EN**	**Timer3 Clock Enable** 1 = Enable Timer3 Clock 0 = Disable Timer3 Clock
[4]	**TMR2_EN**	**Timer2 Clock Enable** 1 = Enable Timer2 Clock 0 = Disable Timer2 Clock
[3]	**TMR1_EN**	**Timer1 Clock Enable** 1 = Enable Timer1 Clock 0 = Disable Timer1 Clock
[2]	**TMR0_EN**	**Timer0 Clock Enable** 1 = Enable Timer0 Clock 0 = Disable Timer0 Clock
[1]	**RTC_EN**	**Real-Time-Clock APB interface Clock Enable** This bit is used to control the RTC APB clock only, The RTC engine clock source is from the external 32.768 kHz low speed crystal. 1 = Enable RTC Clock 0 = Disable RTC Clock
[0]	**WDT_EN**	**Watchdog Timer Clock Enable** (write-protection bit) This bit is the protected bit. It means programming this needs to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Reference the register REGWRPROT at address GCR_BA+0x100. 1 = Enable Watchdog Timer Clock 0 = Disable Watchdog Timer Clock

Clock status Register (CLKSTATUS)

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

Register	Offset	R/W	Description	Reset Value
CLKSTATUS	CLK_BA+0x0C	R/W	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**
CLK_SW_FAIL	Reserved		OSC22M_STB	OSC10K_STB	PLL_STB	XTL32K_STB	XTL12M_STB

Bits	Descriptions	
[31:8]	**Reserved**	Reserved
[7]	**CLK_SW_FAIL**	**Clock Switching Fail Flag** (write-protection bit) 1 = Clock switching failure 0 = Clock switching success 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. Write 1 to clear the bit to zero.
[6:5]	**Reserved**	Reserved
[4]	**OSC22M_STB**	**Internal 22.1184 MHz High Speed Oscillator Clock Source Stable Flag** 1 = Internal 22.1184 MHz high speed oscillator clock is stable 0 = Internal 22.1184 MHz high speed oscillator clock is not stable or disabled This is read only bit
[3]	**OSC10K_STB**	**Internal 10 kHz Low Speed Oscillator Clock Source Stable Flag** 1 = Internal 10 kHz low speed oscillator clock is stable 0 = Internal 10 kHz low speed oscillator clock is not stable or disabled This is read only bit
[2]	**PLL_STB**	**Internal PLL Clock Source Stable Flag** 1 = Internal PLL clock is stable 0 = Internal PLL clock is not stable or disabled This is read only bit

[1]	**XTL32K_STB**	**External 32.768 kHz Low Speed Crystal Clock Source Stable Flag** 1 = External 32.768 kHz low speed crystal clock is stable 0 = External 32.768 kHz low speed crystal clock is not stable or disabled This is read only bit
[0]	**XTL12M_STB**	**External 4~24 MHz High Speed Crystal Clock Source Stable Flag** 1 = External 4~24 MHz high speed crystal clock is stable 0 = External 4~24 MHz high speed crystal clock is not stable or disabled This is read only bit

Clock Source Select Control Register 0 (CLKSEL0)

Register	Offset	R/W	Description	Reset Value
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**
Reserved		**STCLK_S**			**HCLK_S**		

Bits	Descriptions	
[31:6]	**Reserved**	Reserved
[5:3]	**STCLK_S**	**Cortex_M0 SysTick clock source select** (write-protection bits) If SYST_CSR[2]=0, SysTick uses listed clock source below These bits are protected bit. It means programming this bit needs to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Reference the register REGWRPROT at address GCR_BA+0x100. 000 = Clock source from external 4~24 MHz high speed crystal clock 001 = Clock source from external 32.768 kHz low speed crystal clock 010 = Clock source from external 4~24 MHz high speed crystal clock/2 011 = Clock source from HCLK/2 111 = Clock source from internal 22.1184 MHz high speed oscillator clock/2
[2:0]	**HCLK_S**	**HCLK clock source select** (write-protection bits) 1. Before clock switching, the related clock sources (both pre-select and new-select) must be turn on 2. The 3-bit 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. 3. These bits are protected bit, It means programming this bit needs to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Reference the register REGWRPROT at address GCR_BA+0x100. 000 = Clock source from external 4~24 MHz high speed crystal clock 001 = Clock source from external 32.768 kHz low speed crystal clock 010 = Clock source from PLL clock 011 = Clock source from internal 10 kHz low speed oscillator clock

		111 = Clock source from internal 22.1184 MHz high speed oscillator clock

Clock Source Select Control Register 1(CLKSEL1)

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

Register	Offset	R/W	Description	Reset Value
CLKSEL1	CLK_BA+0x14	R/W	Clock Source Select Control Register 1	0xFFFF_FFFF

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**
PWM23_S		**PWM01_S**		**Reserved**		**UART_S**	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved	**TMR3_S**			**Reserved**	**TMR2_S**		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved	**TMR1_S**			**Reserved**	**TMR0_S**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**ADC_S**		**WDT_S**	

Bits	Descriptions	
[31:30]	**PWM23_S**	**PWM2 and PWM3 clock source select** PWM2 and PWM3 uses the same Engine clock source, both of them use the same prescaler 00 = Clock source from external 4~24 MHz high speed crystal clock 01 = Clock source from external 32.768 kHz low speed crystal clock 10 = Clock source from HCLK 11 = Clock source from internal 22.1184 MHz high speed oscillator clock
[29:28]	**PWM01_S**	**PWM0 and PWM1 clock source select** PWM0 and PWM1 uses the same Engine clock source, both of them use the same prescaler 00 = Clock source from external 4~24 MHz high speed crystal clock 01 = Clock source from external 32.768 kHz low speed crystal clock 10 = Clock source from HCLK 11 = Clock source from internal 22.1184 MHz high speed oscillator clock
[27:26]	**Reserved**	Reserved
[25:24]	**UART_S**	**UART clock source select** 00 = Clock source from external 4~24 MHz high speed crystal clock 01 = Clock source from PLL clock 11 = Clock source from internal 22.1184 MHz high speed oscillator clock
[23]	**Reserved**	Reserved

[22:20]	**TMR3_S**	**TIMER3 clock source select** 000 = Clock source from external 4~24 MHz high speed crystal clock 001 = Clock source from external 32.768 kHz low speed crystal clock 010 = Clock source from HCLK 111 = Clock source from internal 22.1184 MHz high speed oscillator clock
[19]	**Reserved**	Reserved
[18:16]	**TMR2_S**	**TIMER2 clock source select** 000 = Clock source from external 4~24 MHz high speed crystal clock 001 = Clock source from external 32.768 kHz low speed crystal clock 010 = Clock source from HCLK 111 = Clock source from internal 22.1184 MHz high speed oscillator clock
[15]	**Reserved**	Reserved
[14:12]	**TMR1_S**	**TIMER1 clock source select** 000 = Clock source from external 4~24 MHz high speed crystal clock 001 = Clock source from external 32.768 kHz low speed crystal clock 010 = Clock source from HCLK 111 = Clock source from internal 22.1184 MHz high speed oscillator clock
[11]	**Reserved**	Reserved
[10:8]	**TMR0_S**	**TIMER0 clock source select** 000 = Clock source from external 4~24 MHz high speed crystal clock 001 = Clock source from external 32.768 kHz low speed crystal clock 010 = Clock source from HCLK 111 = Clock source from internal 22.1184 MHz high speed oscillator clock
[7:4]	**Reserved**	Reserved
[3:2]	**ADC_S**	**ADC clock source select** 00 = Clock source from external 4~24 MHz high speed crystal clock 01 = Clock source from PLL clock 10 = Clock source from HCLK 11 = Clock source from internal 22.1184 MHz high speed oscillator clock
[1:0]	**WDT_S**	**Watchdog Timer clock source select** (write-protection bits) These bits are protected bit, program this need to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Reference the register REGWRPROT at address GCR_BA+0x100. 01 = Clock source from external 32.768 kHz low speed crystal clock 10 = Clock source from HCLK/2048 clock 11 = Clock source from internal 10 kHz low speed oscillator clock

Clock Source Select Control Register 2 (CLKSEL2)

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

Register	Offset	R/W	Description	Reset Value
CLKSEL2	CLK_BA+0x1C	R/W	Clock Source Select Control Register 2	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**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PWM67_S		**PWM45_S**		**FRQDIV_S**		**I2S_S**	

Bits	Descriptions	
[31:8]	**Reserved**	Reserved
[7:6]	**PWM67_S**	**PWM6 and PWM7 Clock Source Select** PWM6 and PWM7 used the same Engine clock source, both of them use the same prescaler 00 = Clock source from external 4~24 MHz high speed crystal clock 01 = Clock source from external 32.768 kHz low speed crystal clock 10 = Clock source from HCLK 11 = Clock source from internal 22.1184 MHz high speed oscillator clock
[5:4]	**PWM45_S**	**PWM4 and PWM5 Clock Source Select** PWM4 and PWM5 used the same Engine clock source, both of them use the same prescaler 00 = Clock source from external 4~24 MHz high speed crystal clock 01 = Clock source from external 32.768 kHz low speed crystal clock 10 = Clock source from HCLK 11 = Clock source from internal 22.1184 MHz high speed oscillator clock
[3:2]	**FRQDIV_S**	**Clock Divider Clock Source Select** 00 = Clock source from external 4~24 MHz high speed crystal clock 01 = Clock source from external 32.768 kHz low speed crystal clock 10 = Clock source from HCLK 11 = Clock source from internal 22.1184 MHz high speed oscillator clock
[1:0]	**I2S_S**	**I^{2}S Clock Source Select** 00 = Clock source from external 4~24 MHz high speed crystal clock

		01 = Clock source from PLL clock 10 = Clock source from HCLK 11 = Clock source from internal 22.1184 MHz high speed oscillator clock

Clock Divider Register (CLKDIV)

Register	Offset	R/W	Description	Reset Value
CLKDIV	CLK_BA+0x18	R/W	Clock Divider Number Register	0x0000_0000

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

Bits	Descriptions	
[31:24]	**Reserved**	Reserved
[23:16]	**ADC_N**	**ADC clock divide number from ADC clock source** The ADC clock frequency = (ADC clock source frequency) / (ADC_N + 1)
[15:12]	**Reserved**	Reserved
[11:8]	**UART_N**	**UART clock divide number from UART clock source** The UART clock frequency = (UART clock source frequency) / (UART_N + 1)
[7:4]	**USB_N**	**USB clock divide number from PLL clock** The USB clock frequency = (PLL frequency) / (USB_N + 1)
[3:0]	**HCLK_N**	**HCLK clock divide number from HCLK clock source** The HCLK clock frequency = (HCLK clock source frequency) / (HCLK_N + 1)

PLL Control Register (PLLCON)

The PLL reference clock input is from the external 4~24 MHz high speed crystal clock input or from the internal 22.1184 MHz high speed oscillator. These registers are use to control the PLL output frequency and PLL operating mode

Register	Offset	R/W	Description	Reset Value
PLLCON	CLK_BA+0x20	R/W	PLL Control Register	0x0005_C22E

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

Bits	Descriptions	
[31:20]	**Reserved**	Reserved
[19]	**PLL_SRC**	**PLL Source Clock Select** 1 = PLL source clock from internal 22.1184 MHz high speed oscillator 0 = PLL source clock from external 4~24 MHz high speed crystal
[18]	**OE**	**PLL OE (FOUT enable) pin Control** 0 = PLL FOUT enable 1 = PLL FOUT is fixed low
[17]	**BP**	**PLL Bypass Control** 0 = PLL is in normal mode (default) 1 = PLL clock output is same as clock input (XTALin)
[16]	**PD**	**Power Down Mode** If set the PWR_DOWN_EN bit to 1 in PWRCON register, the PLL will enter power down mode too. 0 = PLL is in normal mode 1 = PLL is in power down mode (default)
[15:14]	**OUT_DV**	**PLL Output Divider Control Pins** Refer to the formulas below the table.
[13:9]	**IN_DV**	**PLL Input Divider Control Pins** Refer to the formulas below the table.
[8:0]	**FB_DV**	**PLL Feedback Divider Control Pins**

		Refer to the formulas below the table.

Output Clock Frequency Setting

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

Constraint:

1. $3.2MHz < FIN < 150MHz$
2. $800KHz < \frac{FIN}{2*NR} < 8MHz$

$$100MHz < FCO = FIN \times \frac{NF}{NR} < 200MHz$$

3. $120MHz < FCO$ is preferred

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

Default Frequency Setting

The default value : 0xC22E
FIN = 12 MHz
NR = (1+2) = 3
NF = (46+2) = 48
NO = 4
FOUT = 12/4 x 48 x 1/3 = 48 MHz

Frequency Divider Control Register (FRQDIV)

Register	Offset	R/W	Description	Reset Value
FRQDIV	CLK_BA+ 24	R/W	Frequency Divider 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			**DIVIDER_EN**	**FSEL**			

Bits	Descriptions	
[31:5]	**Reserved**	Reserved
[4]	**DIVIDER_EN**	**Frequency Divider Enable Bit** 0 = Disable Frequency Divider 1 = Enable Frequency Divider
[3:0]	**FSEL**	**Divider Output Frequency Selection Bits** The formula of output frequency is $F_{out} = F_{in}/2^{(N+1)}$ F_{in} is the input clock frequency. F_{out} is the frequency of divider output clock. N is the 4-bit value of FSEL[3:0].

5.4 USB Device Controller (USB)

5.4.1 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 support 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. Users need to set the effective starting address of SRAM for each endpoint buffer through "buffer segmentation register (BUFSEGx)".

There are 6 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 states, current start address, transaction status, and data buffer status for each endpoint.

There are four different interrupt events in this controller. They are the wake-up function, 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 (USB_INTSTS) to acknowledge what kind of interrupt occurring, and then check the related USB Endpoint Status Register (USB_EPSTS) to acknowledge what kind of event occurring in this endpoint.

A software-disable function is also supported for this USB controller. It is used to simulate the disconnection of this device from the host. If user enables DRVSE0 bit (USB_DRVSE0), the USB controller will force the output of USB_DP and USB_DM to level low and its function is disabled. After disable the DRVSE0 bit, host will enumerate the USB device again.

Reference: Universal Serial Bus Specification Revision 1.1

5.4.2 Features

This Universal Serial Bus (USB) performs a serial interface with a single connector type for attaching all USB peripherals to the host system. Following is the feature listing of this USB.

- Compliant with USB 2.0 Full-Speed specification
- Provide 1 interrupt vector with 4 different interrupt events (WAKEUP, FLDET, USB and BUS)
- Support Control/Bulk/Interrupt/Isochronous transfer type
- Support suspend function when no bus activity existing for 3 ms
- Provide 6 endpoints for configurable Control/Bulk/Interrupt/Isochronous transfer types and maximum 512 bytes buffer size
- Provide remote wake-up capability

5.4.3 Block Diagram

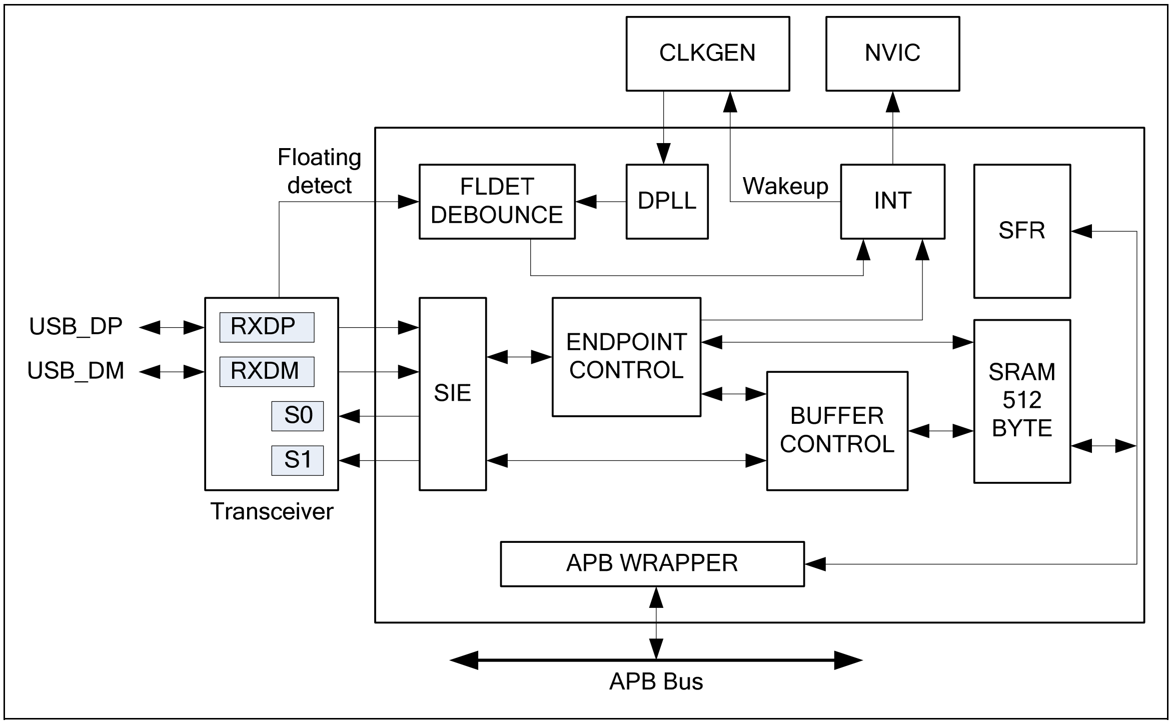


Figure 5-10 USB Block Diagram

5.4.4 Function Description

5.4.4.1 SIE (Serial Interface Engine)

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, transaction sequencing
- SOP, EOP, RESET, RESUME signal detection/generation
- Clock/Data separation
- NRZI Data encoding/decoding and bit-stuffing
- CRC generation and checking (for Token and Data)
- Packet ID (PID) generation and checking/ decoding
- Serial-Parallel/ Parallel-Serial conversion

5.4.4.2 Endpoint Control

There are 6 endpoints in this controller. 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.

5.4.4.3 Digital Phase Lock Loop

The bit rate of USB data is 12 MHz. The DPLL use 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.

5.4.4.4 Floating De-bounce

A USB device may be plug-in or plug-out from the USB host. In order to monitor the state of a USB device when it is detached from the USB host, the device controller provides hardware de-bounce for USB floating detect interrupt to avoid bounce problems on USB plug-in or unplug. Floating detect interrupt appears about 10 ms later than USB plug-in or plug-out. A user can acknowledge USB plug-in/plug-out by reading register “USB_FLDET”. The flag in “FLDET” represents the current state on the bus without de-bounce. If the FLDET is 1, it means the controller has plug-in the USB. If the user polling this flag to check USB state, he/she must add software de-bounce if necessary.

5.4.4.5 Interrupt

This USB provides 1 interrupt vector with 4 interrupt events (WAKEUP, FLDET, USB and BUS). The WAKEUP event is used to wake-up the system clock when the power down mode is enabled. (The power mode function is defined in system power down control register, PWRCON). The FLDET event is used for USB plug-in or unplug. The USB event notifies users of some USB requests, like IN ACK, OUT ACK etc., and the BUS event notifies users of some bus events, like suspend, resume, etc. User must set related bits in the interrupt enable register (USB_INTEN) of USB Device Controller to enable USB interrupts.

Wake-up interrupt is only present when the chip entered power down mode and then wake-up event had happened. After the chip enters power down mode, any change on USB_DP and USB_DM can wake-up this chip (provided that USB wake-up function is enabled). If this change is not intentionally, no interrupt but wake-up interrupt will occur. After USB wake-up, this interrupt will occur when no other USB interrupt events are present for more than 20ms. The following figure is the control flow of wake-up interrupt.

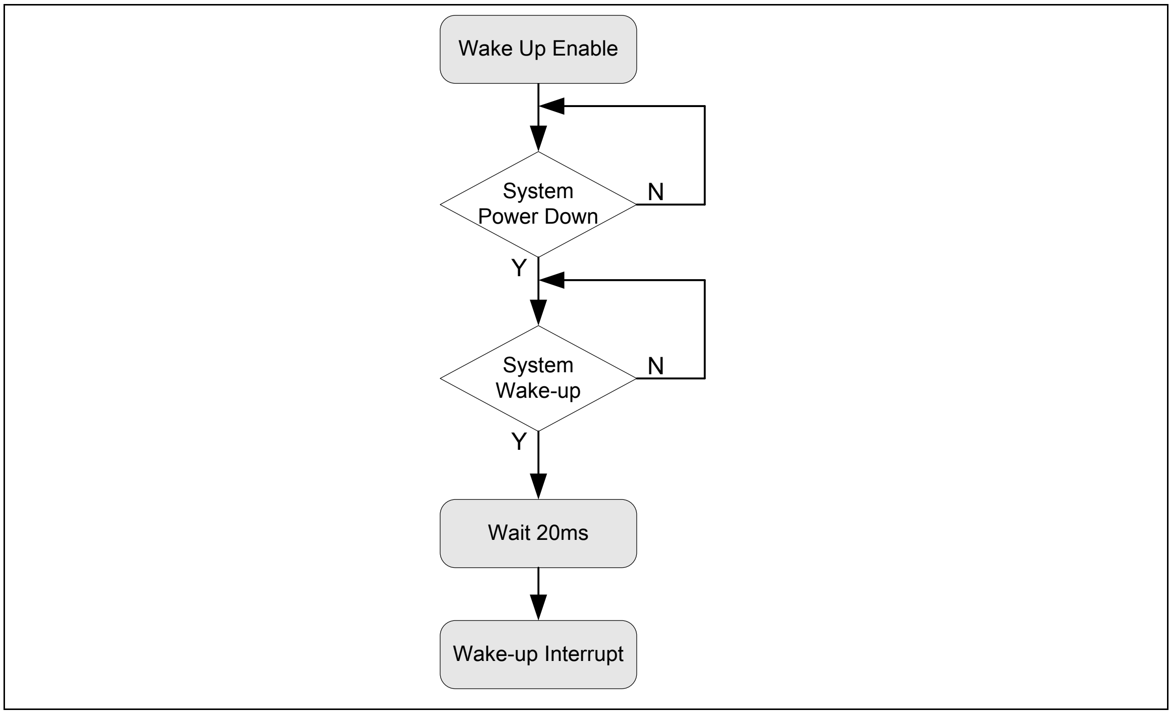


Figure 5-11 Wake-up Interrupt Operation Flow

USB interrupt is used to notify users of any USB event on the bus, and a user can read EPSTS (USB_EPSTS[25:8]) and EPEVT5~0 (USB_INTSTS[21:16]) to know what kind of request is to which endpoint and 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 USB_ATTR to acknowledge bus events.

5.4.4.6 Power Saving

USB turns off PHY transceiver automatically to save power while this chip enters power down mode. Furthermore, a user can write 0 into USB_ATTR[4] to turn off PHY under special circumstances like suspend to save power.

5.4.4.7 Buffer Control

There is 512 bytes SRAM in the controller and the 6 endpoints share this buffer. The 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 MXPLD register.

Figure 5-12 depicts the starting address for each endpoint according the content of BUFSEG and MXPLD registers. If the BUFSEG0 is programmed as 0x08h and MXPLD0 is set as 0x40h, the SRAM size of endpoint 0 is start from USB_BA+0x108h and end in USB_BA+0x148h. (Note: the USB SRAM base is USB_BA+0x100h).

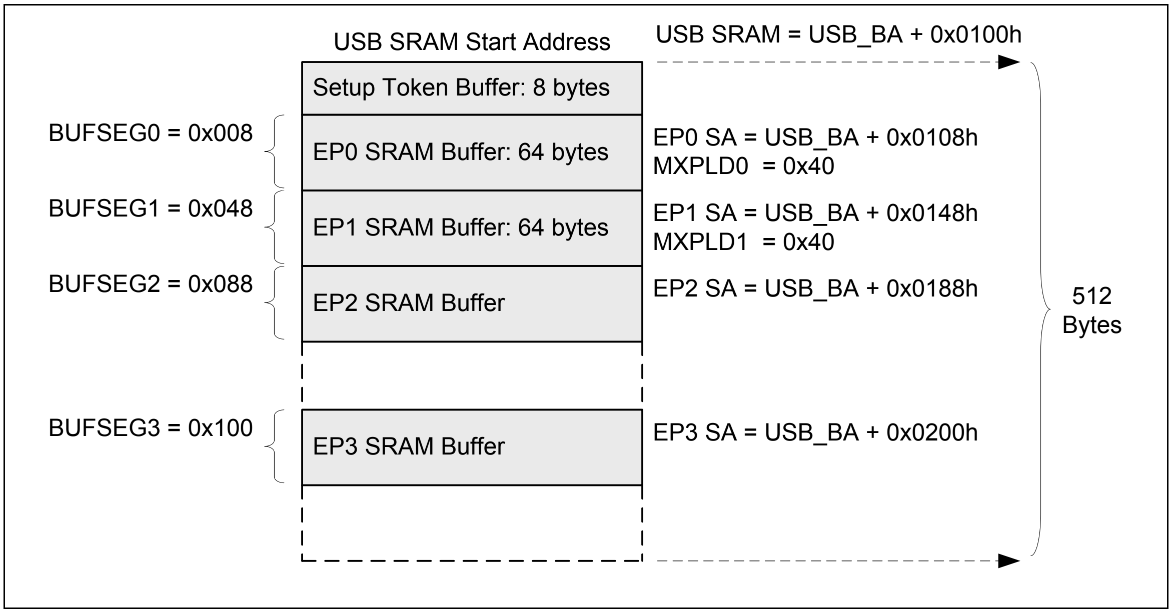


Figure 5-12 Endpoint SRAM Structure

5.4.4.8 Handling Transactions with USB Device Peripheral

User can use interrupt or polling USB_INTSTS to monitor the USB Transactions, when transactions occur, USB_INTSTS will be set by hardware and send an interrupt request to CPU (if related interrupt enabled), or user can polling USB_INTSTS to get these events without interrupt. The following is the control flow with interrupt enable.

When USB host has requested data from device controller, users need to prepare related data into the specified endpoint buffer in advance. After buffering the required data, users need to write the actual data length in the specified MAXPLD 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.

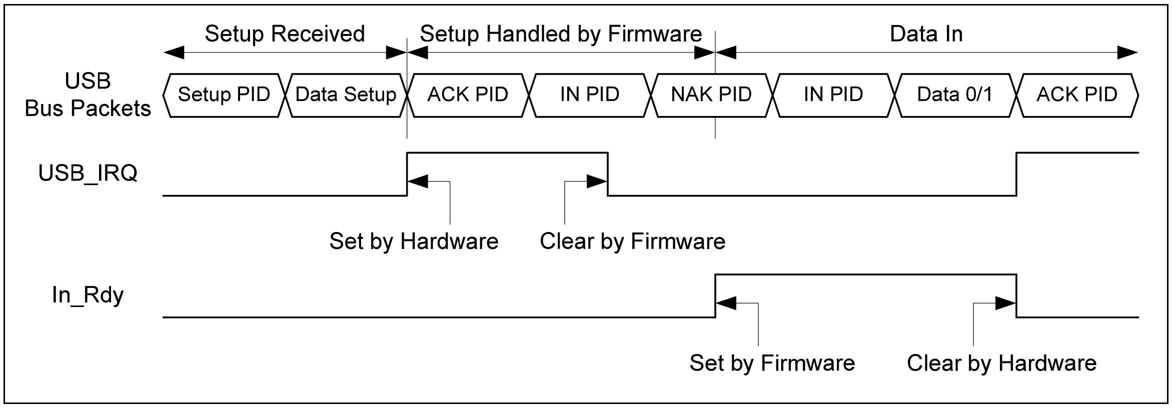


Figure 5-13 Setup Transaction followed by Data in Transaction

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 related MAXPLD register and de-assert the signal "Out_Rdy". This will avoid hardware accepting next transaction until users move out current data in the related endpoint buffer. Once users have processed this transaction, the related register "MAXPLD" needs to be written by firmware to assert the signal "Out_Rdy" again to accept next transaction.

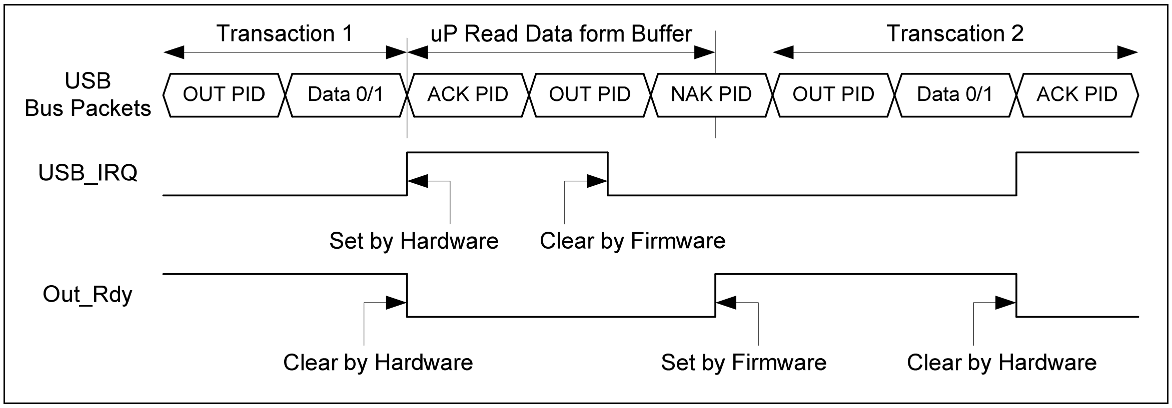


Figure 5-14 Data Out Transfer

5.4.5 Register and Memory Map

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

Register	Offset	R/W	Description	Reset Value
USB_BA = 0x4006_0000				
USB_INTEN	USB_BA+0x000	R/W	USB Interrupt Enable Register	0x0000_0000
USB_INTSTS	USB_BA+0x004	R/W	USB Interrupt Event Status Register	0x0000_0000
USB_FADDR	USB_BA+0x008	R/W	USB Device Function Address Register	0x0000_0000
USB_EPSTS	USB_BA+0x00C	R	USB Endpoint Status Register	0x0000_00x0
USB_ATTR	USB_BA+0x010	R/W	USB Bus Status and Attribution Register	0x0000_0040
USB_FLDET	USB_BA+0x014	R	USB Floating Detected Register	0x0000_0000
USB_BUFSEG	USB_BA+0x018	R/W	Setup Token Buffer Segmentation Register	0x0000_0000
USB_BUFSEG0	USB_BA+0x020	R/W	Endpoint 0 Buffer Segmentation Register	0x0000_0000
USB_MXPLD0	USB_BA+0x024	R/W	Endpoint 0 Maximal Payload Register	0x0000_0000
USB_CFG0	USB_BA+0x028	R/W	Endpoint 0 Configuration Register	0x0000_0000
USB_CFGP0	USB_BA+0x02C	R/W	Endpoint 0 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_BUFSEG1	USB_BA+0x030	R/W	Endpoint 1 Buffer Segmentation Register	0x0000_0000
USB_MXPLD1	USB_BA+0x034	R/W	Endpoint 1 Maximal Payload Register	0x0000_0000
USB_CFG1	USB_BA+0x038	R/W	Endpoint 1 Configuration Register	0x0000_0000
USB_CFGP1	USB_BA+0x03C	R/W	Endpoint 1 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_BUFSEG2	USB_BA+0x040	R/W	Endpoint 2 Buffer Segmentation Register	0x0000_0000
USB_MXPLD2	USB_BA+0x044	R/W	Endpoint 2 Maximal Payload Register	0x0000_0000
USB_CFG2	USB_BA+0x048	R/W	Endpoint 2 Configuration Register	0x0000_0000
USB_CFGP2	USB_BA+0x04C	R/W	Endpoint 2 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_BUFSEG3	USB_BA+0x050	R/W	Endpoint 3 Buffer Segmentation Register	0x0000_0000
USB_MXPLD3	USB_BA+0x054	R/W	Endpoint 3 Maximal Payload Register	0x0000_0000
USB_CFG3	USB_BA+0x058	R/W	Endpoint 3 Configuration Register	0x0000_0000
USB_CFGP3	USB_BA+0x05C	R/W	Endpoint 3 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_BUFSEG4	USB_BA+0x060	R/W	Endpoint 4 Buffer Segmentation Register	0x0000_0000

USB_MXPLD4	USB_BA+0x064	R/W	Endpoint 4 Maximal Payload Register	0x0000_0000
USB_CFG4	USB_BA+0x068	R/W	Endpoint 4 Configuration Register	0x0000_0000
USB_CFGP4	USB_BA+0x06C	R/W	Endpoint 4 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_BUFSEG5	USB_BA+0x070	R/W	Endpoint 5 Buffer Segmentation Register	0x0000_0000
USB_MXPLD5	USB_BA+0x074	R/W	Endpoint 5 Maximal Payload Register	0x0000_0000
USB_CFG5	USB_BA+0x078	R/W	Endpoint 5 Configuration Register	0x0000_0000
USB_CFGP5	USB_BA+0x07C	R/W	Endpoint 5 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_DRVSE0	USB_BA+0x090	R/W	USB Drive SE0 Control Register	0x0000_0001

Memory Type	**Address**	**Size**	**Description**
USB_BA = 0x4006_0000			
SRAM	USB_BA+0x100 ~ USB_BA+0x2FF	512 Bytes	The SRAM is used for the entire endpoints buffer. Refer to section 5.4.4.7 for the endpoint SRAM structure and its description.

5.4.6 Register Description

USB Interrupt Enable Register (USB_INTEN)

Register	Offset	R/W	Description	Reset Value
USB_INTEN	USB_BA+0x000	R/W	USB 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**
INNAK_EN	Reserved						WAKEUP_EN
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				WAKEUP_IE	FLDET_IE	USB_IE	BUS_IE

Bits	Descriptions	
[31:16]	**Reserved**	Reserved
[15]	**INNAK_EN**	**Active NAK Function and its Status in IN Token** 1 = The NAK status is updated into the endpoint status register, USB_EPSTS, when it is set to 1 and there is NAK response in IN token. It also enable the interrupt event when the device responds NAK after receiving IN token 0 = The NAK status doesn't be updated into the endpoint status register when it was set to 0. It also disable the interrupt event when device responds NAK after receiving IN token
[14:9]	**Reserved**	Reserved
[8]	**WAKEUP_EN**	**Wake-up Function Enable** 1 = Enable USB wake-up function 0 = Disable USB wake-up function
[7:4]	**Reserved**	Reserved
[3]	**WAKEUP_IE**	**USB Wake-up Interrupt Enable** 1 = Enable wake-up Interrupt 0 = Disable wake-up Interrupt
[2]	**FLDET_IE**	**Floating Detected Interrupt Enable** 1 = Enable Floating detect Interrupt 0 = Disable Floating detect Interrupt
[1]	**USB_IE**	**USB Event Interrupt Enable** 1 = Enable USB event interrupt

		0 = Disable USB event interrupt
[0]	**BUS_IE**	**Bus Event Interrupt Enable** 1 = Enable BUS event interrupt 0 = Disable BUS event interrupt

USB Interrupt Event Status Register (USB_INTSTS)

This register is USB Interrupt Event Status register; clear by write ‘1’ to the corresponding bit.

Register	Offset	R/W	Description	Reset Value
USB_INTSTS	USB_BA+0x004	R/W	USB Interrupt Event Status Register	0x0000_0000

31	30	29	28	27	26	25	24
SETUP	Reserved						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved		EPEVT5	EPEVT4	EPEVT3	EPEVT2	EPEVT1	EPEVT0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				WAKEUP_STS	FLDET_STS	USB_STS	BUS_STS

Bits	Descriptions	
[31]	**SETUP**	**Setup Event Status** 1 = Setup event occurred, cleared by write 1 to USB_INTSTS[31] 0 = No Setup event
[30:22]	**Reserved**	Reserved
[21]	**EPEVT5**	**Endpoint 5’s USB Event Status** 1 = USB event occurred on Endpoint 5, check USB_EPSTS[25:23] to know which kind of USB event was occurred, cleared by write 1 to USB_INTSTS[21] or USB_INTSTS[1] 0 = No event occurred in endpoint 5
[20]	**EPEVT4**	**Endpoint 4’s USB Event Status** 1 = USB event occurred on Endpoint 4, check USB_EPSTS[22:20] to know which kind of USB event was occurred, cleared by write 1 to USB_INTSTS[20] or USB_INTSTS[1] 0 = No event occurred in endpoint 4
[19]	**EPEVT3**	**Endpoint 3’s USB Event Status** 1 = USB event occurred on Endpoint 3, check USB_EPSTS[19:17] to know which kind of USB event was occurred, cleared by write 1 to USB_INTSTS[19] or USB_INTSTS[1] 0 = No event occurred in endpoint 3
[18]	**EPEVT2**	**Endpoint 2’s USB Event Status** 1 = USB event occurred on Endpoint 2, check USB_EPSTS[16:14] to know which kind of USB event was occurred, cleared by write 1 to USB_INTSTS[18] or USB_INTSTS[1] 0 = No event occurred in endpoint 2

[17]	**EPEVT1**	**Endpoint 1's USB Event Status** 1 = USB event occurred on Endpoint 1, check USB_EPSTS[13:11] to know which kind of USB event was occurred, cleared by write 1 to USB_INTSTS[17] or USB_INTSTS[1] 0 = No event occurred in endpoint 1
[16]	**EPEVT0**	**Endpoint 0's USB Event Status** 1 = USB event occurred on Endpoint 0, check USB_EPSTS[10:8] to know which kind of USB event was occurred, cleared by write 1 to USB_INTSTS[16] or USB_INTSTS[1] 0 = No event occurred in endpoint 0
[15:4]	**Reserved**	Reserved
[3]	**WAKEUP_STS**	**Wake-up Interrupt Status** 1 = Wake-up event occurred, cleared by write 1 to USB_INTSTS[3] 0 = No Wake-up event is occurred
[2]	**FLDET_STS**	**Floating Detected Interrupt Status** 1 = There is attached/detached event in the USB bus and it is cleared by write 1 to USB_INTSTS[2]. 0 = There is not attached/detached event in the USB
[1]	**USB_STS**	**USB event Interrupt Status** The USB event includes the Setup Token, IN Token, OUT ACK, ISO IN, or ISO OUT events in the bus. 1 = USB event occurred, check EPSTS0~5[2:0] to know which kind of USB event was occurred, cleared by write 1 to USB_INTSTS[1] or EPSTS0~5 and SETUP (USB_INTSTS[31]) 0 = No any USB event is occurred
[0]	**BUS_STS**	**BUS Interrupt Status** The BUS event means that there is one of the suspense or the resume function in the bus. 1 = Bus event occurred; check USB_ATTR[3:0] to know which kind of bus event was occurred, cleared by write 1 to USB_INTSTS[0]. 0 = No any BUS event is occurred

USB Device Function Address Register (USB_FADDR)

A seven-bit value uses as the address of a device on the USB BUS.

Register	Offset	R/W	Description	Reset Value
USB_FADDR	USB_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	Descriptions	
[31:7]	**Reserved**	Reserved
[6:0]	**FADDR**	USB device's Function Address

USB Endpoint Status Register (USB_EPSTS)

Register	Offset	R/W	Description	Reset Value
USB_EPSTS	USB_BA+0x00C	R	USB Endpoint Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						**EPSTS5[2:1]**	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
EPSTS5[0]	**EPSTS4[2:0]**			**EPSTS3[2:0]**			**EPSTS2[2]**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
EPSTS2[1:0]		**EPSTS1[2:0]**			**EPSTS0[2:0]**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
OVERRUN	**Reserved**						

Bits	Descriptions	
[31:26]	**Reserved**	Reserved
[25:23]	**EPSTS5**	**Endpoint 5 Bus Status** These bits are used to indicate the current status of this endpoint 000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end
[22:20]	**EPSTS4**	**Endpoint 4 Bus Status** These bits are used to indicate the current status of this endpoint 000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end
[19:17]	**EPSTS3**	**Endpoint 3 Bus Status** These bits are used to indicate the current status of this endpoint 000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK

		011 = Setup ACK 111 = Isochronous transfer end
[16:14]	**EPSTS2**	**Endpoint 2 Bus Status** These bits are used to indicate the current status of this endpoint 000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end
[13:11]	**EPSTS1**	**Endpoint 1 Bus Status** These bits are used to indicate the current status of this endpoint 000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end
[10:8]	**EPSTS0**	**Endpoint 0 Bus Status** These bits are used to indicate the current status of this endpoint 000 = In ACK 001 = In NAK 010 = Out Packet Data0 ACK 110 = Out Packet Data1 ACK 011 = Setup ACK 111 = Isochronous transfer end
[7]	**OVERRUN**	**Overrun** It indicates that the received data is over the maximum payload number or not. 1 = It indicates that the Out Data more than the Max Payload in MXPLD register or the Setup Data more than 8 Bytes 0 = No overrun
[6:0]	**Reserved**	Reserved

USB Bus Status and Attribution Register (USB_ATTR)

Register	Offset	R/W	Description	Reset Value
USB_ATTR	USB_BA+0x010	R/W	USB 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**	**PWRDN**	**DPPU_EN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
USB_EN	**Reserved**	**RWAKEUP**	**PHY_EN**	**TIMEOUT**	**RESUME**	**SUSPEND**	**USBRST**

Bits	Descriptions	
[31:11]	**Reserved**	Reserved
[10]	**BYTEM**	**CPU access USB SRAM Size Mode Select** 1 = Byte Mode: The size of the transfer from CPU to USB SRAM can be Byte only. 0 = Word Mode: The size of the transfer from CPU to USB SRAM can be Word only.
[9]	**PWRDN**	**Power down PHY Transceiver, low active** 1 = Turn-on related circuit of PHY transceiver 0 = Power down related circuit of PHY transceiver
[8]	**DPPU_EN**	**Pull-up resistor on USB_DP enable** 1 = The pull-up resistor in USB_DP bus active 0 = Disable the pull-up resistor in USB_DP bus
[7]	**USB_EN**	**USB Controller Enable** 1 = Enable USB Controller 0 = Disable USB Controller
[6]	**Reserved**	Reserved
[5]	**RWAKEUP**	**Remote wake-up** 1 = Force USB bus to K (USB_DP low, USB_DM: high) state, used for remote wake-up 0 = Release the USB bus from K state
[4]	**PHY_EN**	**PHY Transceiver Function Enable** 1 = Enable PHY transceiver function 0 = Disable PHY transceiver function
[3]	**TIMEOUT**	**Time Out Status**

		1 = Bus no any response more than 18 bits time 0 = No time out It is a read only bit.
[2]	**RESUME**	**Resume Status** 1 = Resume from suspend 0 = No bus resume It is a read only bit.
[1]	**SUSPEND**	**Suspend Status** 1 = Bus idle more than 3ms, either cable is plugged off or host is sleeping 0 = Bus no suspend It is a read only bit.
[0]	**USBRST**	**USB Reset Status** 1 = Bus reset when SE0 (single-ended 0) more than 2.5us 0 = Bus no reset It is a read only bit.

Floating detection Register (USB_FLDET)

Register	Offset	R/W	Description	Reset Value
USB_FLDET	USB_BA+0x014	R	USB Floating Detected 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							**FLDET**

Bits	Descriptions	
[31:1]	**Reserved**	Reserved
[0]	**FLDET**	**Device Floating Detected** 1 = When the controller is attached into the BUS, this bit will be set as 1 0 = The controller didn't attached into the USB host

Buffer Segmentation Register (USB_BUFSEG)

For Setup token only.

Register	Offset	R/W	Description	Reset Value
USB_BUFSEG	USB_BA+0x018	R/W	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							**BUFSEG[8]**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
BUFSEG[7:3]					**Reserved**		

Bits	Descriptions	
[31:9]	**Reserved**	Reserved
[8:3]	**BUFSEG**	It is used to indicate the offset address for the Setup token with the USB SRAM starting address. The effective starting address is USB_SRAM address + { BUFSEG[8:3], 3'b000} Where the USB_SRAM address = USB_BA+0x100h. Note: It is used for Setup token only.
[2:0]	**Reserved**	Reserved

Buffer Segmentation Register (BUFSEGx) x = 0~5

Register	Offset	R/W	Description	Reset Value
USB_BUFSEG0	USB_BA+0x020	R/W	Endpoint 0 Buffer Segmentation Register	0x0000_0000
USB_BUFSEG1	USB_BA+0x030	R/W	Endpoint 1 Buffer Segmentation Register	0x0000_0000
USB_BUFSEG2	USB_BA+0x040	R/W	Endpoint 2 Buffer Segmentation Register	0x0000_0000
USB_BUFSEG3	USB_BA+0x050	R/W	Endpoint 3 Buffer Segmentation Register	0x0000_0000
USB_BUFSEG4	USB_BA+0x060	R/W	Endpoint 4 Buffer Segmentation Register	0x0000_0000
USB_BUFSEG5	USB_BA+0x070	R/W	Endpoint 5 Buffer Segmentation Register	0x0000_0000

<table>
<tr><td>31</td><td>30</td><td>29</td><td>28</td><td>27</td><td>26</td><td>25</td><td>24</td></tr>
<tr><td colspan="8">Reserved</td></tr>
<tr><td>23</td><td>22</td><td>21</td><td>20</td><td>19</td><td>18</td><td>17</td><td>16</td></tr>
<tr><td colspan="8">Reserved</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>12</td><td>11</td><td>10</td><td>9</td><td>8</td></tr>
<tr><td colspan="7">Reserved</td><td>BUFSEG[8]x</td></tr>
<tr><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
<tr><td colspan="5">BUFSEG[7:3]x</td><td colspan="3">Reserved</td></tr>
</table>

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[31:9]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[8:3]</td><td>BUFSEGx</td><td>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
USB_SRAM address + { BUFSEG[8:3], 3’b000}
Where the USB_SRAM address = USB_BA+0x100h.
Refer to section 5.4.4.7 for the endpoint SRAM structure and its description.</td></tr>
<tr><td>[2:0]</td><td>Reserved</td><td>Reserved</td></tr>
</table>

Maximal Payload Register (USB_MXPLDx) x = 0~5

Register	Offset	R/W	Description	Reset Value
USB_MXPLD0	USB_BA+0x024	R/W	Endpoint 0 Maximal Payload Register	0x0000_0000
USB_MXPLD1	USB_BA+0x034	R/W	Endpoint 1 Maximal Payload Register	0x0000_0000
USB_MXPLD2	USB_BA+0x044	R/W	Endpoint 2 Maximal Payload Register	0x0000_0000
USB_MXPLD3	USB_BA+0x054	R/W	Endpoint 3 Maximal Payload Register	0x0000_0000
USB_MXPLD4	USB_BA+0x064	R/W	Endpoint 4 Maximal Payload Register	0x0000_0000
USB_MXPLD5	USB_BA+0x074	R/W	Endpoint 5 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[8]
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MXPLD[7:0]							

Bits	Descriptions	
[31:9]	**Reserved**	Reserved
[8:0]	**MXPLD**	**Maximal Payload** It is used to 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 in 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 the data length be transmitted to host For OUT token, the value of MXPLD is indicated the actual data length receiving from host. Note that once MXPLD is written, the data packets will be transmitted/received immediately after IN/OUT token arrived.

Configuration Register (USB_CFGx) x = 0~5

Register	Offset	R/W	Description	Reset Value
USB_CFG0	USB_BA+0x028	R/W	Endpoint 0's Configuration Register	0x0000_0000
USB_CFG1	USB_BA+0x038	R/W	Endpoint 1's Configuration Register	0x0000_0000
USB_CFG2	USB_BA+0x048	R/W	Endpoint 2's Configuration Register	0x0000_0000
USB_CFG3	USB_BA+0x058	R/W	Endpoint 3's Configuration Register	0x0000_0000
USB_CFG4	USB_BA+0x068	R/W	Endpoint 4's Configuration Register	0x0000_0000
USB_CFG5	USB_BA+0x078	R/W	Endpoint 5's 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**
DSQ_SYNC	STATE		ISOCH	EP_NUM			

Bits	Descriptions	
[31:10]	**Reserved**	Reserved
[9]	**CSTALL**	**Clear STALL Response** 1 = Clear the device to response STALL handshake in setup stage 0 = Disable the device to clear the STALL handshake in setup stage
[8]	**Reserved**	Reserved
[7]	**DSQ_SYNC**	**Data Sequence Synchronization** 1 = DATA1 PID 0 = DATA0 PID It is used to specify the DATA0 or DATA1 PID in the following IN token transaction. H/W 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 handshake. 1 = Isochronous endpoint 0 = No Isochronous endpoint
[3:0]	**EP_NUM**	**Endpoint Number** These bits are used to define the endpoint number of the current endpoint

Extra Configuration Register (USB_CFGPx) x = 0~5

Register	Offset	R/W	Description	Reset Value
USB_CFGP0	USB_BA+0x02C	R/W	Endpoint 0 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_CFGP1	USB_BA+0x03C	R/W	Endpoint 1 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_CFGP2	USB_BA+0x04C	R/W	Endpoint 2 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_CFGP3	USB_BA+0x05C	R/W	Endpoint 3 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_CFGP4	USB_BA+0x06C	R/W	Endpoint 4 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_CFGP5	USB_BA+0x07C	R/W	Endpoint 5 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	Descriptions	
[31:2]	**Reserved**	Reserved
[1]	**SSTALL**	**Set STALL** 1 = Set the device to respond STALL automatically 0 = Disable the device to response STALL
[0]	**CLRRDY**	**Clear Ready** When the MXPLD register is set by user, it means that the endpoint is ready to transmit or receive data. If the user wants to turn off this transaction before the transaction start, users can set this bit to 1 to turn it off and it is auto clear to 0. For IN token, write '1' is used to clear the IN token had ready to transmit the data to USB. For OUT token, write '1' is used to clear the OUT token had ready to receive the data from USB. This bit is write 1 only and it is always 0 when it was read back.

USB Drive SE0 Register (USB_DRVSE0)

Register	Offset	R/W	Description	Reset Value
USB_DRVSE0	USB_BA+0x090	R/W	Force USB PHY to drive SE0	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							**DRVSE0**

Bits	Descriptions	
[31:1]	**Reserved**	Reserved
[0]	**DRVSE0**	**Drive Single Ended Zero in USB Bus** The Single Ended Zero (SE0) is when both lines (USB_DP and USB_DM) are being pulled low. 1 = Force USB PHY transceiver to drive SE0 0 = None

5.5 General Purpose I/O (GPIO)

5.5.1 Overview

NuMicro™ NUC130/NUC140 has up to 80 General Purpose I/O pins can be shared with other function pins; it depends on the chip configuration. These 80 pins are arranged in 5 ports named with GPIOA, GPIOB, GPIOC, GPIOD and GPIOE. Each port equips maximum 16 pins. Each one of the 80 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, output, open-drain or quasi-bidirectional mode. After reset, the I/O type of all pins stay in quasi-bidirectional mode and port data register GPIOx_DOUT[15:0] resets to 0x0000_FFFF. Each I/O pin equips 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.

5.5.2 Features

- Four I/O modes:
  - Quasi bi-direction
  - Push-Pull output
  - Open-Drain output
  - Input only with high impendence
- TTL/Schmitt trigger input selectable
- I/O pin can be configured as interrupt source with edge/level setting
- High driver and high sink IO mode support

5.5.3 Function Description

5.5.3.1 Input Mode Explanation

Set GPIOx_PMD (PMDn[1:0]) to 00b the GPIOx port [n] pin is in Input mode and the I/O pin is in tri-state (high impedance) without output drive capability. The GPIOx_PIN value reflects the status of the corresponding port pins.

5.5.3.2 Output Mode Explanation

Set GPIOx_PMD (PMDn[1:0]) to 01b the GPIOx port [n] pin is in Output mode and the I/O pin supports digital output function with source/sink current capability. The bit value in the corresponding bit [n] of GPIOx_DOUT is driven on the pin.

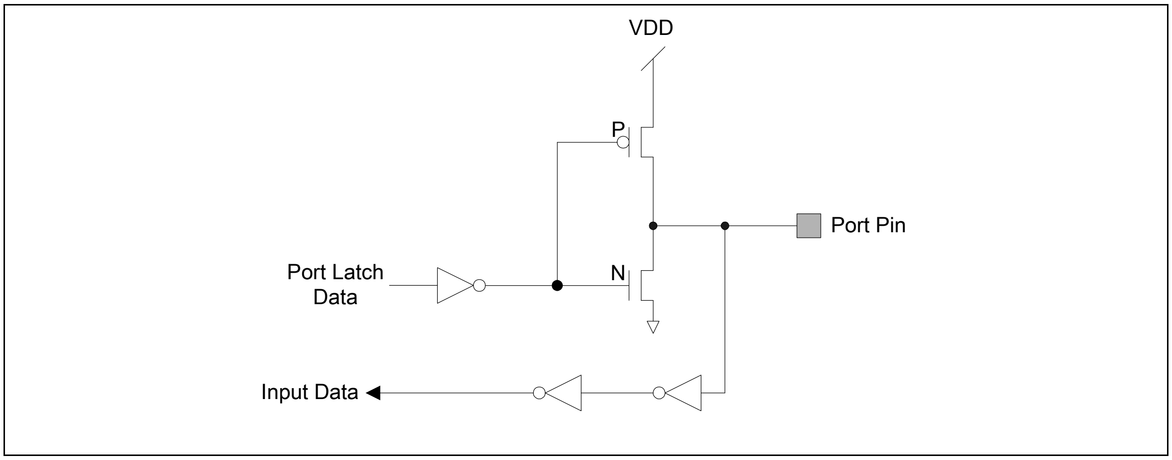


Figure 5-15 Push-Pull Output

5.5.3.3 Open-Drain Mode Explanation

Set GPIOx_PMD (PMDn[1:0]) to 10b the GPIOx port [n] pin is in Open-Drain mode and the digital output function of I/O pin supports only sink current capability, an additional pull-up register is needed for driving high state. If the bit value in the corresponding bit [n] of GPIOx_DOUT is 0, the pin drive a “low” output on the pin. If the bit value in the corresponding bit [n] of GPIOx_DOUT is 1, the pin output drives high that is controlled by external pull high resistor.

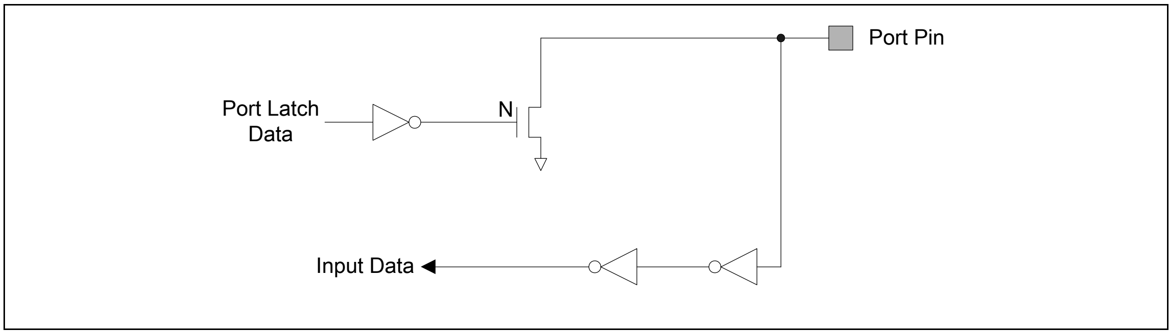


Figure 5-16 Open-Drain Output

5.5.3.4 Quasi-bidirectional Mode Explanation

Set GPIOx_PMD (PMDn[1:0]) to 11b the GPIOx port [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 bit in GPIOx_DOUT 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 bit [n] of GPIOx_DOUT is 0, the pin drive a “low” output on the pin. If the bit value in the corresponding bit [n] of GPIOx_DOUT is 1, the pin will check the pin value. If pin value is high, no action takes. If pin state is low, then pin will drive strong high with 2 clock cycles on the pin and then disable the strong output drive and then the pin status is control by internal pull-up resistor. Note that the source current capability in quasi-bidirectional mode is only about 200 uA to 30 uA for VDD is form 5.0 V to 2.5 V.

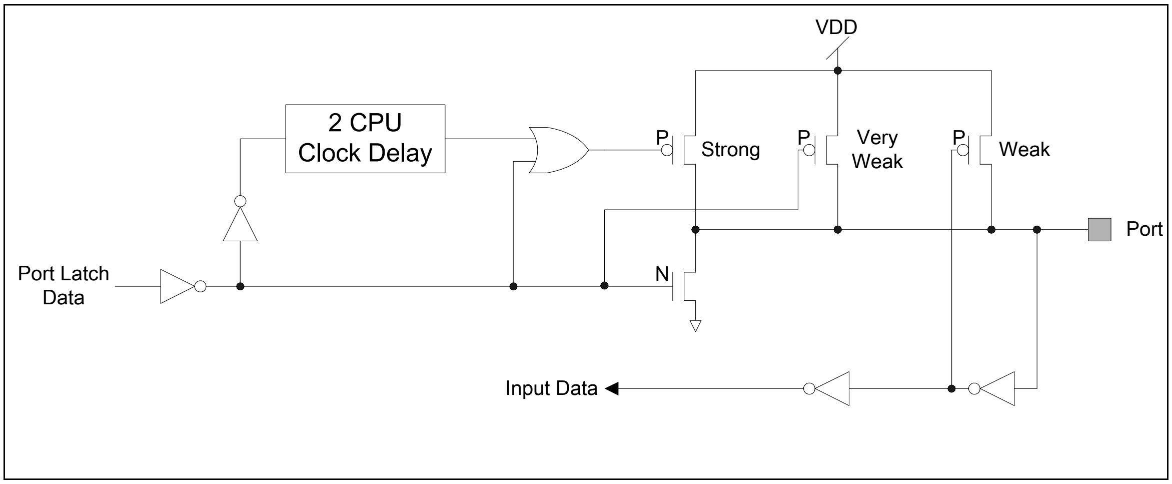


Figure 5-17 Quasi-bidirectional I/O Mode

5.5.4 Register Map

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

Register	Offset	R/W	Description	Reset Value
GP_BA = 0x5000_4000				
GPIOA_PMD	GP_BA+0x000	R/W	GPIO Port A Pin I/O Mode Control	0xFFFF_FFFF
GPIOA_OFFD	GP_BA+0x004	R/W	GPIO Port A Pin OFF Digital Enable	0x0000_0000
GPIOA_DOUT	GP_BA+0x008	R/W	GPIO Port A Data Output Value	0x0000_FFFF
GPIOA_DMASK	GP_BA+0x00C	R/W	GPIO Port A Data Output Write Mask	0x0000_0000
GPIOA_PIN	GP_BA+0x010	R	GPIO Port A Pin Value	0x0000_XXXX
GPIOA_DBEN	GP_BA+0x014	R/W	GPIO Port A De-bounce Enable	0x0000_0000
GPIOA_IMD	GP_BA+0x018	R/W	GPIO Port A Interrupt Mode Control	0x0000_0000
GPIOA_IEN	GP_BA+0x01C	R/W	GPIO Port A Interrupt Enable	0x0000_0000
GPIOA_ISRC	GP_BA+0x020	R/W	GPIO Port A Interrupt Source Flag	0xXXXX_XXXX
GPIOB_PMD	GP_BA+0x040	R/W	GPIO Port B Pin I/O Mode Control	0xFFFF_FFFF
GPIOB_OFFD	GP_BA+0x044	R/W	GPIO Port B Pin OFF Digital Enable	0x0000_0000
GPIOB_DOUT	GP_BA+0x048	R/W	GPIO Port B Data Output Value	0x0000_FFFF
GPIOB_DMASK	GP_BA+0x04C	R/W	GPIO Port B Data Output Write Mask	0x0000_0000
GPIOB_PIN	GP_BA+0x050	R	GPIO Port B Pin Value	0x0000_XXXX
GPIOB_DBEN	GP_BA+0x054	R/W	GPIO Port B De-bounce Enable	0x0000_0000
GPIOB_IMD	GP_BA+0x058	R/W	GPIO Port B Interrupt Mode Control	0x0000_0000
GPIOB_IEN	GP_BA+0x05C	R/W	GPIO Port B Interrupt Enable	0x0000_0000
GPIOB_ISRC	GP_BA+0x060	R/W	GPIO Port B Interrupt Source Flag	0xXXXX_XXXX
GPIOC_PMD	GP_BA+0x080	R/W	GPIO Port C Pin I/O Mode Control	0xFFFF_FFFF
GPIOC_OFFD	GP_BA+0x084	R/W	GPIO Port C Pin OFF digital Enable	0x0000_0000
GPIOC_DOUT	GP_BA+0x088	R/W	GPIO Port C Data Output Value	0x0000_FFFF
GPIOC_DMASK	GP_BA+0x08C	R/W	GPIO Port C Data Output Write Mask	0x0000_0000
GPIOC_PIN	GP_BA+0x090	R	GPIO Port C Pin Value	0x0000_XXXX
GPIOC_DBEN	GP_BA+0x094	R/W	GPIO Port C De-bounce Enable	0x0000_0000
GPIOC_IMD	GP_BA+0x098	R/W	GPIO Port C Interrupt Mode Control	0x0000_0000

Register	Offset	R/W	Description	Reset Value
GPIOC_IEN	GP_BA+0x09C	R/W	GPIO Port C Interrupt Enable	0x0000_0000
GPIOC_ISRC	GP_BA+0x0A0	R/W	GPIO Port C Interrupt Source Flag	0xXXXX_XXXX
GPIOD_PMD	GP_BA+0x0C0	R/W	GPIO Port D Pin I/O Mode Control	0xFFFF_FFFF
GPIOD_OFFD	GP_BA+0x0C4	R/W	GPIO Port D Pin OFF Digital Enable	0x0000_0000
GPIOD_DOUT	GP_BA+0x0C8	R/W	GPIO Port D Data Output Value	0x0000_FFFF
GPIOD_DMASK	GP_BA+0x0CC	R/W	GPIO Port D Data Output Write Mask	0x0000_0000
GPIOD_PIN	GP_BA+0x0D0	R	GPIO Port D Pin Value	0x0000_XXXX
GPIOD_DBEN	GP_BA+0x0D4	R/W	GPIO Port D De-bounce Enable	0x0000_0000
GPIOD_IMD	GP_BA+0x0D8	R/W	GPIO Port D Interrupt Mode Control	0x0000_0000
GPIOD_IEN	GP_BA+0x0DC	R/W	GPIO Port D Interrupt Enable	0x0000_0000
GPIOD_ISRC	GP_BA+0x0E0	R/W	GPIO Port D Interrupt Source Flag	0xXXXX_XXXX
GPIOE_PMD	GP_BA+0x100	R/W	GPIO Port E Pin I/O Mode Control	0xFFFF_FFFF
GPIOE_OFFD	GP_BA+0x104	R/W	GPIO Port E Pin OFF Digital Enable	0x0000_0000
GPIOE_DOUT	GP_BA+0x108	R/W	GPIO Port E Data Output Value	0x0000_FFFF
GPIOE_DMASK	GP_BA+0x10C	R/W	GPIO Port E Data Output Write Mask	0x0000_0000
GPIOE_PIN	GP_BA+0x110	R	GPIO Port E Pin Value	0x0000_XXXX
GPIOE_DBEN	GP_BA+0x114	R/W	GPIO Port E De-bounce Enable	0x0000_0000
GPIOE_IMD	GP_BA+0x118	R/W	GPIO Port E Interrupt Mode Control	0x0000_0000
GPIOE_IEN	GP_BA+0x11C	R/W	GPIO Port E Interrupt Enable	0x0000_0000
GPIOE_ISRC	GP_BA+0x120	R/W	GPIO Port E Interrupt Source Flag	0xXXXX_XXXX
DBNCECON	GP_BA+0x180	R/W	De-bounce Cycle Control	0x0000_0020
GPIOA0_DOUT	GP_BA+0x200	R/W	GPIO PA.0 Bit Output/Input Control	0x0000_0001
GPIOA1_DOUT	GP_BA+0x204	R/W	GPIO PA.1 Bit Output/Input Control	0x0000_0001
GPIOA2_DOUT	GP_BA+0x208	R/W	GPIO PA.2 Bit Output/Input Control	0x0000_0001
GPIOA3_DOUT	GP_BA+0x20C	R/W	GPIO PA.3 Bit Output/Input Control	0x0000_0001
GPIOA4_DOUT	GP_BA+0x210	R/W	GPIO PA.4 Bit Output/Input Control	0x0000_0001
GPIOA5_DOUT	GP_BA+0x214	R/W	GPIO PA.5 Bit Output/Input Control	0x0000_0001
GPIOA6_DOUT	GP_BA+0x218	R/W	GPIO PA.6 Bit Output/Input Control	0x0000_0001
GPIOA7_DOUT	GP_BA+0x21C	R/W	GPIO PA.7 Bit Output/Input Control	0x0000_0001

Register	Offset	R/W	Description	Reset Value
GPIOA8_DOUT	GP_BA+0x220	R/W	GPIO PA.8 Bit Output/Input Control	0x0000_0001
GPIOA9_DOUT	GP_BA+0x224	R/W	GPIO PA.9 Bit Output/Input Control	0x0000_0001
GPIOA10_DOUT	GP_BA+0x228	R/W	GPIO PA.10 Bit Output/Input Control	0x0000_0001
GPIOA11_DOUT	GP_BA+0x22C	R/W	GPIO PA.11 Bit Output/Input Control	0x0000_0001
GPIOA12_DOUT	GP_BA+0x230	R/W	GPIO PA.12 Bit Output/Input Control	0x0000_0001
GPIOA13_DOUT	GP_BA+0x234	R/W	GPIO PA.13 Bit Output/Input Control	0x0000_0001
GPIOA14_DOUT	GP_BA+0x238	R/W	GPIO PA.14 Bit Output/Input Control	0x0000_0001
GPIOA15_DOUT	GP_BA+0x23C	R/W	GPIO PA.15 Bit Output/Input Control	0x0000_0001
GPIOB0_DOUT	GP_BA+0x240	R/W	GPIO PB.0 Bit Output/Input Control	0x0000_0001
GPIOB1_DOUT	GP_BA+0x244	R/W	GPIO PB.1 Bit Output/Input Control	0x0000_0001
GPIOB2_DOUT	GP_BA+0x248	R/W	GPIO PB.2 Bit Output/Input Control	0x0000_0001
GPIOB3_DOUT	GP_BA+0x24C	R/W	GPIO PB.3 Bit Output/Input Control	0x0000_0001
GPIOB4_DOUT	GP_BA+0x250	R/W	GPIO PB.4 Bit Output/Input Control	0x0000_0001
GPIOB5_DOUT	GP_BA+0x254	R/W	GPIO PB.5 Bit Output/Input Control	0x0000_0001
GPIOB6_DOUT	GP_BA+0x258	R/W	GPIO PB.6 Bit Output/Input Control	0x0000_0001
GPIOB7_DOUT	GP_BA+0x25C	R/W	GPIO PB.7 Bit Output/Input Control	0x0000_0001
GPIOB8_DOUT	GP_BA+0x260	R/W	GPIO PB.8 Bit Output/Input Control	0x0000_0001
GPIOB9_DOUT	GP_BA+0x264	R/W	GPIO PB.9 Bit Output/Input Control	0x0000_0001
GPIOB10_DOUT	GP_BA+0x268	R/W	GPIO PB.10 Bit Output/Input Control	0x0000_0001
GPIOB11_DOUT	GP_BA+0x26C	R/W	GPIO PB.11 Bit Output/Input Control	0x0000_0001
GPIOB12_DOUT	GP_BA+0x270	R/W	GPIO PB.12 Bit Output/Input Control	0x0000_0001
GPIOB13_DOUT	GP_BA+0x274	R/W	GPIO PB.13 Bit Output/Input Control	0x0000_0001
GPIOB14_DOUT	GP_BA+0x278	R/W	GPIO PB.14 Bit Output/Input Control	0x0000_0001
GPIOB15_DOUT	GP_BA+0x27C	R/W	GPIO PB.15 Bit Output/Input Control	0x0000_0001
GPIOC0_DOUT	GP_BA+0x280	R/W	GPIO PC.0 Bit Output/Input Control	0x0000_0001
GPIOC1_DOUT	GP_BA+0x284	R/W	GPIO PC.1 Bit Output/Input Control	0x0000_0001
GPIOC2_DOUT	GP_BA+0x288	R/W	GPIO PC.2 Bit Output/Input Control	0x0000_0001
GPIOC3_DOUT	GP_BA+0x28C	R/W	GPIO PC.3 Bit Output/Input Control	0x0000_0001
GPIOC4_DOUT	GP_BA+0x290	R/W	GPIO PC.4 Bit Output/Input Control	0x0000_0001

Register	Offset	R/W	Description	Reset Value
GPIOC5_DOUT	GP_BA+0x294	R/W	GPIO PC.5 Bit Output/Input Control	0x0000_0001
GPIOC6_DOUT	GP_BA+0x298	R/W	GPIO PC.6 Bit Output/Input Control	0x0000_0001
GPIOC7_DOUT	GP_BA+0x29C	R/W	GPIO PC.7 Bit Output/Input Control	0x0000_0001
GPIOC8_DOUT	GP_BA+0x2A0	R/W	GPIO PC.8 Bit Output/Input Control	0x0000_0001
GPIOC9_DOUT	GP_BA+0x2A4	R/W	GPIO PC.9 Bit Output/Input Control	0x0000_0001
GPIOC10_DOUT	GP_BA+0x2A8	R/W	GPIO PC.10 Bit Output/Input Control	0x0000_0001
GPIOC11_DOUT	GP_BA+0x2AC	R/W	GPIO PC.11 Bit Output/Input Control	0x0000_0001
GPIOC12_DOUT	GP_BA+0x2B0	R/W	GPIO PC.12 Bit Output/Input Control	0x0000_0001
GPIOC13_DOUT	GP_BA+0x2B4	R/W	GPIO PC.13 Bit Output/Input Control	0x0000_0001
GPIOC14_DOUT	GP_BA+0x2B8	R/W	GPIO PC.14 Bit Output/Input Control	0x0000_0001
GPIOC15_DOUT	GP_BA+0x2BC	R/W	GPIO PC.15 Bit Output/Input Control	0x0000_0001
GPIOD0_DOUT	GP_BA+0x2C0	R/W	GPIO PD.0 Bit Output/Input Control	0x0000_0001
GPIOD1_DOUT	GP_BA+0x2C4	R/W	GPIO PD.1 Bit Output/Input Control	0x0000_0001
GPIOD2_DOUT	GP_BA+0x2C8	R/W	GPIO PD.2 Bit Output/Input Control	0x0000_0001
GPIOD3_DOUT	GP_BA+0x2CC	R/W	GPIO PD.3 Bit Output/Input Control	0x0000_0001
GPIOD4_DOUT	GP_BA+0x2D0	R/W	GPIO PD.4 Bit Output/Input Control	0x0000_0001
GPIOD5_DOUT	GP_BA+0x2D4	R/W	GPIO PD.5 Bit Output/Input Control	0x0000_0001
GPIOD6_DOUT	GP_BA+0x2D8	R/W	GPIO PD.6 Bit Output/Input Control	0x0000_0001
GPIOD7_DOUT	GP_BA+0x2DC	R/W	GPIO PD.7 Bit Output/Input Control	0x0000_0001
GPIOD8_DOUT	GP_BA+0x2E0	R/W	GPIO PD.8 Bit Output/Input Control	0x0000_0001
GPIOD9_DOUT	GP_BA+0x2E4	R/W	GPIO PD.9 Bit Output/Input Control	0x0000_0001
GPIOD10_DOUT	GP_BA+0x2E8	R/W	GPIO PD.10 Bit Output/Input Control	0x0000_0001
GPIOD11_DOUT	GP_BA+0x2EC	R/W	GPIO PD.11 Bit Output/Input Control	0x0000_0001
GPIOD12_DOUT	GP_BA+0x2F0	R/W	GPIO PD.12 Bit Output/Input Control	0x0000_0001
GPIOD13_DOUT	GP_BA+0x2F4	R/W	GPIO PD.13 Bit Output/Input Control	0x0000_0001
GPIOD14_DOUT	GP_BA+0x2F8	R/W	GPIO PD.14 Bit Output/Input Control	0x0000_0001
GPIOD15_DOUT	GP_BA+0x2FC	R/W	GPIO PD.15 Bit Output/Input Control	0x0000_0001
GPIOE0_DOUT	GP_BA+0x300	R/W	GPIO PE.0 Bit Output/Input Control	0x0000_0001
GPIOE1_DOUT	GP_BA+0x304	R/W	GPIO PE.1 Bit Output/Input Control	0x0000_0001

Register	Offset	R/W	Description	Reset Value
GPIOE2_DOUT	GP_BA+0x308	R/W	GPIO PE.2 Bit Output/Input Control	0x0000_0001
GPIOE3_DOUT	GP_BA+0x30C	R/W	GPIO PE.3 Bit Output/Input Control	0x0000_0001
GPIOE4_DOUT	GP_BA+0x310	R/W	GPIO PE.4 Bit Output/Input Control	0x0000_0001
GPIOE5_DOUT	GP_BA+0x314	R/W	GPIO PE.5 Bit Output/Input Control	0x0000_0001
GPIOE6_DOUT	GP_BA+0x318	R/W	GPIO PE.6 Bit Output/Input Control	0x0000_0001
GPIOE7_DOUT	GP_BA+0x31C	R/W	GPIO PE.7 Bit Output/Input Control	0x0000_0001
GPIOE8_DOUT	GP_BA+0x320	R/W	GPIO PE.8 Bit Output/Input Control	0x0000_0001
GPIOE9_DOUT	GP_BA+0x324	R/W	GPIO PE.9 Bit Output/Input Control	0x0000_0001
GPIOE10_DOUT	GP_BA+0x328	R/W	GPIO PE.10 Bit Output/Input Control	0x0000_0001
GPIOE11_DOUT	GP_BA+0x32C	R/W	GPIO PE.11 Bit Output/Input Control	0x0000_0001
GPIOE12_DOUT	GP_BA+0x330	R/W	GPIO PE.12 Bit Output/Input Control	0x0000_0001
GPIOE13_DOUT	GP_BA+0x334	R/W	GPIO PE.13 Bit Output/Input Control	0x0000_0001
GPIOE14_DOUT	GP_BA+0x338	R/W	GPIO PE.14 Bit Output/Input Control	0x0000_0001
GPIOE15_DOUT	GP_BA+0x33C	R/W	GPIO PE.15 Bit Output/Input Control	0x0000_0001

5.5.5 Register Description

GPIO Port [A/B/C/D/E] I/O Mode Control (GPIOx_PMD)

Register	Offset	R/W	Description	Reset Value
GPIOA_PMD	GP_BA+0x000	R/W	GPIO Port A Pin I/O Mode Control	0xFFFF_FFFF
GPIOB_PMD	GP_BA+0x040	R/W	GPIO Port B Pin I/O Mode Control	0xFFFF_FFFF
GPIOC_PMD	GP_BA+0x080	R/W	GPIO Port C Pin I/O Mode Control	0xFFFF_FFFF
GPIOD_PMD	GP_BA+0x0C0	R/W	GPIO Port D Pin I/O Mode Control	0xFFFF_FFFF
GPIOE_PMD	GP_BA+0x100	R/W	GPIO Port E Pin I/O Mode Control	0xFFFF_FFFF

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**
PMD15		**PMD14**		**PMD13**		**PMD12**	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PMD11		**PMD10**		**PMD9**		**PMD8**	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PMD7		**PMD6**		**PMD5**		**PMD4**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PMD3		**PMD2**		**PMD1**		**PMD0**	

Bits	Descriptions	
[2n+1:2n]	**PMDn**	**GPIOx I/O Pin[n] Mode Control** Determine each I/O type of GPIOx pins. 00 = GPIO port [n] pin is in INPUT mode 01 = GPIO port [n] pin is in OUTPUT mode 10 = GPIO port [n] pin is in Open-Drain mode 11 = GPIO port [n] pin is in Quasi-bidirectional mode

GPIO Port [A/B/C/D/E] Pin OFF Digital Resistor Enable (GPIOx_OFFD)

Register	Offset	R/W	Description	Reset Value
GPIOA_OFFD	GP_BA+0x004	R/W	GPIO Port A Pin OFF Digital Enable	0x0000_0000
GPIOB_OFFD	GP_BA+0x044	R/W	GPIO Port B Pin OFF Digital Enable	0x0000_0000
GPIOC_OFFD	GP_BA+0x084	R/W	GPIO Port C Pin OFF Digital Enable	0x0000_0000
GPIOD_OFFD	GP_BA+0x0C4	R/W	GPIO Port D Pin OFF Digital Enable	0x0000_0000
GPIOE_OFFD	GP_BA+0x104	R/W	GPIO Port E Pin OFF Digital Enable	0x0000_0000

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

Bits	Descriptions	
[16:31]	**OFFD**	**GPIOx Pin[n] OFF digital input path Enable** Each of these bits is used to control if the input path of corresponding GPIO pin is disabled. If input is analog signal, users can OFF digital input path to avoid creepage 1 = Disable IO digital input path (digital input tied to low) 0 = Enable IO digital input path
[0:15]	**Reserved**	Reserved

GPIO Port [A/B/C/D/E] Data Output Value (GPIOx_DOUT)

Register	Offset	R/W	Description	Reset Value
GPIOA_DOUT	GP_BA+0x008	R/W	GPIO Port A Data Output Value	0x0000_FFFF
GPIOB_DOUT	GP_BA+0x048	R/W	GPIO Port B Data Output Value	0x0000_FFFF
GPIOC_DOUT	GP_BA+0x088	R/W	GPIO Port C Data Output Value	0x0000_FFFF
GPIOD_DOUT	GP_BA+0x0C8	R/W	GPIO Port D Data Output Value	0x0000_FFFF
GPIOE_DOUT	GP_BA+0x108	R/W	GPIO Port E Data Output Value	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**
DOUT[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DOUT[7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	Reserved
[n]	**DOUT[n]**	**GPIOx Pin[n] Output Value** Each of these bits control the status of a GPIO pin when the GPIO pin is configures as output, open-drain and quasi-mode. 1 = GPIO port [A/B/C/D/E] Pin[n] will drive High if the GPIO pin is configures as output, open-drain and quasi-mode. 0 = GPIO port [A/B/C/D/E] Pin[n] will drive Low if the GPIO pin is configures as output, open-drain and quasi-mode.

GPIO Port [A/B/C/D/E] Data Output Write Mask (GPIOx _DMASK)

Register	Offset	R/W	Description	Reset Value
GPIOA_DMASK	GP_BA+0x00C	R/W	GPIO Port A Data Output Write Mask	0xXXXX_0000
GPIOB_DMASK	GP_BA+0x04C	R/W	GPIO Port B Data Output Write Mask	0xXXXX_0000
GPIOC_DMASK	GP_BA+0x08C	R/W	GPIO Port C Data Output Write Mask	0xXXXX_0000
GPIOD_DMASK	GP_BA+0x0CC	R/W	GPIO Port D Data Output Write Mask	0xXXXX_0000
GPIOE_DMASK	GP_BA+0x10C	R/W	GPIO Port E Data Output Write Mask	0xXXXX_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**
DMASK[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DMASK[7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	Reserved
[n]	**DMASK[n]**	**Port [A/B/C/D/E] Data Output Write Mask** These bits are used to protect the corresponding register of GPIOx_DOUT bit[n]. When set the DMASK bit[n] to 1, the corresponding GPIOx_DOUT[n] bit is protected. The write signal is masked, write data to the protect bit is ignored 1 = The corresponding GPIOx_DOUT[n] bit is protected 0 = The corresponding GPIOx_DOUT[n] bit can be updated Note: This function only protect corresponding GPIOx_DOUT[n] bit, and will not protect corresponding bit control register (GPIOAx_DOUT, GPIOBx_DOUT, GPIOCx_DOUT, GPIODx_DOUT, GPIOEx_DOUT).

GPIO Port [A/B/C/D/E] Pin Value (GPIOx _PIN)

Register	Offset	R/W	Description	Reset Value
GPIOA_PIN	GP_BA+0x010	R	GPIO Port A Pin Value	0x0000_XXXX
GPIOB_PIN	GP_BA+0x050	R	GPIO Port B Pin Value	0x0000_XXXX
GPIOC_PIN	GP_BA+0x090	R	GPIO Port C Pin Value	0x0000_XXXX
GPIOD_PIN	GP_BA+0x0D0	R	GPIO Port D Pin Value	0x0000_XXXX
GPIOE_PIN	GP_BA+0x110	R	GPIO Port E Pin Value	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**
PIN[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PIN[7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	Reserved
[n]	**PIN[n]**	**Port [A/B/C/D/E] Pin Values** Each bit of the register reflects the actual status of the respective GPIO pin If bit is 1, it indicates the corresponding pin status is high, else the pin status is low

GPIO Port [A/B/C/D/E] De-bounce Enable (GPIOx _DBEN)

Register	Offset	R/W	Description	Reset Value
GPIOA_DBEN	GP_BA+0x014	R/W	GPIO Port A De-bounce Enable	0xXXXX_0000
GPIOB_DBEN	GP_BA+0x054	R/W	GPIO Port B De-bounce Enable	0xXXXX_0000
GPIOC_DBEN	GP_BA+0x094	R/W	GPIO Port C De-bounce Enable	0xXXXX_0000
GPIOD_DBEN	GP_BA+0x0D4	R/W	GPIO Port D De-bounce Enable	0xXXXX_0000
GPIOE_DBEN	GP_BA+0x114	R/W	GPIO Port E De-bounce Enable	0xXXXX_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[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DBEN[7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	Reserved
[n]	**DBEN[n]**	**Port [A/B/C/D/E] Input Signal De-bounce Enable** DBEN[n]used to enable the de-bounce function for each corresponding bit. If the input signal pulse width can't 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 DBNCECON[4], one de-bounce sample cycle is controlled by DBNCECON[3:0] The DBEN[n] is used for "edge-trigger" interrupt only, and ignored for "level trigger" interrupt 1 = The bit[n] de-bounce function is enabled 0 = The bit[n] de-bounce function is disabled The de-bounce function is valid for edge triggered interrupt. If the interrupt mode is level triggered, the de-bounce enable bit is ignored.

GPIO Port [A/B/C/D/E] Interrupt Mode Control (GPIOx _IMD)

Register	Offset	R/W	Description	Reset Value
GPIOA_IMD	GP_BA+0x018	R/W	GPIO Port A Interrupt Mode Control	0xXXXX_0000
GPIOB_IMD	GP_BA+0x058	R/W	GPIO Port B Interrupt Mode Control	0xXXXX_0000
GPIOC_IMD	GP_BA+0x098	R/W	GPIO Port C Interrupt Mode Control	0xXXXX_0000
GPIOD_IMD	GP_BA+0x0D8	R/W	GPIO Port D Interrupt Mode Control	0xXXXX_0000
GPIOE_IMD	GP_BA+0x118	R/W	GPIO Port E Interrupt Mode Control	0xXXXX_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**
IMD[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IMD[7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	Reserved
[n]	**IMD[n]**	**Port [A/B/C/D/E] Edge or Level Detection Interrupt Control** IMD[n] is used to control the 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. 1 = Level trigger interrupt 0 = Edge trigger interrupt If set pin as the level trigger interrupt, then only one level can be set on the registers GPIOx_IEN. If set both the level to trigger interrupt, the setting is ignored and no interrupt will occur The de-bounce function is valid for edge triggered interrupt. If the interrupt mode is level triggered, the de-bounce enable bit is ignored.

GPIO Port [A/B/C/D] Interrupt Enable Control (GPIOx _IEN)

Register	Offset	R/W	Description	Reset Value
GPIOA_IEN	GP_BA+0x01C	R/W	GPIO Port A Interrupt Enable	0x0000_0000
GPIOB_IEN	GP_BA+0x05C	R/W	GPIO Port B Interrupt Enable	0x0000_0000
GPIOC_IEN	GP_BA+0x09C	R/W	GPIO Port C Interrupt Enable	0x0000_0000
GPIOD_IEN	GP_BA+0x0DC	R/W	GPIO Port D Interrupt Enable	0x0000_0000
GPIOE_IEN	GP_BA+0x11C	R/W	GPIO Port E Interrupt Enable	0x0000_0000

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

Bits	Descriptions	
[n+16]	**IR_EN[n]**	**Port [A/B/C/D/E] Interrupt Enable by Input Rising Edge or Input Level High** IR_EN[n] used to enable the interrupt for each of the corresponding input GPIO_PIN[n]. Set bit to 1 also enable the pin wake-up function When set the IR_EN[n] bit to 1: If the interrupt is level trigger, the input PIN[n] state at level “high” will generate the interrupt. If the interrupt is edge trigger, the input PIN[n] state change from “low-to-high” will generate the interrupt. 1 = Enable the PIN[n] level-high or low-to-high interrupt 0 = Disable the PIN[n] level-high or low-to-high interrupt
[n]	**IF_EN[n]**	**Port [A/B/C/D/E] Interrupt Enable by Input Falling Edge or Input Level Low** IF_EN[n] used to enable the interrupt for each of the corresponding input GPIO_PIN[n]. Set bit to 1 also enable the pin wake-up function When set the IF_EN[n] bit to 1: If the interrupt is level trigger, the input PIN[n] state at level “low” will generate the interrupt. If the interrupt is edge trigger, the input PIN[n] state change from “high-to-low” will generate the interrupt. 1 = Enable the PIN[n] state low-level or high-to-low change interrupt 0 = Disable the PIN[n] state low-level or high-to-low change interrupt

GPIO Port [A/B/C/D/E] Interrupt Trigger Source (GPIOx _ISRC)

Register	Offset	R/W	Description	Reset Value
GPIOA_ISRC	GP_BA+0x020	R/W	GPIO Port A Interrupt Trigger Source Indicator	0x0000_0000
GPIOB_ISRC	GP_BA+0x060	R/W	GPIO Port B Interrupt Trigger Source Indicator	0x0000_0000
GPIOC_ISRC	GP_BA+0x0A0	R/W	GPIO Port C Interrupt Trigger Source Indicator	0x0000_0000
GPIOD_ISRC	GP_BA+0x0E0	R/W	GPIO Port D Interrupt Trigger Source Indicator	0x0000_0000
GPIOE_ISRC	GP_BA+0x120	R/W	GPIO Port E Interrupt Trigger Source Indicator	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**
IF_ ISRC[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IF_ ISRC[7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	Reserved
[n]	**ISRC[n]**	**Port [A/B/C/D/E] Interrupt Trigger Source Indicator** Read : 1 = Indicates GPIOx[n] generate an interrupt 0 = No interrupt at GPIOx[n] Write : 1= Clear the correspond pending interrupt 0= No action

Interrupt De-bounce Cycle Control (DBNCECON)

Register	Offset	R/W	Description	Reset Value
DBNCECON	GP_BA+0x180	R/W	External Interrupt De-bounce Control	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		**ICLK_ON**	**DBCLKSRC**	**DBCLKSEL**			

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[5]</td><td>ICLK_ON</td><td>Interrupt clock On mode
Set this bit to 0 will disable the interrupt generate circuit clock, if the pin[n] interrupt is disabled
1 = Interrupt generated circuit clock always enable
0 = Disable the clock if the GPIOA/B/C/D/E[n] interrupt is disabled</td></tr>
<tr><td>[4]</td><td>DBCLKSRC</td><td>De-bounce counter clock source select
1 = De-bounce counter clock source is the internal 10 kHz low speed oscillator
0 = De-bounce counter clock source is the HCLK</td></tr>
<tr><td>[3:0]</td><td>DBCLKSEL</td><td>De-bounce sampling cycle selection
<table>
<tr><th>DBCLKSEL</th><th>Description</th></tr>
<tr><td>0</td><td>Sample interrupt input once per 1 clocks</td></tr>
<tr><td>1</td><td>Sample interrupt input once per 2 clocks</td></tr>
<tr><td>2</td><td>Sample interrupt input once per 4 clocks</td></tr>
<tr><td>3</td><td>Sample interrupt input once per 8 clocks</td></tr>
<tr><td>4</td><td>Sample interrupt input once per 16 clocks</td></tr>
<tr><td>5</td><td>Sample interrupt input once per 32 clocks</td></tr>
<tr><td>6</td><td>Sample interrupt input once per 64 clocks</td></tr>
<tr><td>7</td><td>Sample interrupt input once per 128 clocks</td></tr>
<tr><td>8</td><td>Sample interrupt input once per 256 clocks</td></tr>
<tr><td>9</td><td>Sample interrupt input once per 2*256 clocks</td></tr>
<tr><td>10</td><td>Sample interrupt input once per 4*256clocks</td></tr>
</table></td></tr>
</table>

11	Sample interrupt input once per 8*256 clocks
12	Sample interrupt input once per 16*256 clocks
13	Sample interrupt input once per 32*256 clocks
14	Sample interrupt input once per 64*256 clocks
15	Sample interrupt input once per 128*256 clocks

GPIO Port [A/B/C/D/E] I/O Bit Output/Input Control (GPIOxx_DOUT)

Register	Offset	R/W	Description	Reset Value
GPIOAx_DOUT	GP_BA+0x200 - GP_BA+0x23C	R/W	GPIO Port A Pin I/O Bit Output/Input Control	0x0000_0001
GPIOBx_DOUT	GP_BA+0x240 - GP_BA+0x27C	R/W	GPIO Port B Pin I/O Bit Output/Input Control	0x0000_0001
GPIOCx_DOUT	GP_BA+0x280 - GP_BA+0x2BC	R/W	GPIO Port C Pin I/O Bit Output/Input Control	0x0000_0001
GPIODx_DOUT	GP_BA+0x2C0 - GP_BA+0x2FC	R/W	GPIO Port D Pin I/O Bit Output/Input Control	0x0000_0001
GPIOEx_DOUT	GP_BA+0x300 - GP_BA+0x3FC	R/W	GPIO Port E Pin I/O Bit Output/Input Control	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							**GPIOxx_DOUT**

Bits	Descriptions	
[0]	**GPIOxx_DOUT**	**GPIOxx I/O Pin Bit Output/Input Control** Write this bit can control one GPIO pin output value 1 = Set corresponding GPIO pin to high 0 = Set corresponding GPIO pin to low Read this register to get IO pin status. For example: write GPIOA0_DOUT will reflect the written value to bit GPIOA_DOUT[0], read GPIOA0_DOUT will return the value of GPIOA_PIN[0]

5.6 I²C Serial Interface Controller (Master/Slave) (I²C)

5.6.1 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.

Data is transferred between a Master and a Slave synchronously to SCL on the SDA line 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. 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 5-18 for more detail I²C BUS Timing.

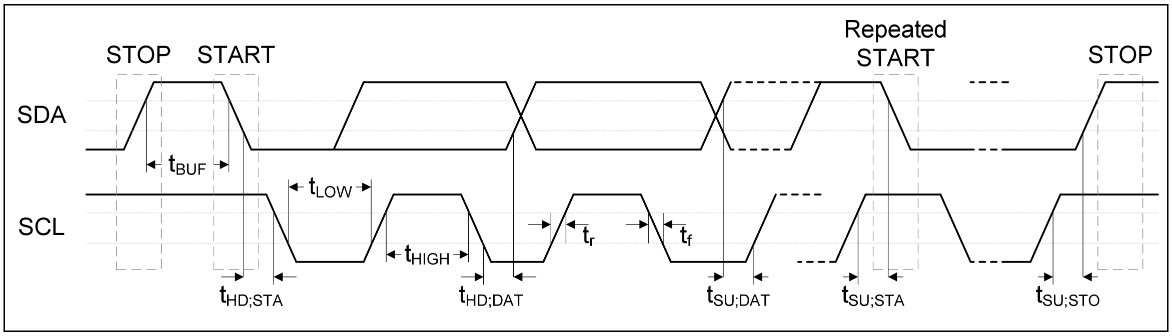


Figure 5-18 I²C Bus Timing

The device’s on-chip I²C logic 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 ENS1 in I2CON should be set to '1'. The I²C H/W interfaces to the I²C bus via two pins: SDA and SCL. Pull up resistor is needed for I²C operation as these are open drain pins. When the I/O pins are used as I²C port, user must set the pins function to I²C in advance.

5.6.2 Features

The I²C bus uses two wires (SDA and SCL) to transfer information between devices connected to the bus. The main features of the bus are:

- 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
- Built-in a 14-bit time-out counter will request the I^2C interrupt if the I^2C bus hangs up and timer-out counter overflows.
- External pull-up are needed for high output
- Programmable clocks allow versatile rate control
- Supports 7-bit addressing mode
- I^2C-bus controllers support multiple address recognition (Four slave address with mask option)

5.6.3 Function Description

5.6.3.1 I²C Protocol

Normally, a standard communication consists of four parts:

1) START or Repeated START signal generation
2) Slave address and R/W bit transfer
3) Data transfer
4) STOP signal generation

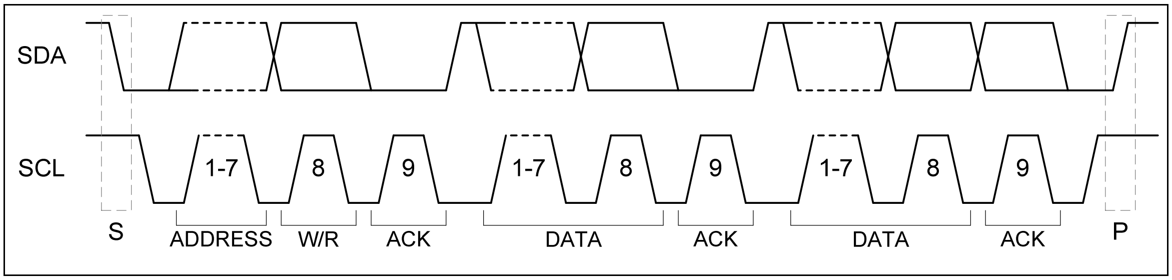


Figure 5-19 I²C Protocol

5.6.3.2 Data transfer on the I²C-bus

A master-transmitter addressing a slave receiver with a 7-bit address

The transfer direction is not changed

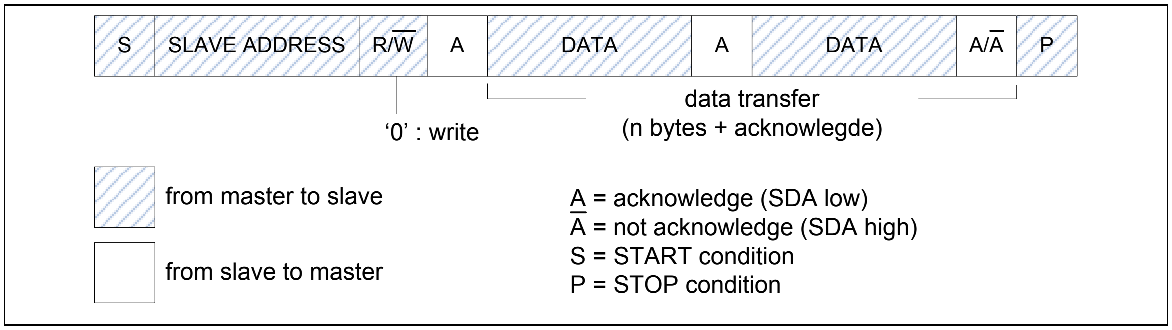


Figure 5-20 Master Transmits Data to Slave

A master reads a slave immediately after the first byte (address)

The transfer direction is changed

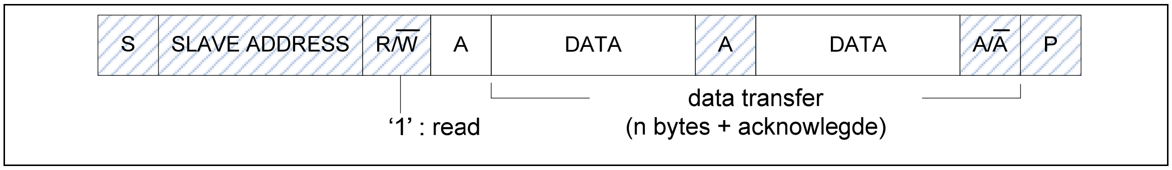


Figure 5-21 Master Reads Data from Slave

5.6.3.3 START or Repeated START signal

When the bus is free/idle, meaning 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 transfer.

A Repeated START (Sr) is no STOP signal between two START signals. The master 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.

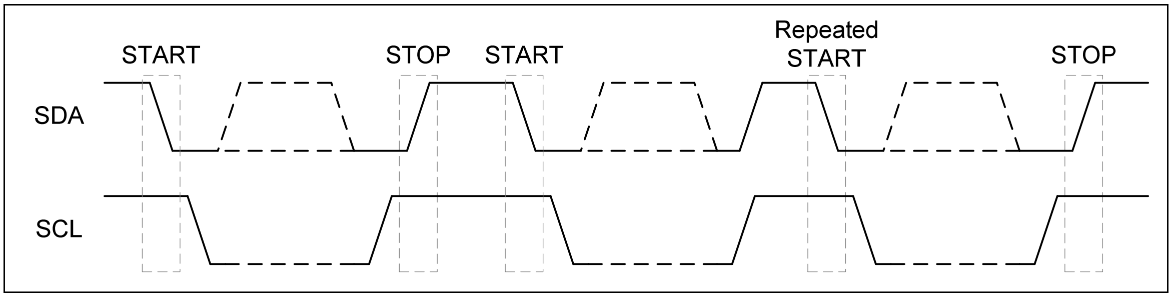


Figure 5-22 START and STOP condition

5.6.3.4 Slave Address Transfer

The first byte of data transferred by the master immediately after the START signal is the slave address. This is a 7-bit calling address followed by a RW bit. The RW bit signals the slave 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.

5.6.3.5 Data Transfer

Once successful slave addressing has been achieved, the data transfer can proceed on a byte-by-byte basis in the direction specified by the RW bit sent by the master. 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 the 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.

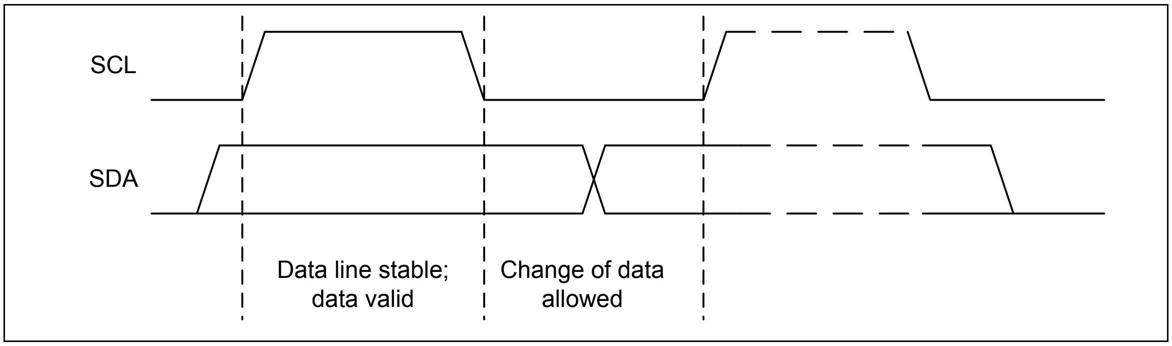


Figure 5-23 Bit Transfer on the I^2C bus

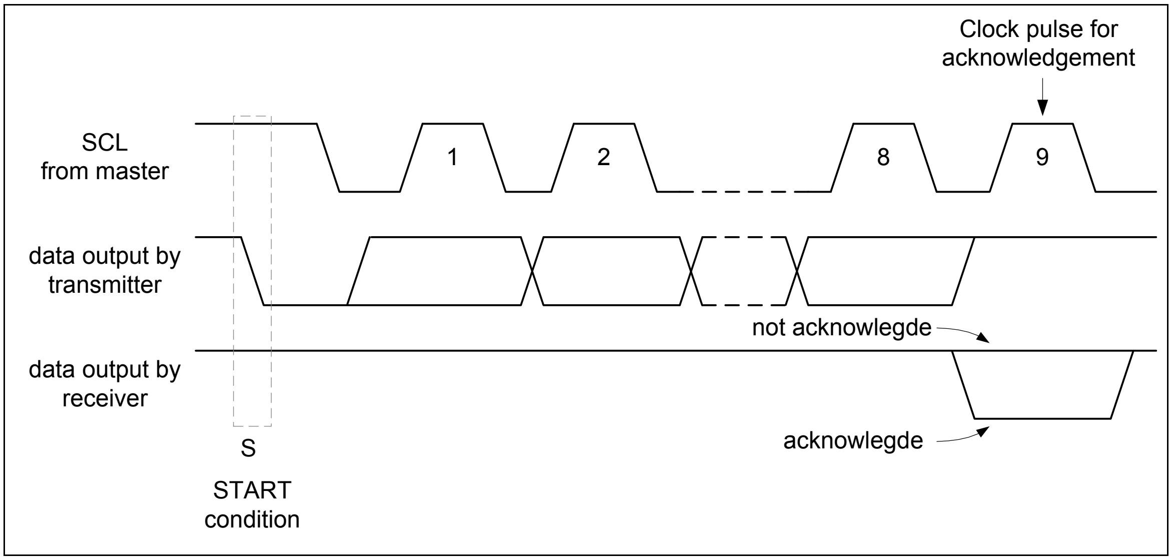


Figure 5-24 Acknowledge on the I^2C bus

5.6.4 Protocol Registers

The CPU interfaces to the I²C port through the following thirteen special function registers: I2CON (control register), I2CSTATUS (status register), I2CDAT (data register), I2CADDRn (address registers, n=0~3), I2CADMn (address mask registers, n=0~3), I2CLK (clock rate register) and I2CTOC (Time-out counter register). All bit 31~ bit 8 of these I²C special function registers are reserved. These bits do not have any functions and are all zero if read back.

When I²C port is enabled by setting ENS1 (I2CON [6]) to high, the internal states will be controlled by I2CON and I²C logic hardware. Once a new status code is generated and stored in I2CSTATUS, the I^{2}C Interrupt Flag bit SI (I2CON [3]) will be set automatically. If the Enable Interrupt bit EI (I2CON [7]) is set high at this time, the I^{2}C interrupt will be generated. The bit field I2CSTATUS[7:3] stores the internal state code, the lowest 3 bits of I2CSTATUS are always zero and the content keeps stable until SI is cleared by software. The base address is 4002_0000 and 4012_0000.

5.6.4.1 Address Registers (I2CADDR)

I^{2}C port is equipped with four slave address registers I2CADDRn (n=0~3). The contents of the register are irrelevant when I^{2}C is in master mode. In the slave mode, the bit field I2CADDRn[7:1] must be loaded with the chip's own slave address. The I^{2}C hardware will react if the contents of I2CADDRn are matched with the received slave address.

The I^{2}C ports support the "General Call" function. If the GC bit (I2CADDRn [0]) is set the I^{2}C port hardware will respond to General Call address (00H). Clear GC bit to disable general call function.

When GC bit is set and the I^{2}C is in Slave mode, it can receive the general call address by 00H after Master send general call address to I^{2}C bus, then it will follow status of GC mode.

I^{2}C bus controllers support multiple address recognition with four address mask registers I2CADMn (n=0~3). 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.

5.6.4.2 Data Register (I2CDAT)

This register contains a byte of serial data to be transmitted or a byte which just has been received. The CPU can read from or write to this 8-bit (I2CDAT [7:0]) directly while it is not in the process of shifting a byte. when I^{2}C is in a defined state and the serial interrupt flag (SI) is set. Data in I2CDAT [7:0] remains stable as long as SI bit is set. While data is being shifted out, data on the bus is simultaneously being shifted in; I2CDAT [7:0] always contains the last data byte present on the bus. Thus, in the event of arbitration lost, the transition from master transmitter to slave receiver is made with the correct data in I2CDAT [7:0].

I2CDAT [7:0] and the acknowledge bit form a 9-bit shift register, the acknowledge bit is controlled by the I^{2}C hardware and cannot be accessed by the CPU. Serial data is shifted through the acknowledge bit into I2CDAT [7:0] on the rising edges of serial clock pulses on the SCL line. When a byte has been shifted into I2CDAT [7:0], the serial data is available in I2CDAT [7:0], and the acknowledge bit (ACK or NACK) is returned by the control logic during the ninth clock pulse. Serial data is shifted out from I2CDAT [7:0] on the falling edges of SCL clock pulses, and is shifted into I2CDAT [7:0] on the rising edges of SCL clock pulses.

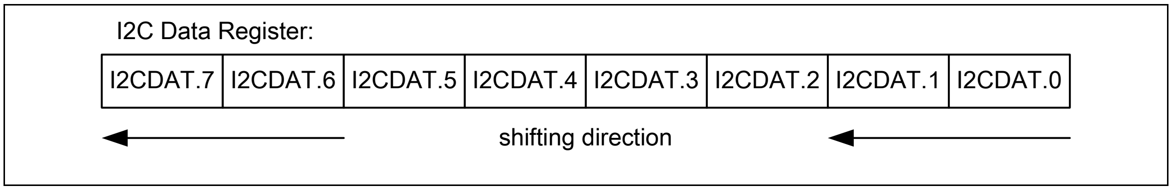


Figure 5-25 I^{2}C Data Shifting Direction

5.6.4.3 Control Register (I2CON)

The CPU can read from and write to this 8-bit field of I2CON [7:0] directly. Two bits are affected by hardware: the SI bit is set when the I^{2}C 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 ENS1 = 0.

EI	Enable Interrupt.
ENS1	Set to enable I^{2}C serial function controller. When ENS1=1 the I^{2}C serial function enables. The Multi Function pin function of SDA and SCL must be set to I^{2}C function.
STA	I^{2}C START Control Bit. Setting STA to logic 1 to enter master mode, the I^{2}C hardware sends a START or repeat START condition to bus when the bus is free.
STO	I^{2}C STOP Control Bit. In master mode, setting STO to transmit a STOP condition to bus then I^{2}C hardware will check the bus condition if a STOP condition is detected this flag will be cleared by hardware automatically. In a slave mode, setting STO resets I^{2}C hardware to the defined "not addressed" slave mode. This means it is NO LONGER in the slave receiver mode to receive data from the master transmit device.
SI	I^{2}C Interrupt Flag. When a new I^{2}C state is present in the I2CSTATUS register, the SI flag is set by hardware, and if bit EI (I2CON [7]) is set, the I^{2}C interrupt is requested. SI must be cleared by software. Clear SI is by writing 1 to this bit. All states are listed in section 5.6.6
AA	Assert Acknowledge Control Bit. When AA=1 prior to address or data 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.

5.6.4.4 Status Register (I2CSTATUS)

I2CSTATUS [7:0] is an 8-bit read-only register. The three least significant bits are always 0. The bit field I2CSTATUS [7:3] contain the status code. There are 26 possible status codes, All states are listed in section 5.6.6. When I2CSTATUS [7:0] contains F8H, no serial interrupt is requested. All other I2CSTATUS [7:3] values correspond to defined I^{2}C states. When each of these states is entered, a status interrupt is requested (SI = 1). A valid status code is present in I2CSTATUS[7:3] one cycle after SI is set by hardware and is still present one cycle after SI has been reset by software.

In addition, state 00H stands for a Bus Error. A Bus Error occurs when a START or STOP condition is present at an illegal position in the format frame. Examples of illegal positions are during the serial transfer of an address byte, a data byte or an acknowledge bit. To recover I^{2}C from bus error, STO should be set and SI should be clear to enter not addressed slave mode. Then clear STO to release bus and to wait new communication. I^{2}C bus can not recognize stop condition during this action when bus error occurs.

5.6.4.5 *I^2C Clock Baud Rate Bits (I2CLK)*

The data baud rate of I^2C is determines by I2CLK [7:0] register when I^2C is in a master mode. It is not important when I^2C is in a slave mode. In the slave modes, I^2C will automatically synchronize 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 (I2CLK [7:0] +1)). If system clock = 16 MHz, the I2CLK [7:0] = 40 (28H), so data baud rate of I^2C = 16 MHz/ (4x (40 +1)) = 97.5 Kbits/sec.

5.6.4.6 *The I^2C Time-out Counter Register (I2CTOC)*

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 (TIF=1) and generates I^2C interrupt to CPU or stops counting by clearing ENTI to 0. When time-out counter is enabled, setting flag SI to high will reset counter and re-start up counting after SI is cleared. If I^2C bus hangs up, it causes the I2CSTATUS 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 5-26 for the 14-bit time-out counter. User may write 1 to clear TIF to zero.

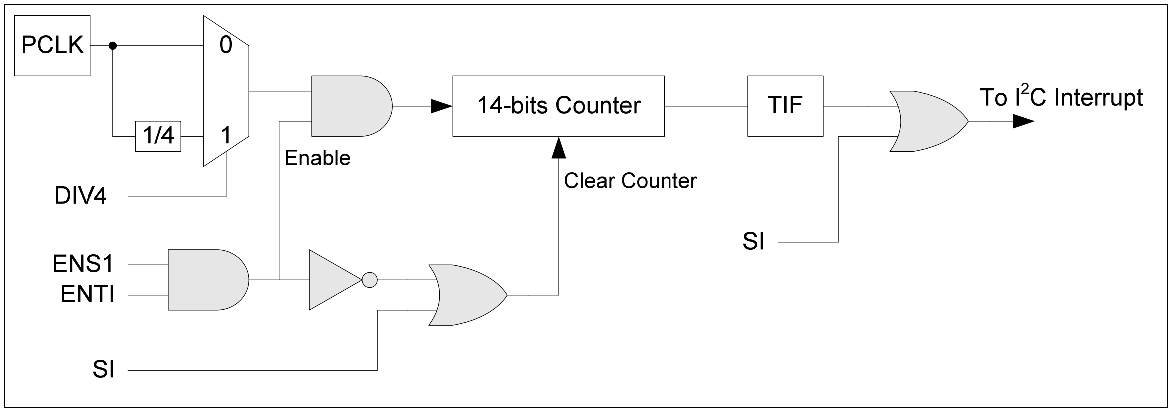


Figure 5-26: I^2C Time-out Count Block Diagram

5.6.5 Register Map

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

Register	Offset	R/W	Description	Reset Value
I2C0_BA = 0x4002_0000 **I2C1_BA = 0x4012_0000**				
I2CON	I2Cx_BA+0x00	R/W	I^{2}C Control Register	0x0000_0000
I2CADDR0	I2Cx_BA+0x04	R/W	I^{2}C Slave Address Register0	0x0000_0000
I2CDAT	I2Cx_BA+0x08	R/W	I^{2}C DATA Register	0x0000_0000
I2CSTATUS	I2Cx_BA+0x0C	R	I^{2}C Status Register	0x0000_00F8
I2CLK	I2Cx_BA+0x10	R/W	I^{2}C Clock Divided Register	0x0000_0000
I2CTOC	I2Cx_BA+0x14	R/W	I^{2}C Time Out Control Register	0x0000_0000
I2CADDR1	I2Cx_BA+0x18	R/W	I^{2}C Slave Address Register1	0x0000_0000
I2CADDR2	I2Cx_BA+0x1C	R/W	I^{2}C Slave Address Register2	0x0000_0000
I2CADDR3	I2Cx_BA+0x20	R/W	I^{2}C Slave Address Register3	0x0000_0000
I2CADM0	I2Cx_BA+0x24	R/W	I^{2}C Slave Address Mask Register0	0x0000_0000
I2CADM1	I2Cx_BA+0x28	R/W	I^{2}C Slave Address Mask Register1	0x0000_0000
I2CADM2	I2Cx_BA+0x2C	R/W	I^{2}C Slave Address Mask Register2	0x0000_0000
I2CADM3	I2Cx_BA+0x30	R/W	I^{2}C Slave Address Mask Register3	0x0000_0000

5.6.6 Register Description

I²C Control Register (I2CON)

Register	Offset	R/W	Description	Reset Value
I2CON	I2C_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**
EI	**ENS1**	**STA**	**STO**	**SI**	**AA**	**Reserved**	

Bits	Descriptions	
[31:8]	**Reserved**	Reserved
[7]	**EI**	**Enable Interrupt** 1 = Enable I²C interrupt 0 = Disable I²C interrupt
[6]	**ENS1**	**I²C Controller Enable Bit** 1 = Enable 0 = Disable Set to enable I²C serial function controller. When ENS1=1 the I²C serial function enables. The multi-function pin function of SDA and SCL must set to I²C function first.
[5]	**STA**	**I²C START Control Bit** 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 Bit** In master mode, setting STO to transmit a STOP condition to bus then I²C hardware will check the bus condition if a STOP condition is detected this bit will be cleared by hardware automatically. In a slave mode, setting STO resets I²C hardware to the defined “not addressed” slave mode. This means it is NO LONGER in the slave receiver mode to receive data from the master transmit device.
[3]	**SI**	**I²C Interrupt Flag** When a new I²C state is present in the I2CSTATUS register, the SI flag is set by hardware, and if bit EI (I2CON [7]) is set, the I²C interrupt is requested. SI must be cleared by software. Clear SI is by writing 1 to this bit.

[2]	**AA**	**Assert Acknowledge Control Bit** When AA=1 prior to address or data 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

I^{2}C Data Register (I2CDAT)

Register	Offset	R/W	Description	Reset Value
I2CDAT	I2C_BA+0x08	R/W	I^{2}C 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**
I2CDAT[7:0]							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved
[7:0]	**I2CDAT**	**I^{2}C Data Register** Bit [7:0] is located with the 8-bit transferred data of I^{2}C serial port.

I^{2}C Status Register (I2CSTATUS)

Register	Offset	R/W	Description	Reset Value
I2CSTATUS	I2C_BA+0x0C	R/W	I^{2}C 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**
I2CSTATUS[7:3]					**0**	**0**	**0**

Bits	Descriptions	
[31:8]	**Reserved**	Reserved
[7:0]	**I2CSTATUS**	**I^{2}C Status Register** The status register of I^{2}C: The three least significant bits are always 0. The five most significant bits contain the status code. There are 26 possible status codes. When I2CSTATUS contains F8H, no serial interrupt is requested. All other I2CSTATUS values correspond to defined I^{2}C states. When each of these states is entered, a status interrupt is requested (SI = 1). A valid status code is present in I2CSTATUS 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.

I^2C Clock Divided Register (I2CLK)

Register	Offset	R/W	Description	Reset Value
I2CLK	I2C_BA+0x10	R/W	I^2C 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**
I2CLK[7:0]							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved
[7:0]	**I2CLK**	**I^2C clock divided Register** The I^2C clock rate bits: Data Baud Rate of I^2C = (system clock) / (4x (I2CLK+1)).

I^{2}C Time-Out Counter Register (I2CTOC)

Register	Offset	R/W	Description	Reset Value
I2CTOC	I2C_BA+0x14	R/W	I^{2}C Time-Out 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					**ENTI**	**DIV4**	**TIF**

Bits	Descriptions	
[31:3]	**Reserved**	Reserved
[2]	**ENTI**	**Time-out counter is enabled/disable** 1 = Enable 0 = Disable When Enable, the 14-bit time-out counter will start counting when SI is clear. Setting flag SI to high will reset counter and re-start up counting after SI is cleared.
[1]	**DIV4**	**Time-Out counter input clock is divided by 4** 1 = Enable 0 = Disable When Enable, The time-Out period is extend 4 times.
[0]	**TIF**	**Time-Out Flag** This bit is set by H/W when I^{2}C time-out happened and it can interrupt CPU if I^{2}C interrupt enable bit (EI) is set to 1. S/W can write 1 to clear this bit.

I^{2}C Slave Address Register (I2CADDRx)

Register	Offset	R/W	Description	Reset Value
I2CADDR0	I2C_BA+0x04	R/W	I^{2}C Slave Address Register0	0x0000_0000
I2CADDR1	I2C_BA+0x18	R/W	I^{2}C Slave Address Register1	0x0000_0000
I2CADDR2	I2C_BA+0x1C	R/W	I^{2}C Slave Address Register2	0x0000_0000
I2CADDR3	I2C_BA+0x20	R/W	I^{2}C 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**
I2CADDR[7:1]							**GC**

Bits	Descriptions	
[31:8]	**Reserved**	Reserved
[7:1]	**I2CADDR**	**I^{2}C Address Register** The content of this register is irrelevant when I^{2}C is in master mode. In the slave mode, the seven most significant bits must be loaded with the chip's own address. The I^{2}C hardware will react if either of the address is matched.
[0]	**GC**	**General Call Function** 0 = Disable General Call Function. 1 = Enable General Call Function.

I^2C Slave Address Mask Register (I2CADMx)

Register	Offset	R/W	Description	Reset Value
I2CADM0	I2C_BA+0x24	R/W	I^2C Slave Address Mask Register0	0x0000_0000
I2CADM1	I2C_BA+0x28	R/W	I^2C Slave Address Mask Register1	0x0000_0000
I2CADM2	I2C_BA+0x2C	R/W	I^2C Slave Address Mask Register2	0x0000_0000
I2CADM3	I2C_BA+0x30	R/W	I^2C 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**
I2CADM[7:1]							**Reserved**

Bits	Descriptions	
[31:8]	**Reserved**	Reserved
[7:1]	**I2CADM**	**I^2C Address Mask register** 1 = Mask enable (the received corresponding address bit is don't care.) 0 = Mask disable (the received corresponding register bit should be exact the same as address register.) I^2C 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

5.6.7 Modes of Operation

The on-chip I^{2}C ports support five operation modes, Master transmitter, Master receiver, Slave transmitter, Slave receiver, and GC call.

In a given application, I^{2}C port may operate as a master or as a slave. In the slave mode, the I^{2}C 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, the hardware waits until the bus is free before the master mode is entered so that a possible slave action didn't be interrupted. If bus arbitration is lost in the master mode, I^{2}C port switches to the slave mode immediately and can detect its own slave address in the same serial transfer.

5.6.7.1 Master Transmitter Mode

Serial data output through SDA while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7-bit) and the data direction bit. In this case the data direction bit (R/W) will be logic 0, and it is represented by "W" in the flow diagrams. Thus the first byte transmitted is SLA+W. Serial data is transmitted 8-bit at a time. After each byte is transmitted, an acknowledge bit is received. START and STOP conditions are output to indicate the beginning and the end of a serial transfer.

5.6.7.2 Master Receiver Mode

In this case the data direction bit (R/W) will be logic 1, and it is represented by "R" in the flow diagrams. Thus the first byte transmitted is SLA+R. Serial data is received via SDA while SCL outputs the serial clock. Serial data is received 8-bit at a time. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are output to indicate the beginning and end of a serial transfer.

5.6.7.3 Slave Receiver Mode

Serial data and the serial clock are received through SDA and SCL. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and direction bit.

5.6.7.4 Slave Transmitter Mode

The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will indicate that the transfer direction is reversed. Serial data is transmitted via SDA while the serial clock is input through SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer.

5.6.8 Data Transfer Flow in Five Operating Modes

The five operating modes are: Master/Transmitter, Master/Receiver, Slave/Transmitter, Slave/Receiver and GC Call. Bits STA, STO and AA in I2CON register will determine the next state of the I^2C hardware after SI flag is cleared. Upon completion of the new action, a new status code will be updated and the SI flag will be set. If the I2C interrupt control bit EI (I2CON [7]) is set, appropriate action or software branch of the new status code can be performed in the Interrupt service routine.

Data transfers in each mode are shown in the Figure 5-27 to Figure 5-32.

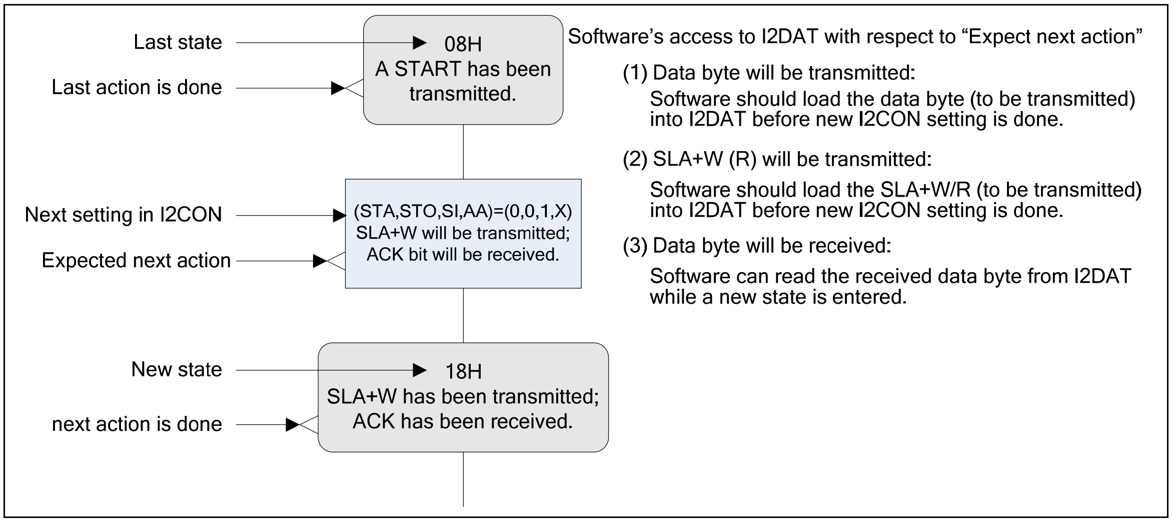


Figure 5-27 Legend for the following five figures

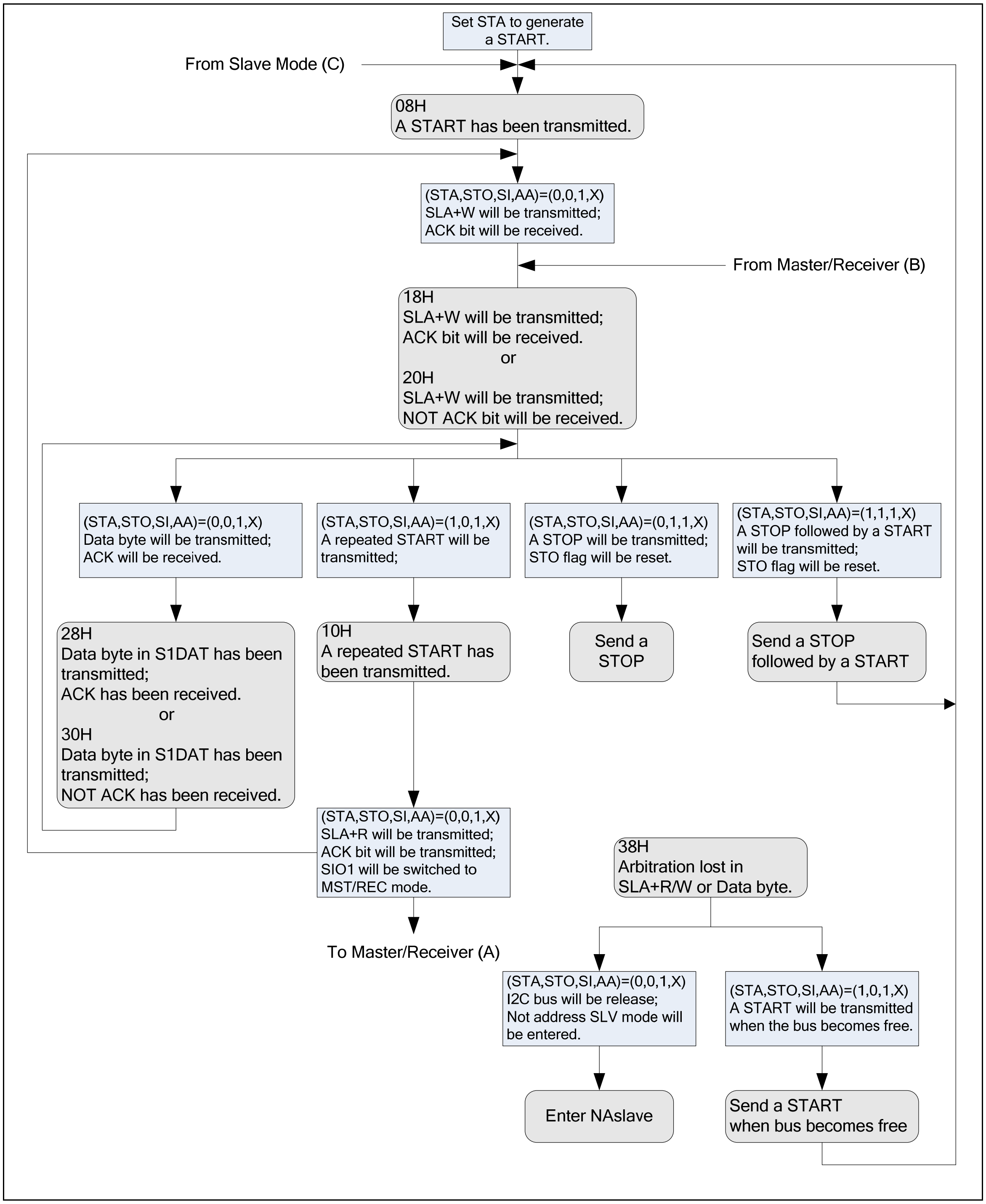


Figure 5-28 Master Transmitter Mode

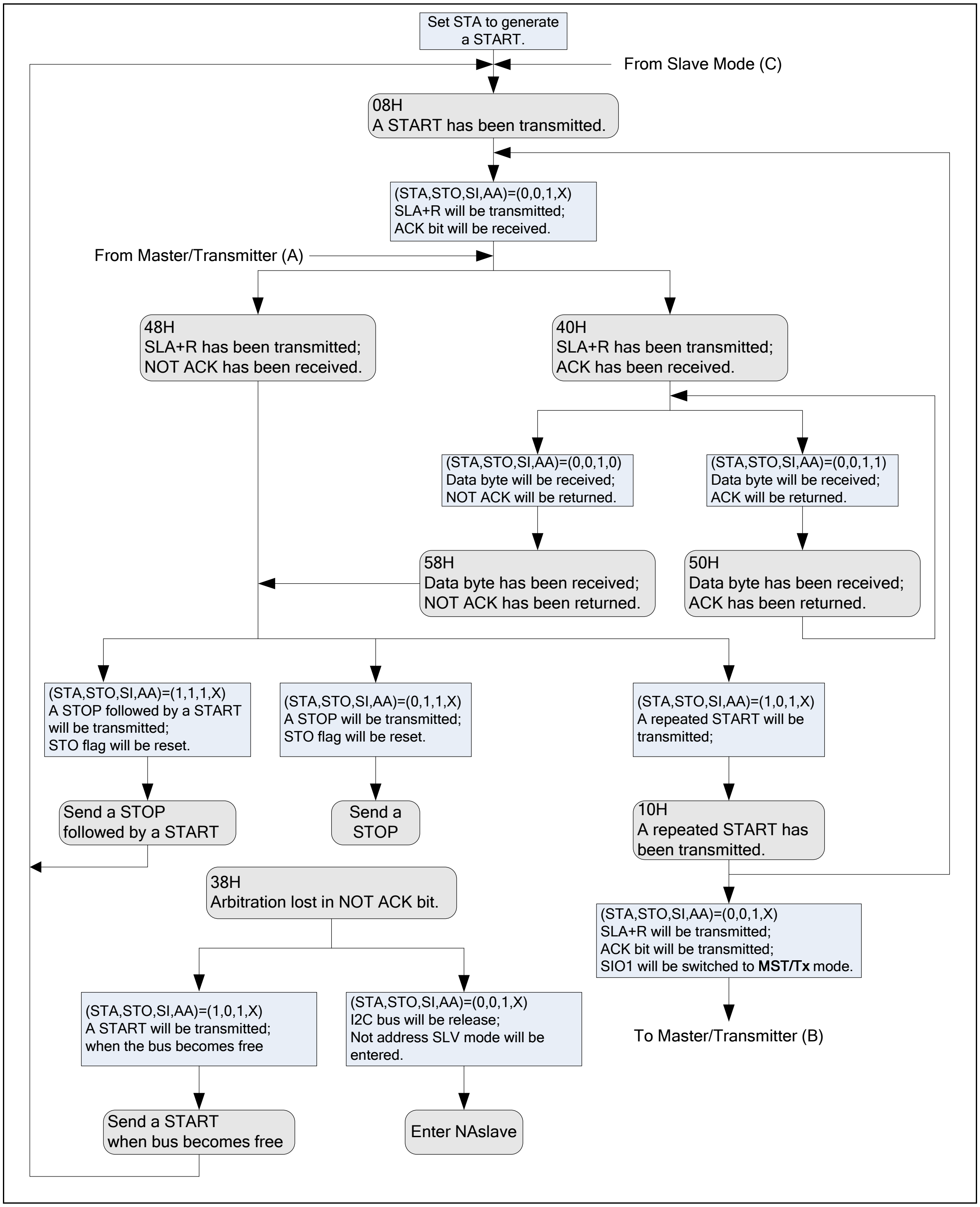


Figure 5-29 Master Receiver Mode

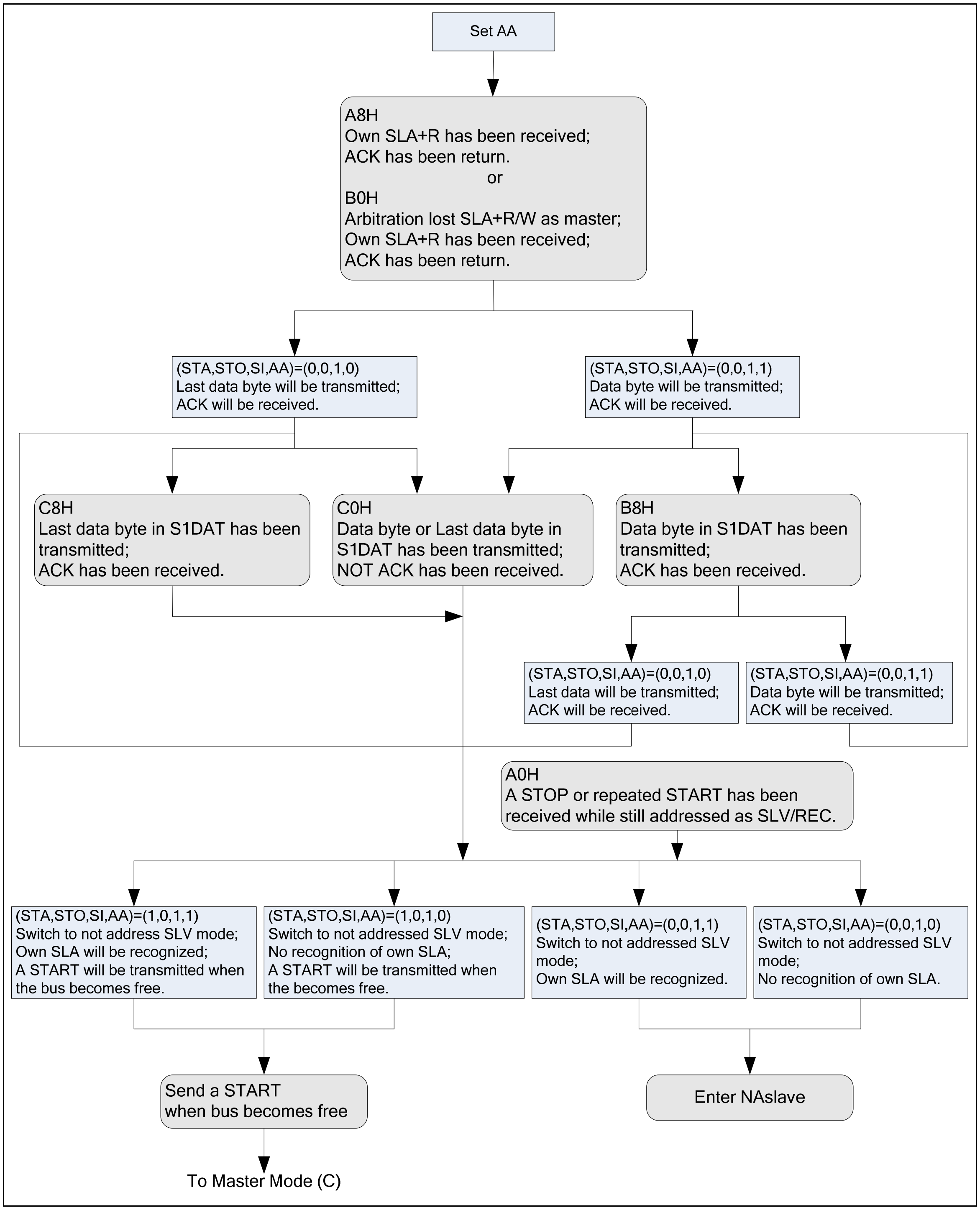


Figure 5-30 Slave Transmitter Mode

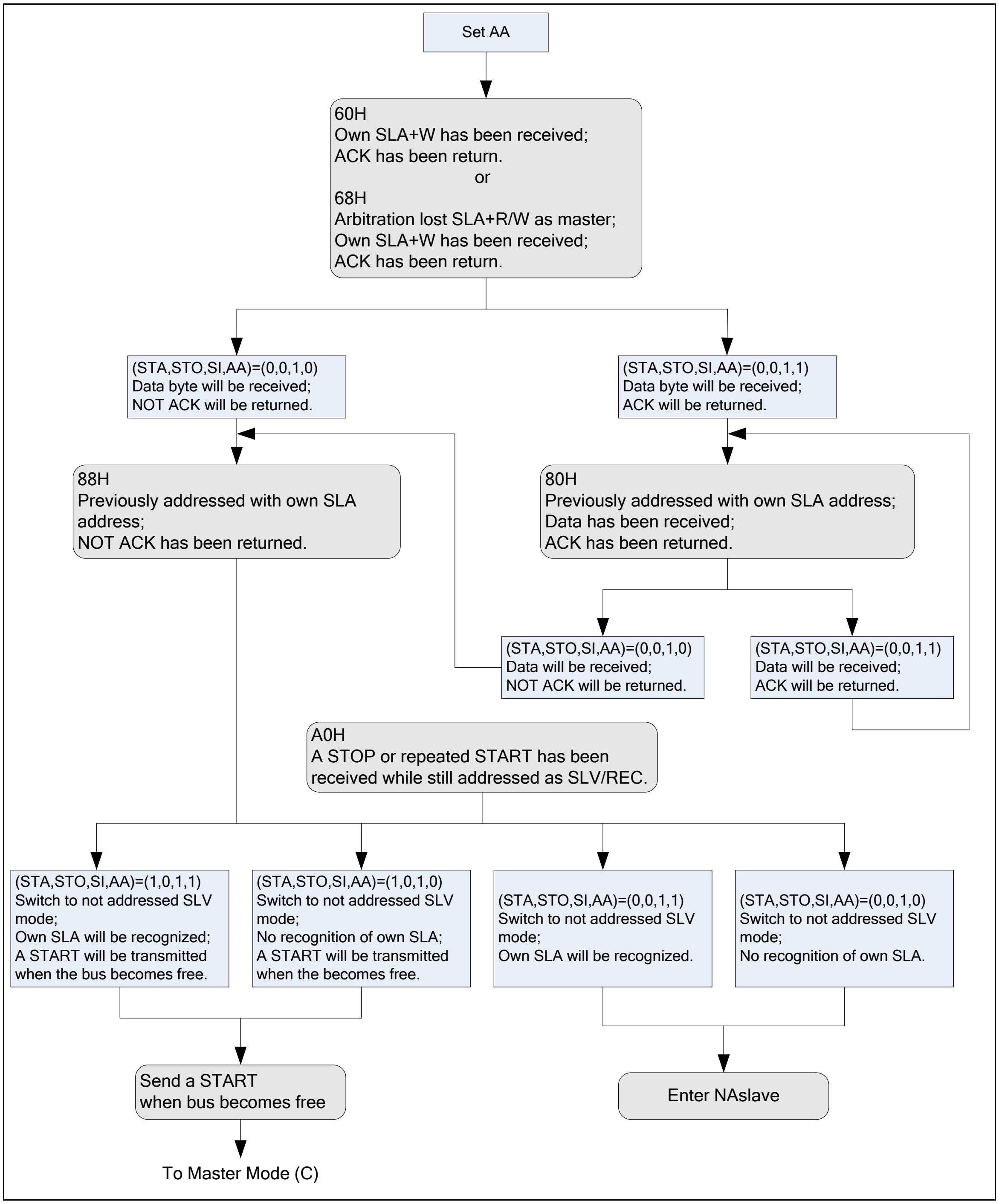


Figure 5-31 Slave Receiver Mode

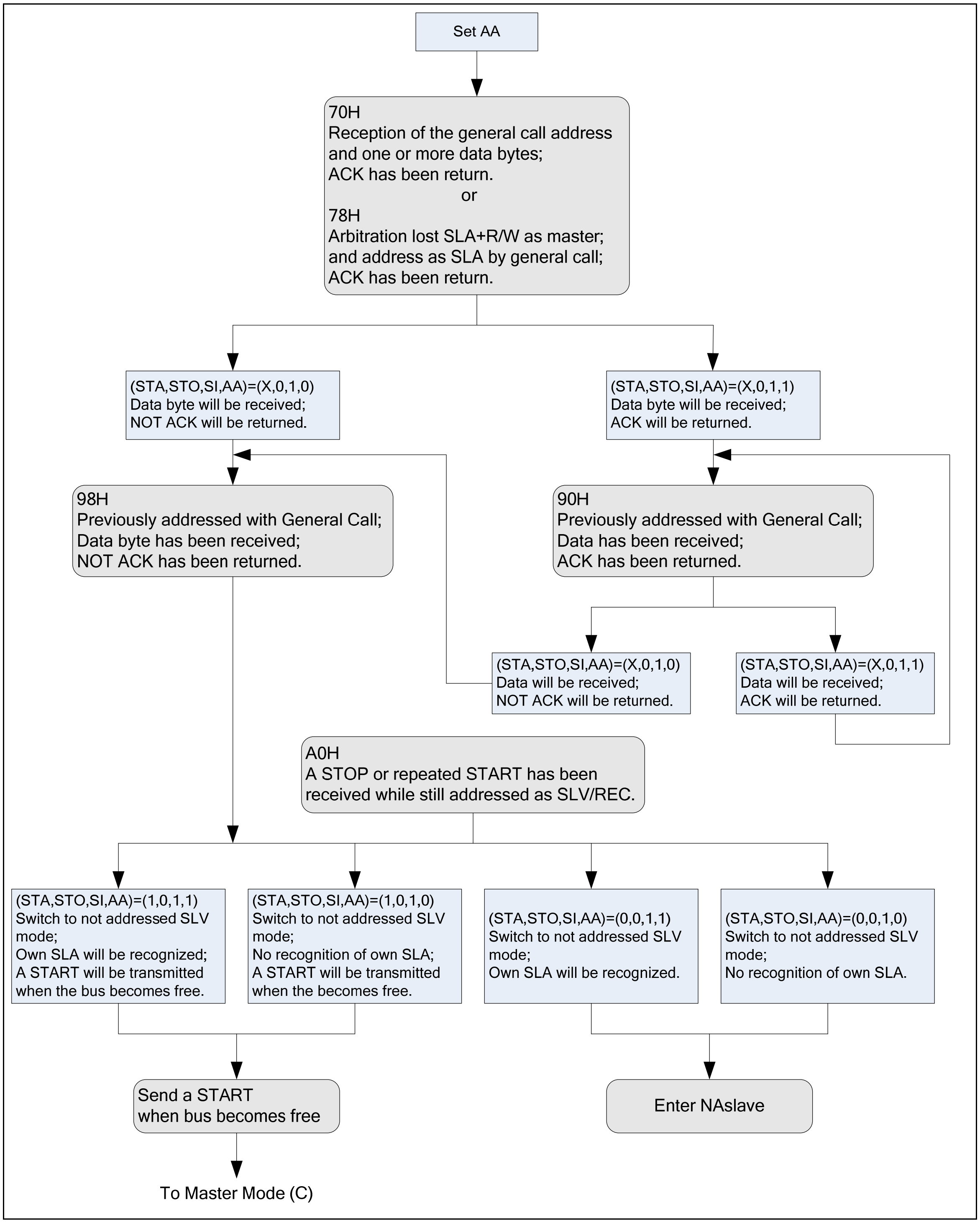


Figure 5-32 GC Mode

5.7 PWM Generator and Capture Timer (PWM)

5.7.1 Overview

NuMicro™ NUC130/NUC140 has 2 sets of PWM group supports total 4 sets of PWM Generators which can be configured as 8 independent PWM outputs, PWM0~PWM7, or as 4 complementary PWM pairs, (PWM0, PWM1), (PWM2, PWM3), (PWM4, PWM5) and (PWM6, PWM7) with 4 programmable dead-zone generators.

Each PWM Generator has one 8-bit prescaler, one clock divider with 5 divided frequencies (1, 1/2, 1/4, 1/8, 1/16), two PWM Timers including two clock selectors, two 16-bit PWM down-counters for PWM period control, two 16-bit comparators for PWM duty control and one dead-zone generator. The 4 sets of PWM Generators provide eight independent PWM interrupt flags which are set by hardware when the corresponding PWM period down counter reaches zero. Each PWM interrupt source with its corresponding enable bit can cause CPU to request PWM interrupt. The PWM generators can be configured as one-shot mode to produce only one PWM cycle signal or auto-reload mode to output PWM waveform continuously.

When PCR.DZEN01 is set, PWM0 and PWM1 perform complementary PWM paired function; the paired PWM period, duty and dead-time are determined by PWM0 timer and Dead-zone generator 0. Similarly, the complementary PWM pairs of (PWM2, PWM3), (PWM4, PWM5) and (PWM6, PWM7) are controlled by PWM2, PWM4 and PWM6 timers and Dead-zone generator 2, 4 and 6, respectively. Refer to Figure 5-33 to Figure 5-40 for the architecture of PWM Timers.

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

When the 16-bit period down counter reaches zero, the interrupt request is generated. If PWM-timer is set as auto-reload mode, when the down counter reaches zero, it is reloaded with PWM Counter Register (CNRx) automatically then start decreasing, repeatedly. If the PWM-timer is set as one-shot mode, the down counter will stop and generate one interrupt request when it reaches zero.

The value of PWM counter comparator is used for pulse high width modulation. The counter control logic changes the output to high level when down-counter value matches the value of compare register.

The alternate feature of the PWM-timer is digital input Capture function. If Capture function is enabled the PWM output pin is switched as capture input mode. The Capture0 and PWM0 share one timer which is included in PWM0 and the Capture1 and PWM1 share PWM1 timer, and etc. Therefore user must setup the PWM-timer before enable Capture feature. After capture feature is enabled, the capture always latched PWM-counter to Capture Rising Latch Register (CRLR) when input channel has a rising transition and latched PWM-counter to Capture Falling Latch Register (CFLR) when input channel has a falling transition. Capture channel 0 interrupt is programmable by setting CCR0.CRL_IE0[1] (Rising latch Interrupt enable) and CCR0.CFL_IE0[2]] (Falling latch Interrupt enable) to decide the condition of interrupt occur. Capture channel 1 has the same feature by setting CCR0.CRL_IE1[17] and CCR0.CFL_IE1[18]. And capture channel 2 to channel 3 on each group have the same feature by setting the corresponding control bits in CCR2. For each group, whenever Capture issues Interrupt 0/1/2/3, the PWM counter 0/1/2/3 will be reload at this moment.

The maximum captured frequency that PWM can capture is confined by the capture interrupt latency. When capture interrupt occurred, software will do at least three steps, they are: Read

PIIR to get interrupt source and Read CRLRx/CFLRx(x=0~3) to get capture value and finally write 1 to clear PIIR to zero. If interrupt latency will take time T0 to finish, the capture signal mustn't transition during this interval (T0). In this case, the maximum capture frequency will be 1/T0. For example:

HCLK = 50 MHz, PWM_CLK = 25 MHz, Interrupt latency is 900 ns

So the maximum capture frequency will is 1/900ns ≈ 1000 kHz

5.7.2 Features

5.7.2.1 PWM function features:

- PWM group has two PWM generators. Each PWM generator supports one 8-bit prescaler, one clock divider, two PWM-timers (down counter), one dead-zone generator and two PWM outputs.
- Up to 16-bit resolution
- PWM Interrupt request synchronized with PWM period
- One-shot or Auto-reload mode PWM
- Up to 2 PWM group (PWMA/PWMB) to support 8 PWM channels or 4 PWM paired channels

5.7.2.2 Capture Function Features:

- Timing control logic shared with PWM Generators
- Support 8 Capture input channels shared with 8 PWM output channels
- Each channel supports one rising latch register (CRLR), one falling latch register (CFLR) and Capture interrupt flag (CAPIFx)

5.7.3 Block Diagram

The Figure 5-33 to Figure 5-40 illustrate the architecture of PWM in pair (PWM-Timer 0/1 are in one pair and PWM-Timer 2/3 are in another one, and so on.).

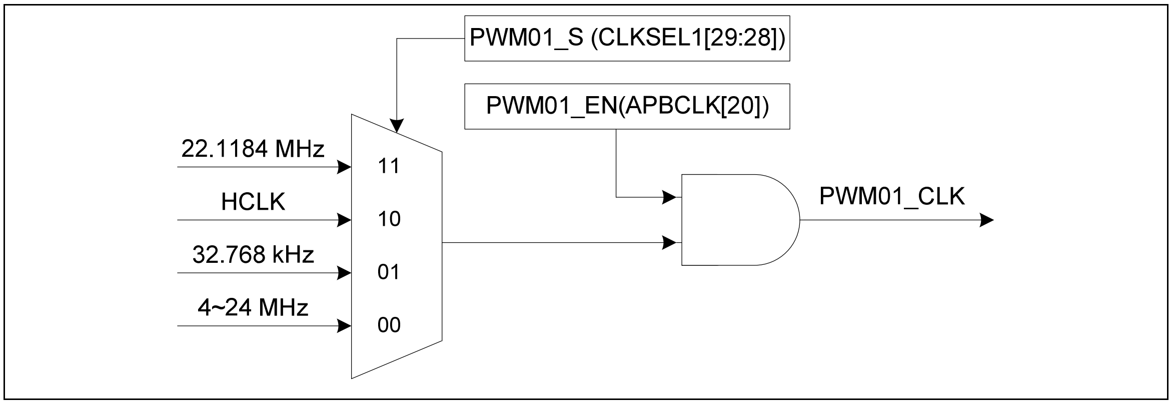


Figure 5-33 PWM Generator 0 Clock Source Control

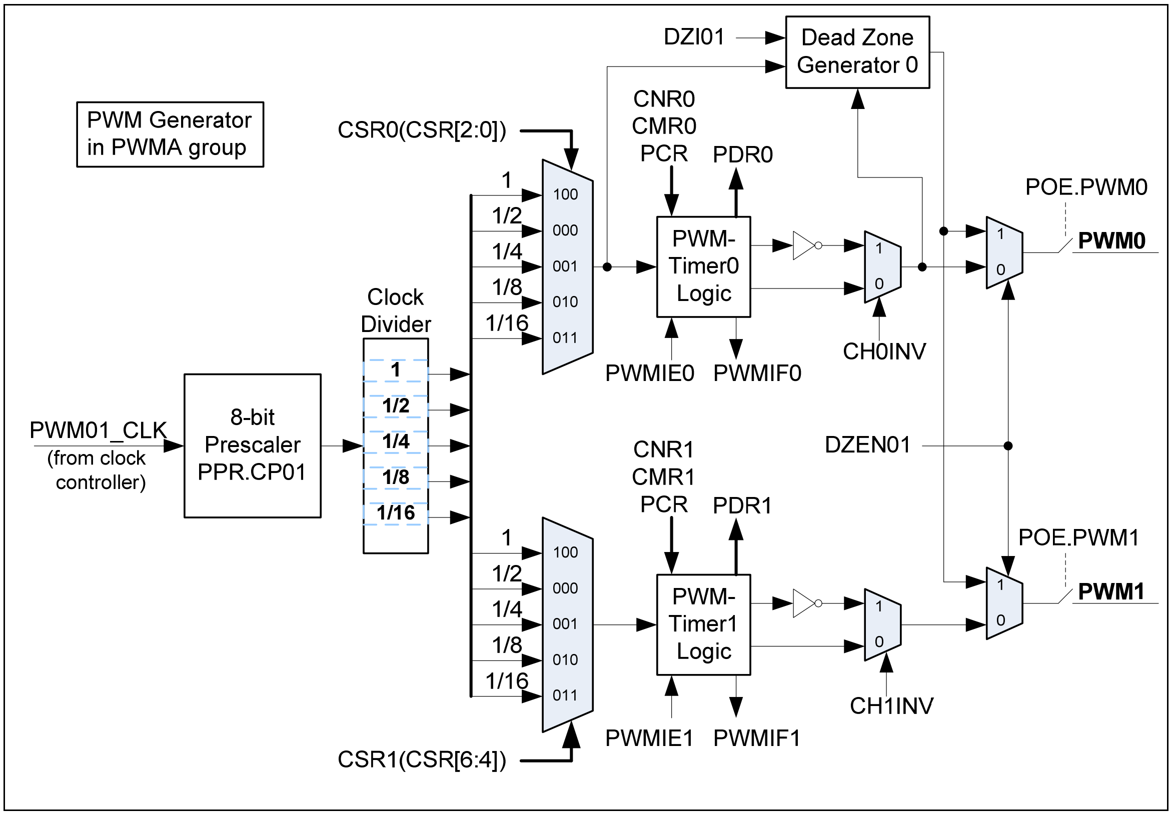


Figure 5-34 PWM Generator 0 Architecture Diagram

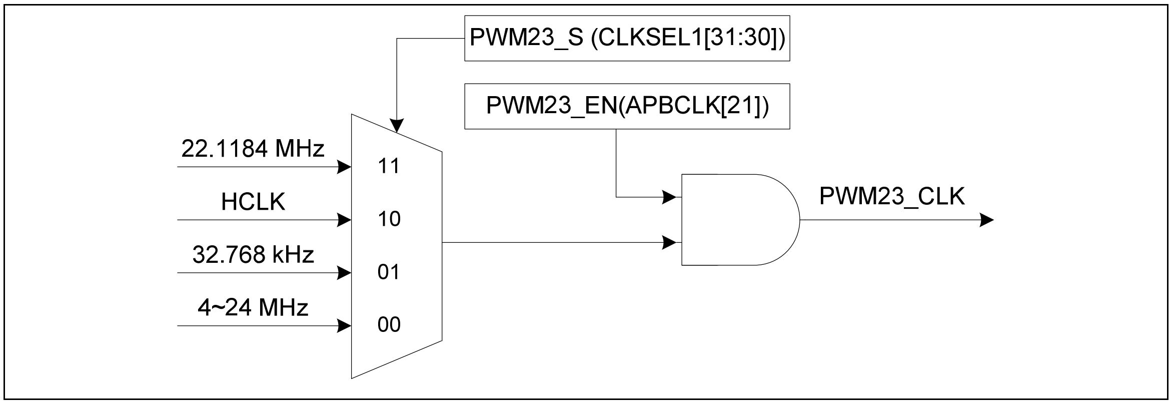


Figure 5-35 PWM Generator 2 Clock Source Control

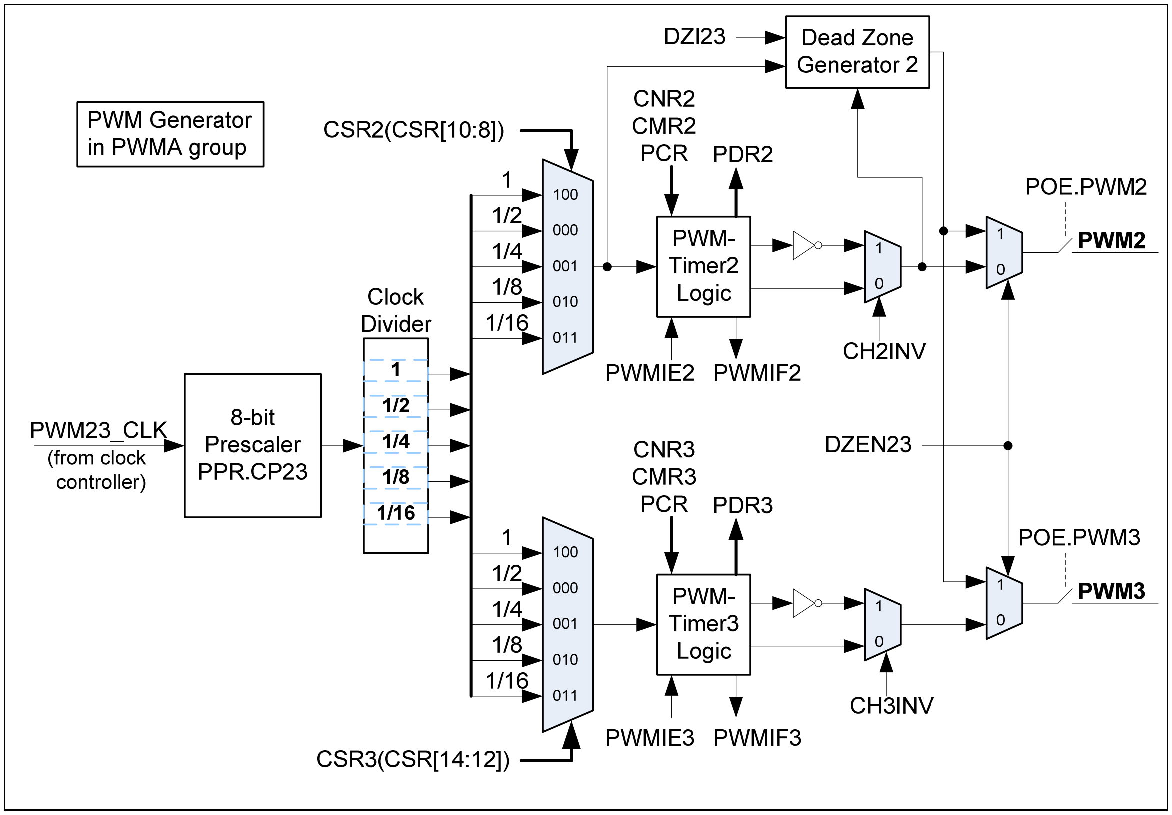


Figure 5-36 PWM Generator 2 Architecture Diagram

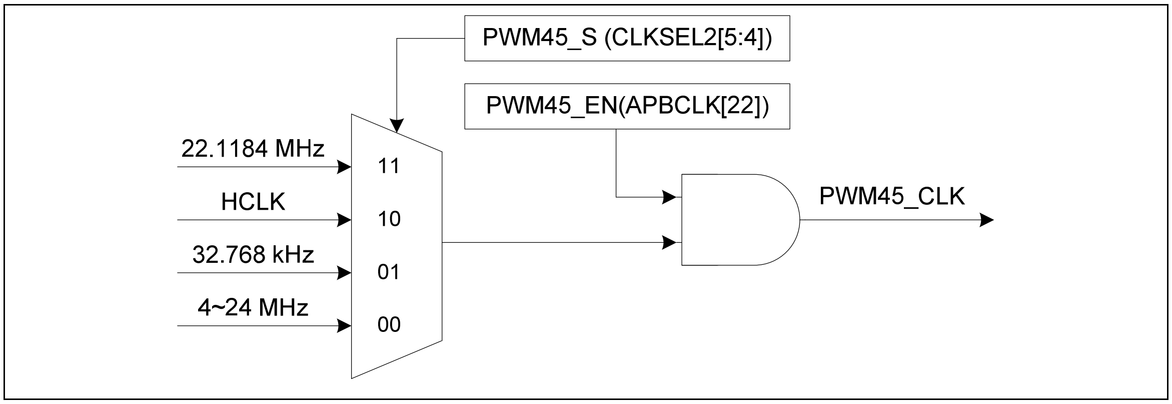


Figure 5-37 PWM Generator 4 Clock Source Control

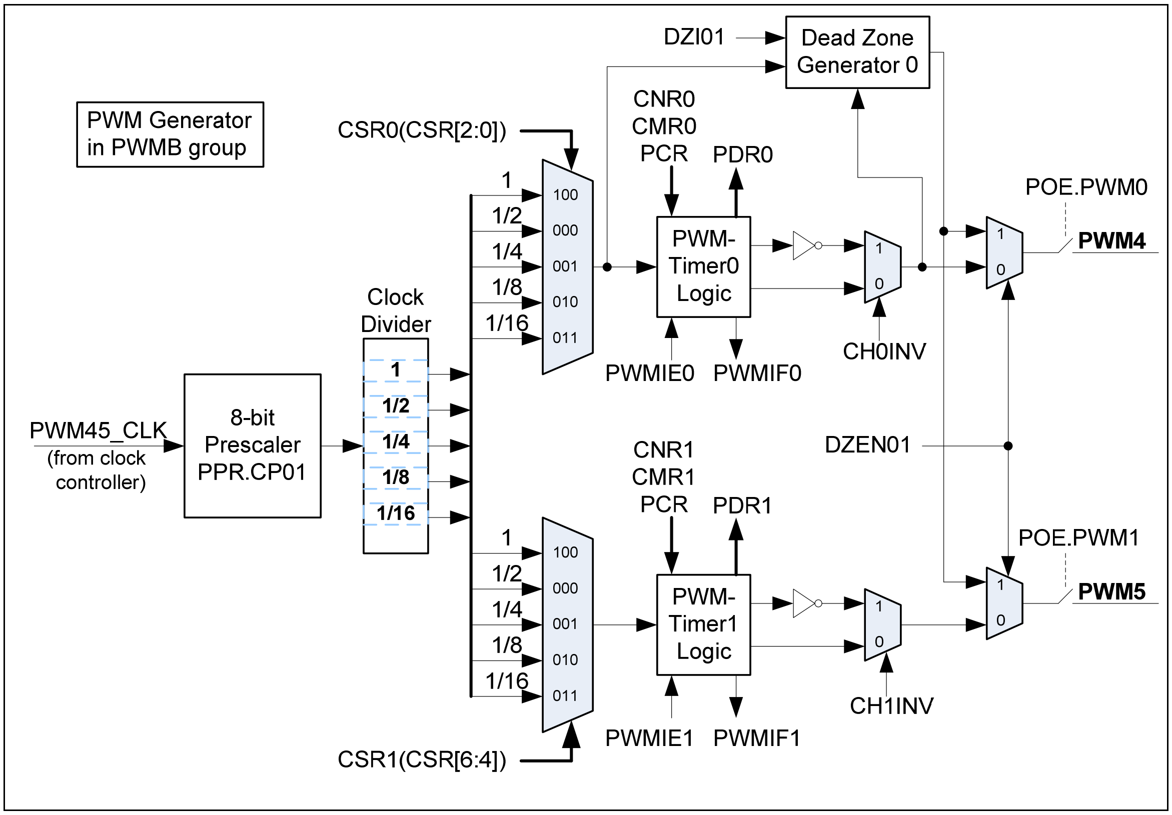


Figure 5-38 PWM Generator 4 Architecture Diagram

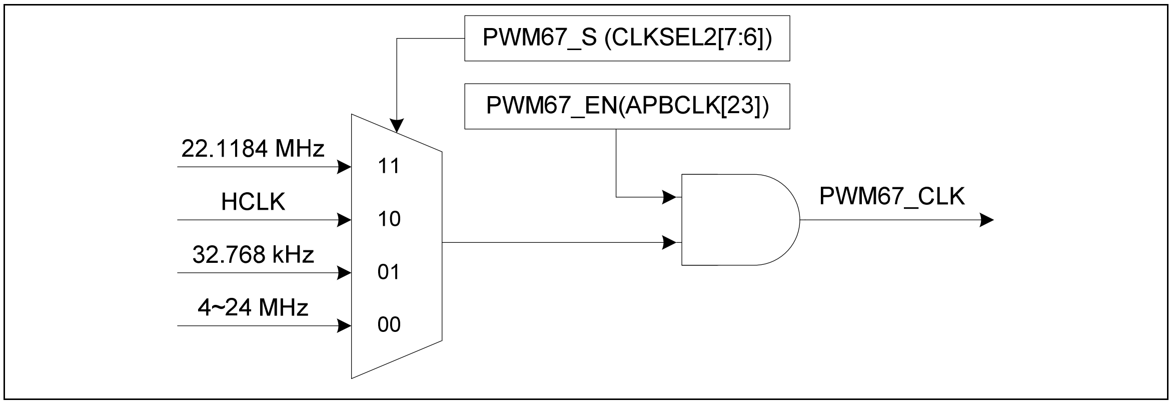


Figure 5-39 PWM Generator 6 Clock Source Control

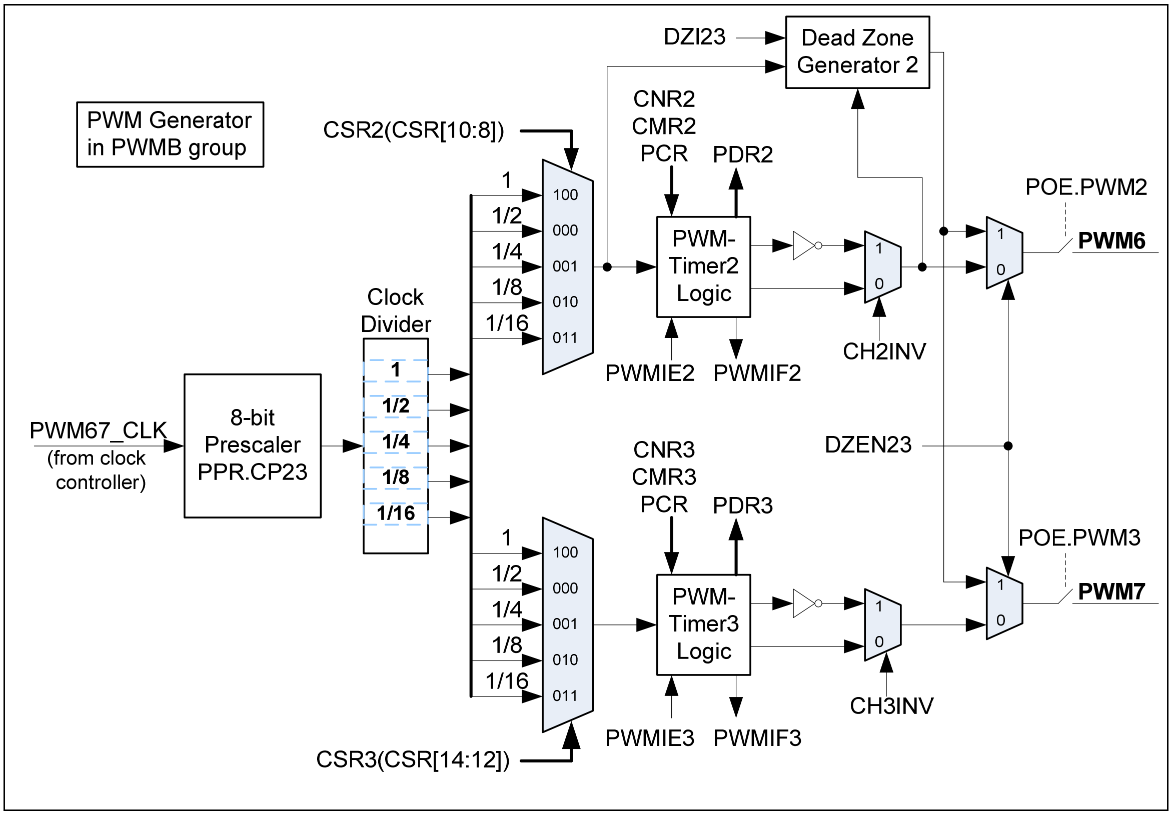


Figure 5-40 PWM Generator 6 Architecture Diagram

5.7.4 Function Description

5.7.4.1 PWM-Timer Operation

The PWM period and duty control are configured by PWM down-counter register (CNR) and PWM comparator register (CMR). The PWM-timer timing operation is shown in Figure 5-42. The pulse width modulation follows the formula as below and the legend of PWM-Timer Comparator is shown as Figure 5-41. Note that the corresponding GPIO pins must be configured as PWM function (enable POE and disable CAPENR) for the corresponding PWM channel.

- PWM frequency = PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)]; where xy, could be 01, 23, 45 or 67, depends on selected PWM channel.
- Duty ratio = (CMR+1)/(CNR+1)
- CMR >= CNR: PWM output is always high
- CMR < CNR: PWM low width= (CNR-CMR) unit[1]; PWM high width = (CMR+1) unit
- CMR = 0: PWM low width = (CNR) unit; PWM high width = 1 unit

Note: [1] Unit = one PWM clock cycle.

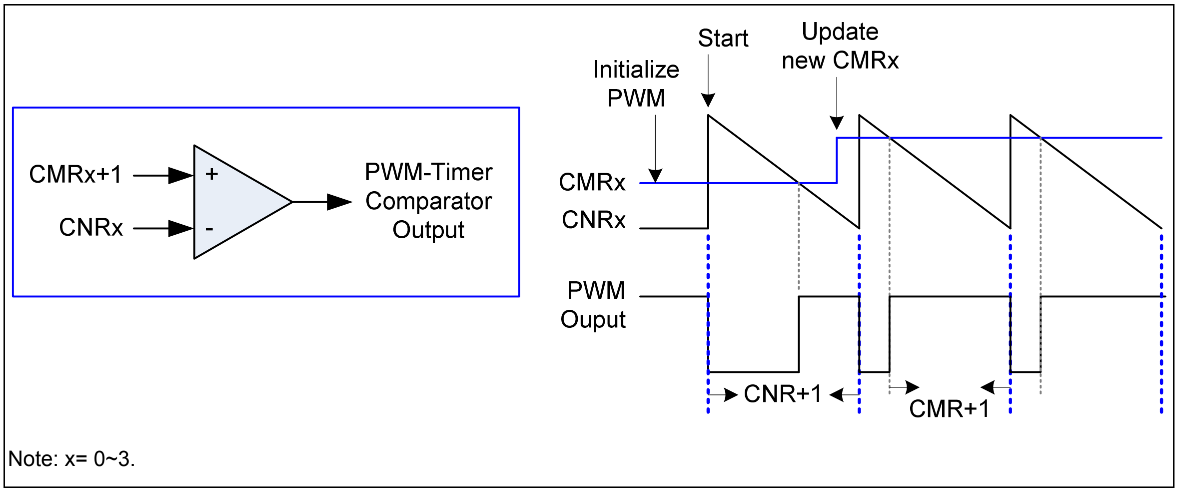


Figure 5-41 Legend of Internal Comparator Output of PWM-Timer

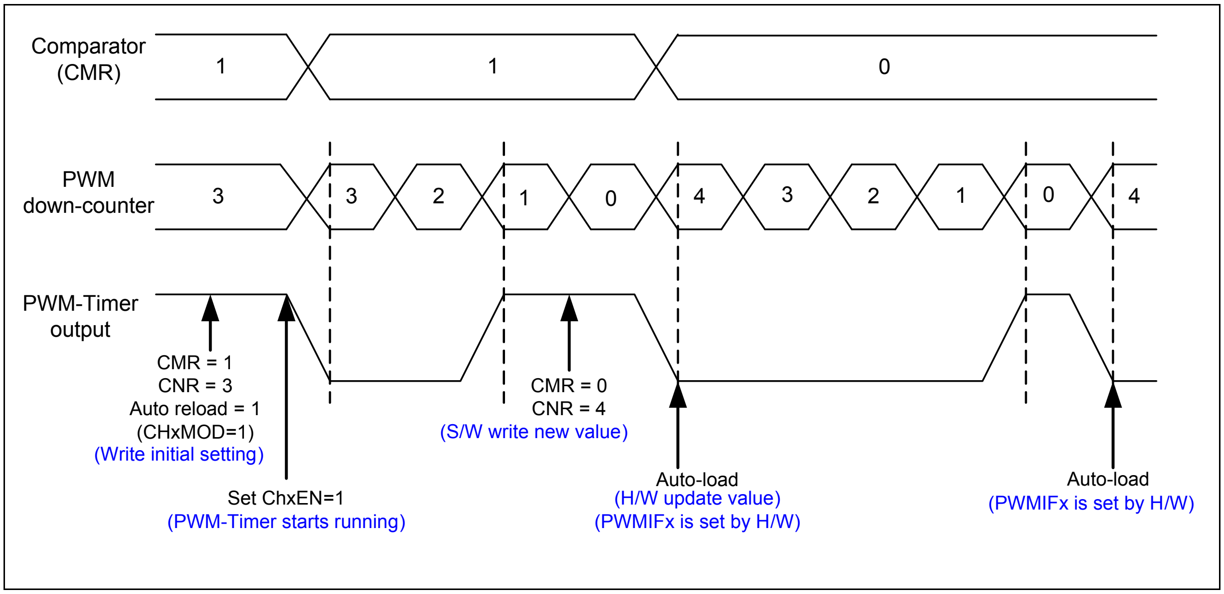


Figure 5-42 PWM-Timer Operation Timing

5.7.4.2 *PWM Double Buffering, Auto-reload and One-shot Operation*

PWM Timers have double buffering function the reload value is updated at the start of next period without affecting current timer operation. The PWM counter value can be written into CNRx and current PWM counter value can be read from PDRx.

The bit CH0MOD in PWM Control Register (PCR) defines PWM0 operates in auto-reload or one-shot mode If CH0MOD is set to one, the auto-reload operation loads CNR0 to PWM counter when PWM counter reaches zero. If CNR0 are set to zero, PWM counter will be halt when PWM counter counts to zero. If CH0MOD is set as zero, counter will be stopped immediately. PWM1~PWM7 performs the same function as PWM0.

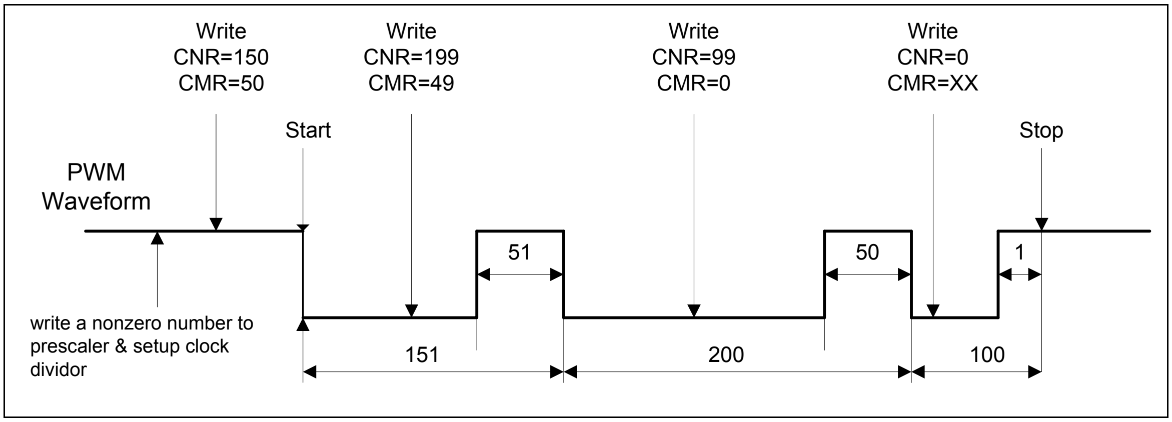


Figure 5-43 PWM Double Buffering Illustration

5.7.4.3 *Modulate Duty Ratio*

The double buffering function allows CMRx written at any point in current cycle. The loaded value will take effect from next cycle.

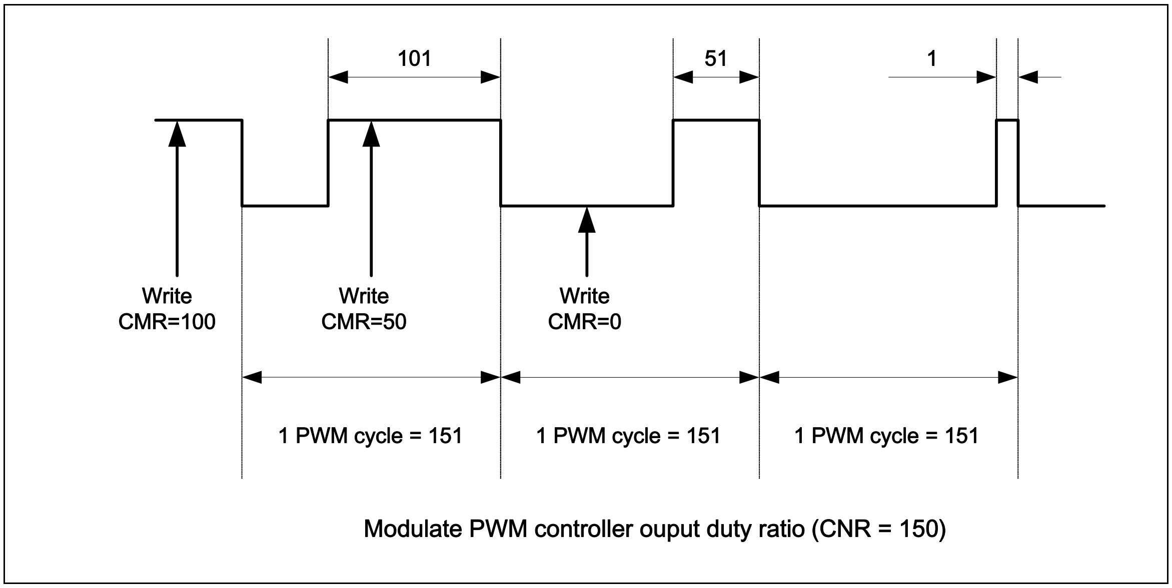


Figure 5-44 PWM Controller Output Duty Ratio

5.7.4.4 *Dead-Zone Generator*

PWM controller is implemented with Dead Zone generator. They are built for power device protection. This function generates a programmable time gap to delay PWM rising output. User can program PPRx.DZI to determine the Dead Zone interval.

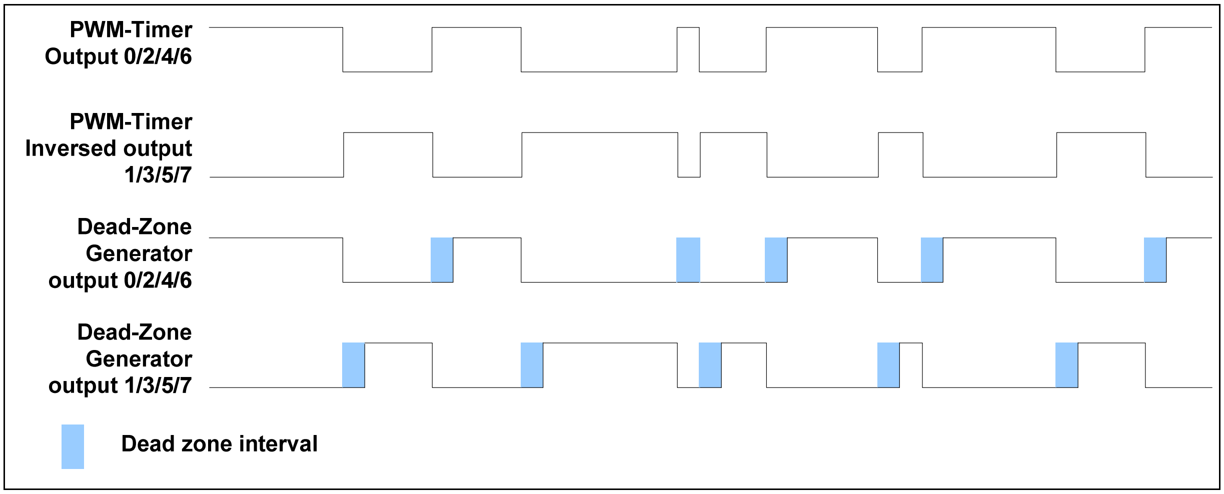


Figure 5-45 Paired-PWM Output with Dead Zone Generation Operation

5.7.4.5 Capture Operation

The Capture 0 and PWM 0 share one timer that included in PWM 0; and the Capture 1 and PWM 1 share another timer, and etc. The capture always latches PWM-counter to CRLRx when input channel has a rising transition and latches PWM-counter to CFLRx when input channel has a falling transition. Capture channel 0 interrupt is programmable by setting CCR0[1] (Rising latch Interrupt enable) and CCR0[2] (Falling latch Interrupt enable) to decide the condition of interrupt occur. Capture channel 1 has the same feature by setting CCR0[17] and CCR0[18], and etc. Whenever the Capture controller issues a capture interrupt, the corresponding PWM counter will be reloaded with CNRx at this moment. Note that the corresponding GPIO pins must be configured as capture function (disable POE and enable CAPENR) for the corresponding capture channel.

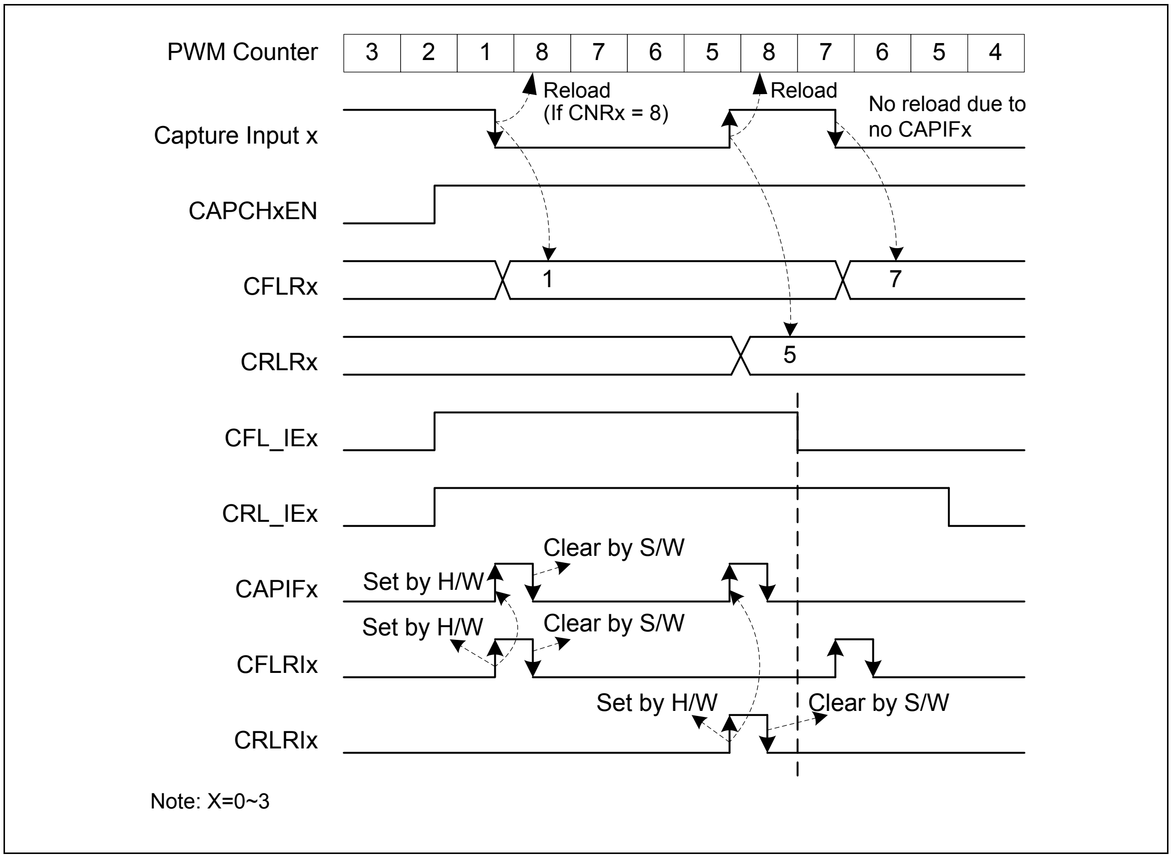


Figure 5-46 Capture Operation Timing

At this case, the CNR is 8:

1. The PWM counter will be reloaded with CNRx when a capture interrupt flag (CAPIFx) is set.
2. The channel low pulse width is (CNR + 1 - CRLR).
3. The channel high pulse width is (CNR + 1 - CFLR).

5.7.4.6 PWM-Timer Interrupt Architecture

There are eight PWM interrupts, PWM0_INT~PWM7_INT, which are divided into PWMA_INT and PWMB_INT for Advanced Interrupt Controller (AIC). PWM 0 and Capture 0 share one interrupt, PWM1 and Capture 1 share the same interrupt and so on. Therefore, PWM function and Capture function in the same channel cannot be used at the same time. Figure 5-47 and Figure 5-48 demonstrates the architecture of PWM-Timer interrupts.

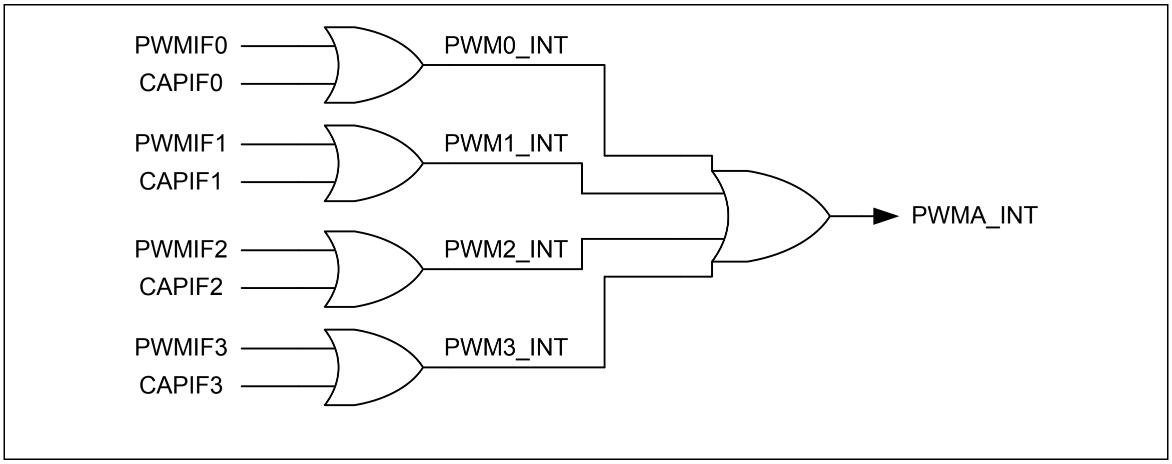


Figure 5-47 PWM Group A PWM-Timer Interrupt Architecture Diagram

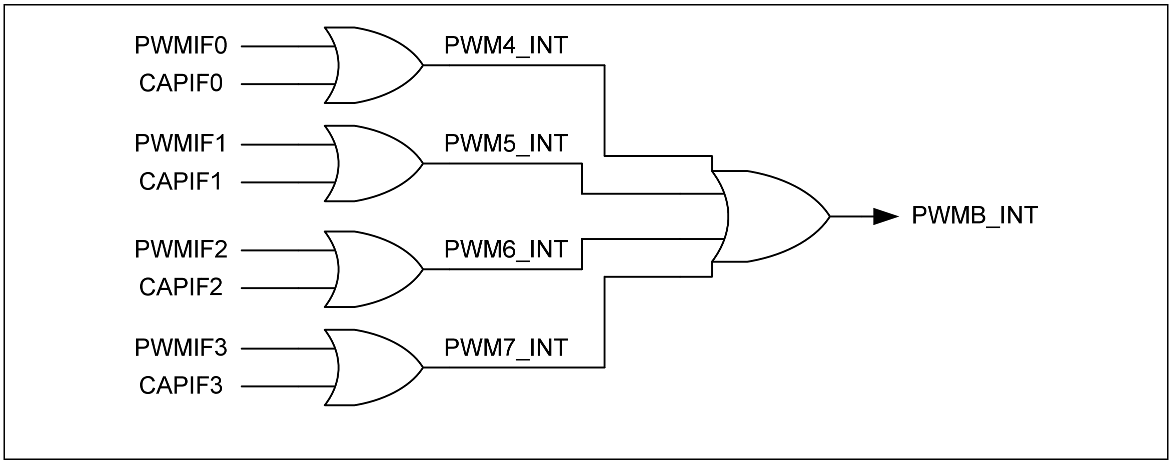


Figure 5-48 PWM Group B PWM-Timer Interrupt Architecture Diagram

5.7.4.7 PWM-Timer Start Procedure

The following procedure is recommended for starting a PWM drive.

1. Setup clock selector (CSR)
2. Setup prescaler (PPR)
3. Setup inverter on/off, dead zone generator on/off, auto-reload/one-shot mode and Stop PWM-timer (PCR)
4. Setup comparator register (CMR) for setting PWM duty.
5. Setup PWM down-counter register (CNR) for setting PWM period.
6. Setup interrupt enable register (PIER)
7. Setup corresponding GPIO pins as PWM function (enable POE and disable CAPENR) for the corresponding PWM channel.
8. Enable PWM timer start running (Set CHxEN = 1 in PCR)

5.7.4.8 PWM-Timer Stop Procedure

Method 1:

Set 16-bit down counter (CNR) as 0, and monitor PDR (current value of 16-bit down-counter). When PDR reaches to 0, disable PWM-Timer (CHxEN in PCR). ***(Recommended)***

Method 2:

Set 16-bit down counter (CNR) as 0. When interrupt request happened, disable PWM-Timer (CHxEN in PCR). ***(Recommended)***

Method 3:

Disable PWM-Timer directly ((CHxEN in PCR). ***(Not recommended)***

The reason why method 3 is not recommended is that disable CHxEN will immediately stop PWM output signal and lead to change the duty of the PWM output, this may cause damage to the control circuit of motor

5.7.4.9 Capture Start Procedure

1. Setup clock selector (CSR)
2. Setup prescaler (PPR)
3. Setup channel enabled, rising/falling interrupt enable and input signal inverter on/off (CCR0, CCR2)
4. Setup PWM down-counter (CNR)
5. Setup corresponding GPIO pins as capture function (disable POE and enable CAPENR) for the corresponding PWM channel.
6. Enable PWM timer start running (Set CHxEN = 1 in PCR)

5.7.5 Register Map

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

Register	**Offset**	**R/W**	**Description**	**Reset Value**
PWMA_BA = 0x4004_0000 (PWM group A) **PWMB_BA = 0x4014_0000 (PWM group B)**				
PPR	PWMA_BA+0x00	R/W	PWM Group A Prescaler Register	0x0000_0000
	PWMB_BA+0x00	R/W	PWM Group B Prescaler Register	0x0000_0000
CSR	PWMA_BA+0x04	R/W	PWM Group A Clock Select Register	0x0000_0000
	PWMB_BA+0x04	R/W	PWM Group B Clock Select Register	0x0000_0000
PCR	PWMA_BA+0x08	R/W	PWM Group A Control Register	0x0000_0000
	PWMB_BA+0x08	R/W	PWM Group B Control Register	0x0000_0000
CNR0	PWMA_BA+0x0C	R/W	PWM Group A Counter Register 0	0x0000_0000
	PWMB_BA+0x0C	R/W	PWM Group B Counter Register 0	0x0000_0000
CMR0	PWMA_BA+0x10	R/W	PWM Group A Comparator Register 0	0x0000_0000
	PWMB_BA+0x10	R/W	PWM Group B Comparator Register 0	0x0000_0000
PDR0	PWMA_BA+0x14	R	PWM Group A Data Register 0	0x0000_0000
	PWMB_BA+0x14	R	PWM Group B Data Register 0	0x0000_0000
CNR1	PWMA_BA+0x18	R/W	PWM Group A Counter Register 1	0x0000_0000
	PWMB_BA+0x18	R/W	PWM Group B Counter Register 1	0x0000_0000
CMR1	PWMA_BA+0x1C	R/W	PWM Group A Comparator Register 1	0x0000_0000
	PWMB_BA+0x1C	R/W	PWM Group B Comparator Register 1	0x0000_0000
PDR1	PWMA_BA+0x20	R	PWM Group A Data Register 1	0x0000_0000
	PWMB_BA+0x20	R	PWM Group B Data Register 1	0x0000_0000
CNR2	PWMA_BA+0x24	R/W	PWM Group A Counter Register 2	0x0000_0000
	PWMB_BA+0x24	R/W	PWM Group B Counter Register 2	0x0000_0000
CMR2	PWMA_BA+0x28	R/W	PWM Group A Comparator Register 2	0x0000_0000
	PWMB_BA+0x28	R/W	PWM Group B Comparator Register 2	0x0000_0000
PDR2	PWMA_BA+0x2C	R	PWM Group A Data Register 2	0x0000_0000
	PWMB_BA+0x2C	R	PWM Group B Data Register 2	0x0000_0000
CNR3	PWMA_BA+0x30	R/W	PWM Group A Counter Register 3	0x0000_0000

	PWMB_BA+0x30	R/W	PWM Group B Counter Register 3	0x0000_0000
CMR3	PWMA_BA+0x34	R/W	PWM Group A Comparator Register 3	0x0000_0000
	PWMB_BA+0x34	R/W	PWM Group B Comparator Register 3	0x0000_0000
PDR3	PWMA_BA+0x38	R	PWM Group A Data Register 3	0x0000_0000
	PWMB_BA+0x38	R	PWM Group B Data Register 3	0x0000_0000
PBCR	PWMA_BA+0x3C	R/W	PWM Group A backward compatible Register	0x0000_0000
	PWMB_BA+0x3C	R/W	PWM Group B backward compatible Register	0x0000_0000
PIER	PWMA_BA+0x40	R/W	PWM Group A Interrupt Enable Register	0x0000_0000
	PWMB_BA+0x40	R/W	PWM Group B Interrupt Enable Register	0x0000_0000
PIIR	PWMA_BA+0x44	R/W	PWM Group A Interrupt Indication Register	0x0000_0000
	PWMB_BA+0x44	R/W	PWM Group B Interrupt Indication Register	0x0000_0000
CCR0	PWMA_BA+0x50	R/W	PWM Group A Capture Control Register 0	0x0000_0000
	PWMB_BA+0x50	R/W	PWM Group B Capture Control Register 0	0x0000_0000
CCR2	PWMA_BA+0x54	R/W	PWM Group A Capture Control Register 2	0x0000_0000
	PWMB_BA+0x54	R/W	PWM Group B Capture Control Register 2	0x0000_0000
CRLR0	PWMA_BA+0x58	R	PWM Group A Capture Rising Latch Register (Channel 0)	0x0000_0000
	PWMB_BA+0x58	R	PWM Group B Capture Rising Latch Register (Channel 0)	0x0000_0000
CFLR0	PWMA_BA+0x5C	R	PWM Group A Capture Falling Latch Register (Channel 0)	0x0000_0000
	PWMB_BA+0x5C	R	PWM Group B Capture Falling Latch Register (Channel 0)	0x0000_0000
CRLR1	PWMA_BA+0x60	R	PWM Group A Capture Rising Latch Register (Channel 1)	0x0000_0000
	PWMB_BA+0x60	R	PWM Group B Capture Rising Latch Register (Channel 1)	0x0000_0000
CFLR1	PWMA_BA+0x64	R	PWM Group A Capture Falling Latch Register (Channel 1)	0x0000_0000
	PWMB_BA+0x64	R	PWM Group B Capture Falling Latch Register (Channel 1)	0x0000_0000
CRLR2	PWMA_BA+0x68	R	PWM Group A Capture Rising Latch Register (Channel 2)	0x0000_0000
	PWMB_BA+0x68	R	PWM Group B Capture Rising Latch Register (Channel 2)	0x0000_0000
CFLR2	PWMA_BA+0x6C	R	PWM Group A Capture Falling Latch Register (Channel 2)	0x0000_0000
	PWMB_BA+0x6C	R	PWM Group B Capture Falling Latch Register (Channel 2)	0x0000_0000
CRLR3	PWMA_BA+0x70	R	PWM Group A Capture Rising Latch Register (Channel 3)	0x0000_0000
	PWMB_BA+0x70	R	PWM Group B Capture Rising Latch Register (Channel 3)	0x0000_0000
CFLR3	PWMA_BA+0x74	R	PWM Group A Capture Falling Latch Register (Channel 3)	0x0000_0000

	PWMB_BA+0x74	R	PWM Group B Capture Falling Latch Register (Channel 3)	0x0000_0000
CAPENR	PWMA_BA+0x78	R/W	PWM Group A Capture Input 0~3 Enable Register	0x0000_0000
	PWMB_BA+0x78	R/W	PWM Group B Capture Input 0~3 Enable Register	0x0000_0000
POE	PWMA_BA+0x7C	R/W	PWM Group A Output Enable for channel 0~3	0x0000_0000
	PWMB_BA+0x7C	R/W	PWM Group B Output Enable for channel 0~3	0x0000_0000

5.7.6 Register Description

PWM Pre-Scale Register (PPR)

Register	Offset	R/W	Description	Reset Value
PPR	PWMA_BA+0x00	R/W	PWM Group A Pre-scale Register	0x0000_0000
	PWMB_BA+0x00	R/W	PWM Group B Pre-scale Register	0x0000_0000

31	30	29	28	27	26	25	24
DZI23							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DZI01							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CP23							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CP01							

Bits	Descriptions	
[31:24]	**DZI23**	**Dead Zone Interval for Pair of Channel2 and Channel3** (PWM2 and PWM3 pair for PWM group A, PWM6 and PWM7 pair for PWM group B) These 8-bit determine dead zone length. The unit time of dead zone length is received from corresponding CSR bits.
[23:16]	**DZI01**	**Dead Zone Interval for Pair of Channel 0 and Channel 1** (PWM0 and PWM1 pair for PWM group A, PWM4 and PWM5 pair for PWM group B) These 8-bit determine dead zone length. The unit time of dead zone length is received from corresponding CSR bits.
[15:8]	**CP23**	**Clock Prescaler 2** (PWM-timer2 / 3 for group A and PWM-timer 6 / 7 for group B) Clock input is divided by (CP23 + 1) before it is fed to the corresponding PWM-timer If CP23=0, then the clock prescaler 2 output clock will be stopped. So corresponding PWM-timer will be stopped also.
[7:0]	**CP01**	**Clock Prescaler 0** (PWM-timer 0 / 1 for group A and PWM-timer 4 5 for group B) Clock input is divided by (CP01 + 1) before it is fed to the corresponding PWM-timerr If CP01=0, then the clock prescaler 0 output clock will be stopped. So corresponding PWM-timer will be stopped also.

PWM Clock Selector Register (CSR)

Register	Offset	R/W	Description	Reset Value
CSR	PWMA_BA+0x04	R/W	PWM Group A Clock Selector Register	0x0000_0000
	PWMB_BA+0x04	R/W	PWM Group B Clock Selector 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	**CSR3**			**Reserved**	**CSR2**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**CSR1**			**Reserved**	**CSR0**		

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[31:15]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[14:12]</td><td>CSR3</td><td>PWM Timer 3 Clock Source Selection (PWM timer 3 for group A and PWM timer 7 for group B)
Select clock input for PWM timer.
<table><tr><th>CSR3 [14:12]</th><th>Input clock divided by</th></tr><tr><td>100</td><td>1</td></tr><tr><td>011</td><td>16</td></tr><tr><td>010</td><td>8</td></tr><tr><td>001</td><td>4</td></tr><tr><td>000</td><td>2</td></tr></table></td></tr>
<tr><td>[11]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[10:8]</td><td>CSR2</td><td>PWM Timer 2 Clock Source Selection (PWM timer 2 for group A and PWM timer 6 for group B)
Select clock input for PWM timer.
(Table is the same as CSR3)</td></tr>
<tr><td>[7]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[6:4]</td><td>CSR1</td><td>PWM Timer 1 Clock Source Selection (PWM timer 1 for group A and PWM timer 5 for group B)
Select clock input for PWM timer.
(Table is the same as CSR3)</td></tr>
</table>

[3]	**Reserved**	Reserved
[2:0]	**CSR0**	**PWM Timer 0 Clock Source Selection** (PWM timer 0 for group A and PWM timer 4 for group B) Select clock input for PWM timer. (Table is the same as CSR3)

PWM Control Register (PCR)

Register	Offset	R/W	Description	Reset Value
PCR	PWMA_BA+0x08	R/W	PWM Group A Control Register (PCR)	0x0000_0000
	PWMB_BA+0x08	R/W	PWM Group B Control Register (PCR)	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				**CH3MOD**	**CH3INV**	**Reserved**	**CH3EN**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved				**CH2MOD**	**CH2INV**	**Reserved**	**CH2EN**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**CH1MOD**	**CH1INV**	**Reserved**	**CH1EN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**DZEN23**	**DZEN01**	**CH0MOD**	**CH0INV**	**Reserved**	**CH0EN**

Bits	Descriptions	
[31:28]	**Reserved**	Reserved
[27]	**CH3MOD**	**PWM-Timer 3 Auto-reload/One-Shot Mode** (PWM timer 3 for group A and PWM timer 7 for group B) 1 = Auto-reload Mode 0 = One-Shot Mode Note: If there is a transition at this bit, it will cause CNR3 and CMR3 be clear.
[26]	**CH3INV**	**PWM-Timer 3 Output Inverter Enable** (PWM timer 3 for group A and PWM timer 7 for group B) 1 = Inverter enable 0 = Inverter disable
[25]	**Reserved**	Reserved
[24]	**CH3EN**	**PWM-Timer 3 Enable** (PWM timer 3 for group A and PWM timer 7 for group B) 1 = Enable corresponding PWM-Timer Start Run 0 = Stop corresponding PWM-Timer Running
[23:20]	**Reserved**	Reserved
[19]	**CH2MOD**	**PWM-Timer 2 Auto-reload/One-Shot Mode** (PWM timer 2 for group A and PWM timer 6 for group B) 1 = Auto-reload Mode 0 = One-Shot Mode Note: If there is a transition at this bit, it will cause CNR2 and CMR2 be clear.
[18]	**CH2INV**	**PWM-Timer 2 Output Inverter Enable** (PWM timer 2 for group A and PWM timer 6 for

		group B) 1 = Inverter enable 0 = Inverter disable
[17]	**Reserved**	Reserved
[16]	**CH2EN**	**PWM-Timer 2 Enable** (PWM timer 2 for group A and PWM timer 6 for group B) 1 = Enable corresponding PWM-Timer Start Run 0 = Stop corresponding PWM-Timer Running
[15:12]	**Reserved**	Reserved
[11]	**CH1MOD**	**PWM-Timer 1 Auto-reload/One-Shot Mode** (PWM timer 1 for group A and PWM timer 5 for group B) 1 = Auto-load Mode 0 = One-Shot Mode Note: If there is a transition at this bit, it will cause CNR1 and CMR1 be clear.
[10]	**CH1INV**	**PWM-Timer 1 Output Inverter Enable** (PWM timer 1 for group A and PWM timer 5 for group B) 1 = Inverter enable 0 = Inverter disable
[9]	**Reserved**	Reserved
[8]	**CH1EN**	**PWM-Timer 1 Enable** (PWM timer 1 for group A and PWM timer 5 for group B) 1 = Enable corresponding PWM-Timer Start Run 0 = Stop corresponding PWM-Timer Running
[7:6]	**Reserved**	Reserved
[5]	**DZEN23**	**Dead-Zone 2 Generator Enable** (PWM2 and PWM3 pair for PWM group A, PWM6 and PWM7 pair for PWM group B) 1 = Enable 0 = Disable Note: When Dead-Zone Generator is enabled, the pair of PWM2 and PWM3 becomes a complementary pair for PWM group A and the pair of PWM6 and PWM7 becomes a complementary pair for PWM group B.
[4]	**DZEN01**	**Dead-Zone 0 Generator Enable** (PWM0 and PWM1 pair for PWM group A, PWM4 and PWM5 pair for PWM group B) 1 = Enable 0 = Disable Note: When Dead-Zone Generator is enabled, the pair of PWM0 and PWM1 becomes a complementary pair for PWM group A and the pair of PWM4 and PWM5 becomes a complementary pair for PWM group B.
[3]	**CH0MOD**	**PWM-Timer 0 Auto-reload/One-Shot Mode** (PWM timer 0 for group A and PWM timer 4 for group B) 1 = Auto-reload Mode 0 = One-Shot Mode Note: If there is a transition at this bit, it will cause CNR0 and CMR0 be clear.
[2]	**CH0INV**	**PWM-Timer 0 Output Inverter Enable** (PWM timer 0 for group A and PWM timer 4 for group B)

		1 = Inverter enable 0 = Inverter disable
[1]	**Reserved**	Reserved
[0]	**CH0EN**	**PWM-Timer 0 Enable** (PWM timer 0 for group A and PWM timer 4 for group B) 1 = Enable corresponding PWM-Timer Start Run 0 = Stop corresponding PWM-Timer Running

PWM Counter Register 3-0 (CNR3-0)

Register	Offset	R/W	Description	Reset Value
CNR0	PWMA_BA+0x0C	R/W	PWM Group A Counter Register 0	0x0000_0000
	PWMB_BA+0x0C	R/W	PWM Group B Counter Register 0	0x0000_0000
CNR1	PWMA_BA+0x18	R/W	PWM Group A Counter Register 1	0x0000_0000
	PWMB_BA+0x18	R/W	PWM Group B Counter Register 1	0x0000_0000
CNR2	PWMA_BA+0x24	R/W	PWM Group A Counter Register 2	0x0000_0000
	PWMB_BA+0x24	R/W	PWM Group B Counter Register 2	0x0000_0000
CNR3	PWMA_BA+0x30	R/W	PWM Group A Counter Register 3	0x0000_0000
	PWMB_BA+0x30	R/W	PWM Group B Counter Register 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**
CNRx [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNRx [7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	Reserved
[15:0]	**CNRx**	**PWM Timer Loaded Value** CNR determines the PWM period. ● PWM frequency = PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)]; where xy, could be 01, 23, 45 or 67, depends on selected PWM channel. ● Duty ratio = (CMR+1)/(CNR+1). ● CMR >= CNR: PWM output is always high. ● CMR < CNR: PWM low width = (CNR-CMR) unit; PWM high width = (CMR+1) unit. ● CMR = 0: PWM low width = (CNR) unit; PWM high width = 1 unit (Unit = one PWM clock cycle) Note: Any write to CNR will take effect in next PWM cycle.

PWM Comparator Register 3-0 (CMR3-0)

Register	Offset	R/W	Description	Reset Value
CMR0	PWMA_BA+0x10	R/W	PWM Group A Comparator Register 0	0x0000_0000
	PWMB_BA+0x10	R/W	PWM Group B Comparator Register 0	0x0000_0000
CMR1	PWMA_BA+0x1C	R/W	PWM Group A Comparator Register 1	0x0000_0000
	PWMB_BA+0x1C	R/W	PWM Group B Comparator Register 1	0x0000_0000
CMR2	PWMA_BA+0x28	R/W	PWM Group A Comparator Register 2	0x0000_0000
	PWMB_BA+0x28	R/W	PWM Group B Comparator Register 2	0x0000_0000
CMR3	PWMA_BA+0x34	R/W	PWM Group A Comparator Register 3	0x0000_0000
	PWMB_BA+0x34	R/W	PWM Group B Comparator Register 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**
CMRx [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMRx [7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	Reserved
[15:0]	**CMRx**	**PWM Comparator Register** CMR determines the PWM duty. ● PWM frequency = PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)]; where xy, could be 01, 23, 45 or 67, depends on selected PWM channel. ● Duty ratio = (CMR+1)/(CNR+1). ● CMR >= CNR: PWM output is always high. ● CMR < CNR: PWM low width = (CNR-CMR) unit; PWM high width = (CMR+1) unit. ● CMR = 0: PWM low width = (CNR) unit; PWM high width = 1 unit (Unit = one PWM clock cycle) Note: Any write to CMR will take effect in next PWM cycle.

PWM Data Register 3-0 (PDR 3-0)

Register	Offset	R/W	Description	Reset Value
PDR0	PWMA_BA0+0x14	R	PWM Group A Data Register 0	0x0000_0000
	PWMB_BA0+0x14	R	PWM Group B Data Register 0	0x0000_0000
PDR1	PWMA_BA0+0x20	R	PWM Group A Data Register 1	0x0000_0000
	PWMB_BA0+0x20	R	PWM Group B Data Register 1	0x0000_0000
PDR2	PWMA_BA0+0x2C	R	PWM Group A Data Register 2	0x0000_0000
	PWMB_BA0+0x2C	R	PWM Group B Data Register 2	0x0000_0000
PDR3	PWMA_BA0+0x38	R	PWM Group A Data Register 3	0x0000_0000
	PWMB_BA0+0x38	R	PWM Group B Data Register 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**
PDR[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PDR[7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	Reserved
[15:0]	**PDRx**	**PWM Data Register** User can monitor PDR to know the current value in 16-bit down counter.

PWM Backward Compatible Register

Register	Offset	R/W	Description	Reset Value
PBCR	PWMA_BA+0x3C	R/W	PWM Group A backward compatible Register	0x0000_0000
	PWMB_BA+0x3C	R/W	PWM Group B backward compatible 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							**BCn**

Bits	Descriptions	
[31:1]	**Reserved**	Reserved
[0]	**BCn**	**PWM Backward Compatible Register** 0 = configure write 0 to clear CFLRI0~3 and CRLRI0~3. 1 = configure write 1 to clear CFLRI0~3 and CRLRI0~3. Please reference CCR0/CCR2 register bit 6, 7, 22, 23 description

PWM Interrupt Enable Register (PIER)

Register	Offset	R/W	Description	Reset Value
PIER	PWMA_BA+0x40	R/W	PWM Group A Interrupt Enable Register	0x0000_0000
	PWMB_BA+0x40	R/W	PWM Group B 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				**PWMIE3**	**PWMIE2**	**PWMIE1**	**PWMIE0**

Bits	Descriptions	
[31:4]	**Reserved**	Reserved
[3]	**PWMIE3**	**PWM channel 3 Interrupt Enable** 1 = Enable 0 = Disable
[2]	**PWMIE2**	**PWM channel 2 Interrupt Enable** 1 = Enable 0 = Disable
[1]	**PWMIE1**	**PWM channel 1 Interrupt Enable** 1 = Enable 0 = Disable
[0]	**PWMIE0**	**PWM channel 0 Interrupt Enable** 1 = Enable 0 = Disable

PWM Interrupt Indication Register (PIIR)

Register	Offset	R/W	Description	Reset Value
PIIR	PWMA_BA+0x44	R/W	PWM Group A Interrupt Indication Register	0x0000_0000
	PWMB_BA+0x44	R/W	PWM Group B Interrupt Indication 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				**PWMIF3**	**PWMIF2**	**PWMIF1**	**PWMIF0**

Bits	Descriptions	
[31:4]	**Reserved**	Reserved
[3]	**PWMIF3**	**PWM channel 3 Interrupt Status** This bit is set by hardware when PWM3 down counter reaches zero if PWM3 interrupt enable bit (PWMIE3) is 1, software can write 1 to clear this bit to zero
[2]	**PWMIF2**	**PWM channel 2 Interrupt Status** This bit is set by hardware when PWM2 down counter reaches zero if PWM3 interrupt enable bit (PWMIE2) is 1, software can write 1 to clear this bit to zero
[1]	**PWMIF1**	**PWM channel 1 Interrupt Status** This bit is set by hardware when PWM1 down counter reaches zero if PWM3 interrupt enable bit (PWMIE1) is 1, software can write 1 to clear this bit to zero
[0]	**PWMIF0**	**PWM channel 0 Interrupt Status** This bit is set by hardware when PWM0 down counter reaches zero if PWM3 interrupt enable bit (PWMIE0) is 1, software can write 1 to clear this bit to zero

Note: User can clear each interrupt flag by writing 1 to corresponding bit in PIIR.

Capture Control Register (CCR0)

Register	Offset	R/W	Description	Reset Value
CCR0	PWMA_BA+0x50	R/W	PWM Group A Capture Control Register	0x0000_0000
	PWMB_BA+0x50	R/W	PWM Group B Capture Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CFLRI1	**CRLRI1**	**Reserved**	**CAPIF1**	**CAPCH1EN**	**CFL_IE1**	**CRL_IE1**	**INV1**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CFLRI0	**CRLRI0**	**Reserved**	**CAPIF0**	**CAPCH0EN**	**CFL_IE0**	**CRL_IE0**	**INV0**

Bits	Descriptions	
[31:24]	**Reserved**	Reserved
[23]	**CFLRI1**	**CFLR1 Latched Indicator Bit** When PWM group input channel 1 has a falling transition, CFLR1 was latched with the value of PWM down-counter and this bit is set by hardware. Software can write 0 to clear this bit to zero if BCn bit is 0, and can write 1 to clear this bit to zero if BCn bit is 1.
[22]	**CRLRI1**	**CRLR1 Latched Indicator Bit** When PWM group input channel 1 has a rising transition, CRLR1 was latched with the value of PWM down-counter and this bit is set by hardware. Software can write 0 to clear this bit to zero if BCn bit is 0, and can write 1 to clear this bit to zero if BCn bit is 1.
[5]	**Reserved**	Reserved
[20]	**CAPIF1**	**Channel 1 Capture Interrupt Indication Flag** If PWM group channel 1 rising latch interrupt is enabled (CRL_IE1=1), a rising transition occurs at PWM group channel 1 will result in CAPIF1 to high; Similarly, a falling transition will cause CAPIF1 to be set high if PWM group channel 1 falling latch interrupt is enabled (CFL_IE1=1). Write 1 to clear this bit to zero
[19]	**CAPCH1EN**	**Channel 1 Capture Function Enable** 1 = Enable capture function on PWM group channel 1 0 = Disable capture function on PWM group channel 1 When Enable, Capture latched the PWM-counter and saved to CRLR (Rising latch) and CFLR (Falling latch).

		When Disable, Capture does not update CRLR and CFLR, and disable PWM group channel 1 Interrupt.
[18]	**CFL_IE1**	**Channel 1 Falling Latch Interrupt Enable** 1 = Enable falling latch interrupt 0 = Disable falling latch interrupt When Enable, if Capture detects PWM group channel 1 has falling transition, Capture issues an Interrupt.
[17]	**CRL_IE1**	**Channel 1 Rising Latch Interrupt Enable** 1 = Enable rising latch interrupt 0 = Disable rising latch interrupt When Enable, if Capture detects PWM group channel 1 has rising transition, Capture issues an Interrupt.
[16]	**INV1**	**Channel 1 Inverter Enable** 1 = Inverter enable. Reverse the input signal from GPIO before fed to Capture timer 0 = Inverter disable
[15:8]	**Reserved**	Reserved
[7]	**CFLRI0**	**CFLR0 Latched Indicator Bit** When PWM group input channel 0 has a falling transition, CFLR0 was latched with the value of PWM down-counter and this bit is set by hardware. Software can write 0 to clear this bit to zero if BCn bit is 0, and can write 1 to clear this bit to zero if BCn bit is 1.
[6]	**CRLRI0**	**CRLR0 Latched Indicator Bit** When PWM group input channel 0 has a rising transition, CRLR0 was latched with the value of PWM down-counter and this bit is set by hardware. Software can write 0 to clear this bit to zero if BCn bit is 0, and can write 1 to clear this bit to zero if BCn bit is 1.
[5]	**Reserved**	Reserved
[4]	**CAPIF0**	**Channel 0 Capture Interrupt Indication Flag** If PWM group channel 0 rising latch interrupt is enabled (CRL_IE0=1), a rising transition occurs at PWM group channel 0 will result in CAPIF0 to high; Similarly, a falling transition will cause CAPIF0 to be set high if PWM group channel 0 falling latch interrupt is enabled (CFL_IE0=1). Write 1 to clear this bit to zero
[3]	**CAPCH0EN**	**Channel 0 Capture Function Enable** 1 = Enable capture function on PWM group channel 0. 0 = Disable capture function on PWM group channel 0 When Enable, Capture latched the PWM-counter value and saved to CRLR (Rising latch) and CFLR (Falling latch). When Disable, Capture does not update CRLR and CFLR, and disable PWM group channel 0 Interrupt.
[2]	**CFL_IE0**	**Channel 0 Falling Latch Interrupt Enable** 1 = Enable falling latch interrupt 0 = Disable falling latch interrupt When Enable, if Capture detects PWM group channel 0 has falling transition, Capture

		issues an Interrupt.
[1]	**CRL_IE0**	**Channel 0 Rising Latch Interrupt Enable** 1 = Enable rising latch interrupt 0 = Disable rising latch interrupt When Enable, if Capture detects PWM group channel 0 has rising transition, Capture issues an Interrupt.
[0]	**INV0**	**Channel 0 Inverter Enable** 1 = Inverter enable. Reverse the input signal from GPIO before fed to Capture timer 0 = Inverter disable

Capture Control Register (CCR2)

Register	Offset	R/W	Description	Reset Value
CCR2	PWMA_BA+0x54	R/W	PWM Group A Capture Control Register	0x0000_0000
	PWMB_BA+0x54	R/W	PWM Group B Capture Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CFLRI3	CRLRI3	Reserved	CAPIF3	CAPCH3EN	CFL_IE3	CRL_IE3	INV3
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CFLRI2	CRLRI2	Reserved	CAPIF2	CAPCH2EN	CFL_IE2	CRL_IE2	INV2

Bits	Descriptions	
[31:24]	**Reserved**	Reserved
[23]	**CFLRI3**	**CFLR3 Latched Indicator Bit** When PWM group input channel 3 has a falling transition, CFLR3 was latched with the value of PWM down-counter and this bit is set by hardware. Software can write 0 to clear this bit to zero if BCn bit is 0, and can write 1 to clear this bit to zero if BCn bit is 1.
[22]	**CRLRI3**	**CRLR3 Latched Indicator Bit** When PWM group input channel 3 has a rising transition, CRLR3 was latched with the value of PWM down-counter and this bit is set by hardware. Software can write 0 to clear this bit to zero if BCn bit is 0, and can write 1 to clear this bit to zero if BCn bit is 1.
[21]	**Reserved**	Reserved
[20]	**CAPIF3**	**Channel 3 Capture Interrupt Indication Flag** If PWM group channel 3 rising latch interrupt is enabled (CRL_IE3=1), a rising transition occurs at PWM group channel 3 will result in CAPIF3 to high; Similarly, a falling transition will cause CAPIF3 to be set high if PWM group channel 3 falling latch interrupt is enabled (CFL_IE3=1). Write 1 to clear this bit to zero
[19]	**CAPCH3EN**	**Channel 3 Capture Function Enable** 1 = Enable capture function on PWM group channel 3 0 = Disable capture function on PWM group channel 3 When Enable, Capture latched the PWM-counter and saved to CRLR (Rising latch) and CFLR (Falling latch). When Disable, Capture does not update CRLR and CFLR, and disable PWM group

		channel 3 Interrupt.
[18]	**CFL_IE3**	**Channel 3 Falling Latch Interrupt Enable** 1 = Enable falling latch interrupt 0 = Disable falling latch interrupt When Enable, if Capture detects PWM group channel 3 has falling transition, Capture issues an Interrupt.
[17]	**CRL_IE3**	**Channel 3 Rising Latch Interrupt Enable** 1 = Enable rising latch interrupt 0 = Disable rising latch interrupt When Enable, if Capture detects PWM group channel 3 has rising transition, Capture issues an Interrupt.
[16]	**INV3**	**Channel 3 Inverter Enable** 1 = Inverter enable. Reverse the input signal from GPIO before fed to Capture timer 0 = Inverter disable
[15:8]	**Reserved**	Reserved
[7]	**CFLRI2**	**CFLR2 Latched Indicator Bit** When PWM group input channel 2 has a falling transition, CFLR2 was latched with the value of PWM down-counter and this bit is set by hardware. Software can write 0 to clear this bit to zero if BCn bit is 0, and can write 1 to clear this bit to zero if BCn bit is 1.
[6]	**CRLRI2**	**CRLR2 Latched Indicator Bit** When PWM group input channel 2 has a rising transition, CRLR2 was latched with the value of PWM down-counter and this bit is set by hardware. Software can write 0 to clear this bit to zero if BCn bit is 0, and can write 1 to clear this bit to zero if BCn bit is 1.
[5]	**Reserved**	Reserved
[4]	**CAPIF2**	**Channel 2 Capture Interrupt Indication Flag** If PWM group channel 2 rising latch interrupt is enabled (CRL_IE2=1), a rising transition occurs at PWM group channel 2 will result in CAPIF2 to high; Similarly, a falling transition will cause CAPIF2 to be set high if PWM group channel 2 falling latch interrupt is enabled (CFL_IE2=1). Write 1 to clear this bit to zero
[3]	**CAPCH2EN**	**Channel 2 Capture Function Enable** 1 = Enable capture function on PWM group channel 2 0 = Disable capture function on PWM group channel 2 When Enable, Capture latched the PWM-counter value and saved to CRLR (Rising latch) and CFLR (Falling latch). When Disable, Capture does not update CRLR and CFLR, and disable PWM group channel 2 Interrupt.
[2]	**CFL_IE2**	**Channel 2 Falling Latch Interrupt Enable** 1 = Enable falling latch interrupt 0 = Disable falling latch interrupt When Enable, if Capture detects PWM group channel 2 has falling transition, Capture issues an Interrupt.

[1]	**CRL_IE2**	**Channel 2 Rising Latch Interrupt Enable** 1 = Enable rising latch interrupt 0 = Disable rising latch interrupt When Enable, if Capture detects PWM group channel 2 has rising transition, Capture issues an Interrupt.
[0]	**INV2**	**Channel 2 Inverter Enable** 1 = Inverter enable. Reverse the input signal from GPIO before fed to Capture timer 0 = Inverter disable

Capture Rising Latch Register3-0 (CRLR3-0)

Register	Offset	R/W	Description	Reset Value
CRLR0	PWMA_BA+0x58	R	PWM Group A Capture Rising Latch Register (channel 0)	0x0000_0000
	PWMB_BA+0x58	R	PWM Group B Capture Rising Latch Register (channel 0)	0x0000_0000
CRLR1	PWMA_BA+0x60	R	PWM Group A Capture Rising Latch Register (channel 1)	0x0000_0000
	PWMB_BA+0x60	R	PWM Group B Capture Rising Latch Register (channel 1)	0x0000_0000
CRLR2	PWMA_BA+0x68	R	PWM Group A Capture Rising Latch Register (channel 2)	0x0000_0000
	PWMB_BA+0x68	R	PWM Group B Capture Rising Latch Register (channel 2)	0x0000_0000
CRLR3	PWMA_BA+0x70	R	PWM Group A Capture Rising Latch Register (channel 3)	0x0000_0000
	PWMB_BA+0x70	R	PWM Group B Capture Rising Latch Register (channel 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**
CRLRx [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CRLRx [7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	Reserved
[15:0]	**CRLRx**	**Capture Rising Latch Register** Latch the PWM counter when Channel 0/1/2/3 has rising transition.

Capture Falling Latch Register3-0 (CFLR3-0)

Register	Offset	R/W	Description	Reset Value
CFLR0	PWMA_BA+0x5C	R	PWM Group A Capture Falling Latch Register (channel 0)	0x0000_0000
	PWMB_BA+0x5C	R	PWM Group B Capture Falling Latch Register (channel 0)	0x0000_0000
CFLR1	PWMA_BA+0x64	R	PWM Group A Capture Falling Latch Register (channel 1)	0x0000_0000
	PWMB_BA+0x64	R	PWM Group B Capture Falling Latch Register (channel 1)	0x0000_0000
CFLR2	PWMA_BA+0x6C	R	PWM Group A Capture Falling Latch Register (channel 2)	0x0000_0000
	PWMB_BA+0x6C	R	PWM Group B Capture Falling Latch Register (channel 2)	0x0000_0000
CFLR3	PWMA_BA+0x74	R	PWM Group A Capture Falling Latch Register (channel 3)	0x0000_0000
	PWMB_BA+0x74	R	PWM Group B Capture Falling Latch Register (channel 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**
CFLRx [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CFLRx [7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	Reserved
[15:0]	**CFLRx**	**Capture Falling Latch Register** Latch the PWM counter when Channel 01/2/3 has Falling transition.

Capture Input Enable Register (CAPENR)

Register	Offset	R/W	Description	Reset Value
CAPENR	PWMA_BA+0x78	R/W	PWM Group A Capture Input 0~3 Enable Register	0x0000_0000
	PWMB_BA+0x78	R/W	PWM Group B Capture Input 0~3 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				**CAPENR**			

Bits	Descriptions	
[3:0]	**CAPENR**	**Capture Input Enable Register** There are four capture inputs from pad. Bit0~Bit3 are used to control each input enable or disable. 0 = Disable (PWMx multi-function pin input does not affect input capture function.) 1 = Enable (PWMx multi-function pin input will affect its input capture function.) CAPENR Bit 3210 for PWM group A Bit xxx1 ➔ Capture channel 0 is from pin PA.12 Bit xx1x ➔ Capture channel 1 is from pin PA.13 Bit x1xx ➔ Capture channel 2 is from pin PA.14 Bit 1xxx ➔ Capture channel 3 is from pin PA.15 Bit 3210 for PWM group B Bit xxx1 ➔ Capture channel 0 is from pin PB.11 Bit xx1x ➔ Capture channel 1 is from pin PE.5 Bit x1xx ➔ Capture channel 2 is from pin PE.0 Bit 1xxx ➔ Capture channel 3 is from pin PE.1

PWM Output Enable Register (POE)

Register	Offset	R/W	Description	Reset Value
POE	PWMA_BA+0x7C	R/W	PWM Group A Output Enable Register for channel 0~3	0x0000_0000
	PWMB_BA+0x7C	R/W	PWM Group B Output Enable Register for channel 0~3	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**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**PWM3**	**PWM2**	**PWM1**	**PWM0**

Bits	Descriptions	
[3]	**PWM3**	**Channel 3 Output Enable Register** 1 = Enable PWM channel 3 output to pin 0 = Disable PWM channel 3 output to pin Note: The corresponding GPIO pin also must be switched to PWM function
[2]	**PWM2**	**Channel 2 Output Enable Register** 1 = Enable PWM channel 2 output to pin 0 = Disable PWM channel 2 output to pin Note: The corresponding GPIO pin also must be switched to PWM function
[1]	**PWM1**	**Channel 1 Output Enable Register** 1 = Enable PWM channel 1 output to pin 0 = Disable PWM channel 1 output to pin Note: The corresponding GPIO pin also must be switched to PWM function
[0]	**PWM0**	**Channel 0 Output Enable Register** 1 = Enable PWM channel 0 output to pin 0 = Disable PWM channel 0 output to pin Note: The corresponding GPIO pin also must be switched to PWM function

5.8 Real Time Clock (RTC)

5.8.1 Overview

Real Time Clock (RTC) controller provides user the real time and calendar message. The clock source of RTC is from an external 32.768 kHz low speed crystal connected at pins X32I and X32O (reference to pin descriptions) or from an external 32.768 kHz low speed oscillator output fed at pin X32I. The RTC controller provides the time message (second, minute, hour) in Time Loading Register (TLR) as well as calendar message (day, month, year) in Calendar Loading Register (CLR). The data message is expressed in BCD format. It also offers alarm function that user can preset the alarm time in Time Alarm Register (TAR) and alarm calendar in Calendar Alarm Register (CAR).

The RTC controller supports periodic Time Tick and Alarm Match interrupts. The periodic interrupt has 8 period options 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second which are selected by TTR (TTR[2:0]). When RTC counter in TLR and CLR is equal to alarm setting time registers TAR and CAR, the alarm interrupt flag (RIIR.AIF) is set and the alarm interrupt is requested if the alarm interrupt is enabled (RIER.AIER=1). Both RTC Time Tick and Alarm Match can cause chip wake-up from power down mode if wake-up function is enabled (TWKE (TTR[3])=1).

5.8.2 Features

- There is a time counter (second, minute, hour) and calendar counter (day, month, year) for user to check the time
- Alarm register (second, minute, hour, day, month, year)
- 12-hour or 24-hour mode is selectable
- Leap year compensation automatically
- Day of week counter
- Frequency compensate register (FCR)
- All time and calendar message is expressed in BCD code
- Support 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
- Support RTC Time Tick and Alarm Match interrupt
- Support wake-up chip from power down mode

5.8.3 Block Diagram

The block diagram of Real Time Clock is depicted as following:

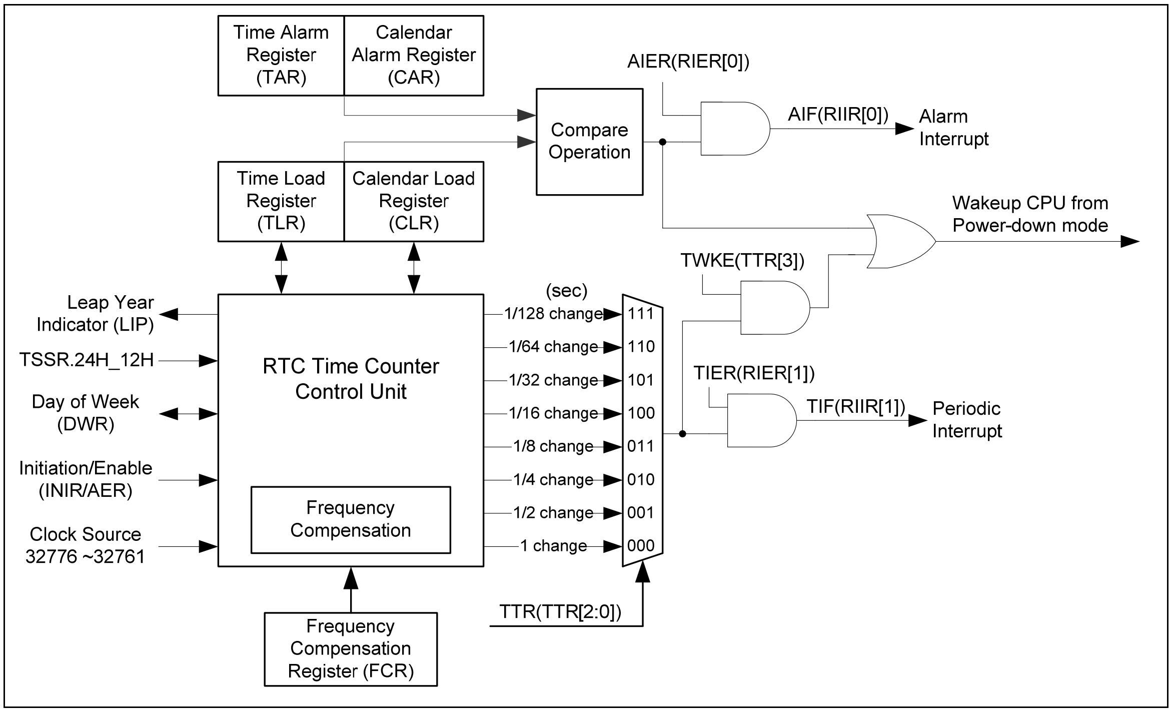


Figure 5-49 RTC Block Diagram

5.8.4 Function Description

5.8.4.1 Access to RTC register

Due to clock difference between RTC clock and system clock, when user write new data to any one of the registers, the register will not be updated until 2 RTC clocks later (60us).

In addition, user must be aware that RTC controller does not check whether loaded data is out of bounds or not. RTC does not check rationality between DWR and CLR either.

5.8.4.2 RTC Initiation

When RTC block is power on, RTC is at reset state. User has to write a number (0xa5eb1357) to INIR to make RTC leaving reset state. Once the INIR is written as 0xa5eb1357, the RTC will be in un-reset state permanently.

5.8.4.3 RTC Read/Write Enable

Register AER bit 15~0 is served as RTC read/write password to protect RTC registers. AER bit 15~0 has to be set as 0xA965 to enable access restriction. Once it is set, it will take effect at least 512 RTC clocks (about 15ms). Programmer can read RTC enabled status flag in AER.ENF to check whether if RTC controller starts operating or not.

5.8.4.4 Frequency Compensation

The RTC FCR allows software to make digital compensation to a clock input. The frequency of clock input must be in the range from 32776 Hz to 32761 Hz. User can utilize a frequency counter to measure RTC clock on one of GPIO pin during manufacture, and store the value in Flash memory for retrieval when the product is first power on. Following are the compensation examples for higher or lower frequency clock input.

Example 1:

Frequency counter measurement : 32773.65 Hz (> 32768 Hz)

Integer part: 32773 => 0x8005

FCR.Integer = 0x05 – 0x01 + 0x08 = 0x0c

Fraction part: 0.65 x 60 = 39 => 0x27

FCR.Fraction = 0x27

Example 2

Frequency counter measurement : 32765.27 Hz (≦ 32768 Hz)

Integer part: 32765 => 0x7FFD

FCR.Integer = 0x0A – 0x01 – 0x08 = 0x04

Fraction part: 0.27 x 60 = 16.2=> 0x10

FCR.Fraction = 0x10

5.8.4.5 Time and Calendar counter

TLR and CLR are used to load the time and calendar. TAR and CAR are used for alarm. They are all represented by BCD.

5.8.4.6 12/24 hour Time Scale Selection

The 12/24 hour time scale selection depends on TSSR bit 0.

5.8.4.7 *Day of the week counter*

The RTC controller provides day of week in Day of the Week Register (DWR). The value is defined from 0 to 6 to represent Sunday to Saturday respectively.

5.8.4.8 *Periodic Time Tick Interrupt*

The periodic interrupt has 8 period option 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second which are selected by TTR.TTR[2:0]. When periodic time tick interrupt is enabled by setting RIER.TIER to 1, the Periodic Time Tick Interrupt is requested periodically in the period selected by TTR register.

5.8.4.9 *Alarm interrupt*

When RTC counter in TLR and CLR is equal to alarm setting time TAR and CAR the alarm interrupt flag (RIIR.AIF) is set and the alarm interrupt is requested if the alarm interrupt is enabled (RIER.AIER=1).

5.8.4.10 *Application note:*

1. TAR, CAR, TLR and CLR registers are all BCD counter.
2. Programmer has to make sure that the loaded values are reasonable. For example, Load CLR as 201a (year), 13 (month), 00 (day), or CLR does not match with DWR, etc.
3. Reset state :

Register	Reset State
AER	0
CLR	05/1/1 (year/month/day)
TLR	00:00:00 (hour : minute : second)
CAR	00/00/00 (year/month/day)
TAR	00:00:00 (hour : minute : second)
TSSR	1 (24 hr mode)
DWR	6 (Saturday)
RIER	0
RIIR	0
LIR	0
TTR	0

4. In TLR and TAR, only 2 BCD digits are used to express “year”. We assume 2 BCD digits of xY denote 20xY, but not 19xY or 21xY.

5.8.5 Register Map

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

Register	**Offset**	**R/W**	**Description**	**Reset Value**
RTC_BA = 0x4000_8000				
INIR	RTC_BA+0x00	R/W	RTC Initiation Register	0x0000_0000
AER	RTC_BA+0x04	R/W	RTC Access Enable Register	0x0000_0000
FCR	RTC_BA+0x08	R/W	RTC Frequency Compensation Register	0x0000_0700
TLR	RTC_BA+0x0C	R/W	Time Loading Register	0x0000_0000
CLR	RTC_BA+0x10	R/W	Calendar Loading Register	0x0005_0101
TSSR	RTC_BA+0x14	R/W	Time Scale Selection Register	0x0000_0001
DWR	RTC_BA+0x18	R/W	Day of the Week Register	0x0000_0006
TAR	RTC_BA+0x1C	R/W	Time Alarm Register	0x0000_0000
CAR	RTC_BA+0x20	R/W	Calendar Alarm Register	0x0000_0000
LIR	RTC_BA+0x24	R	Leap year Indicator Register	0x0000_0000
RIER	RTC_BA+0x28	R/W	RTC Interrupt Enable Register	0x0000_0000
RIIR	RTC_BA+0x2C	R/W	RTC Interrupt Indicator Register	0x0000_0000
TTR	RTC_BA+0x30	R/W	RTC Time Tick Register	0x0000_0000

5.8.6 Register Description

RTC Initiation Register (INIR)

Register	Offset	R/W	Description	Reset Value
INIR	RTC_BA+0x00	R/W	RTC Initiation Register	0x0000_0000

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

Bits	Descriptions	
[31:0]	**INIR**	**RTC Initiation** When RTC block is power on, RTC is at reset state. User has to write a number (0x a5eb1357) to INIR to make RTC leaving reset state. Once the INIR is written as 0xa5eb1357, the RTC will be in un-reset state permanently.
[0]	**Active**	**RTC Active Status (Read only)** 0 = RTC is at reset state 1 = RTC is at normal active state.

RTC Access Enable Register (AER)

Register	Offset	R/W	Description	Reset Value
AER	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							**ENF**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
AER							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
AER							

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[31:17]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[16]</td><td>ENF</td><td>RTC Register Access Enable Flag (Read only)
1 = RTC register read/write enable
0 = RTC register read/write disable
This bit will be set after AER[15:0] register is load a 0xA965, and be clear automatically 512 RTC clock or AER[15:0] is not 0xA965.
<table>
<tr><th>Register AER.ENF</th><th>1</th><th>0</th></tr>
<tr><td>INIR</td><td>R/W</td><td>R/W</td></tr>
<tr><td>AER</td><td>R/W</td><td>R/W</td></tr>
<tr><td>FCR</td><td>R/W</td><td>-</td></tr>
<tr><td>TLR</td><td>R/W</td><td>R</td></tr>
<tr><td>CLR</td><td>R/W</td><td>R</td></tr>
<tr><td>TSSR</td><td>R/W</td><td>R/W</td></tr>
<tr><td>DWR</td><td>R/W</td><td>R</td></tr>
<tr><td>TAR</td><td>R/W</td><td>-</td></tr>
<tr><td>CAR</td><td>R/W</td><td>-</td></tr>
<tr><td>LIR</td><td>R</td><td>R</td></tr>
<tr><td>RIER</td><td>R/W</td><td>R/W</td></tr>
<tr><td>RIIR</td><td>R/W</td><td>R/W</td></tr>
<tr><td>TTR</td><td>R/W</td><td>-</td></tr>
</table>
</td></tr>
</table>

[15:0]	**AER**	**RTC Register Access Enable Password (Write only)** 0xA965 = Enable RTC access Others = Disable RTC access

RTC Frequency Compensation Register (FCR)

Register	Offset	R/W	Description	Reset Value
FCR	RTC_BA+0x08	R/W	Frequency Compensation Register	0x0000_0700

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th></tr>
<tr><td colspan="8">Reserved</td></tr>
<tr><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="8">Reserved</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th></tr>
<tr><td colspan="4">Reserved</td><td colspan="4">INTEGER</td></tr>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="2">Reserved</td><td colspan="6">FRACTION</td></tr>
</table>

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[31:12]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[11:8]</td><td>INTEGER</td><td>Integer Part
<table>
<tr><th>Integer part of detected value</th><th>FCR[11:8]</th><th>Integer part of detected value</th><th>FCR[11:8]</th></tr>
<tr><td>32776</td><td>1111</td><td>32768</td><td>0111</td></tr>
<tr><td>32775</td><td>1110</td><td>32767</td><td>0110</td></tr>
<tr><td>32774</td><td>1101</td><td>32766</td><td>0101</td></tr>
<tr><td>32773</td><td>1100</td><td>32765</td><td>0100</td></tr>
<tr><td>32772</td><td>1011</td><td>32764</td><td>0011</td></tr>
<tr><td>32771</td><td>1010</td><td>32763</td><td>0010</td></tr>
<tr><td>32770</td><td>1001</td><td>32762</td><td>0001</td></tr>
<tr><td>32769</td><td>1000</td><td>32761</td><td>0000</td></tr>
</table></td></tr>
<tr><td>[5:0]</td><td>FRACTION</td><td>Fraction Part
Formula = (fraction part of detected value) x 60
Note: Digit in FCR must be expressed as hexadecimal number. Refer to 5.8.4.4 for the examples.</td></tr>
</table>

Note: This register can be read back after the RTC register access enable bit ENF (AER[16]) is active.

RTC Time Loading Register (TLR)

Register	Offset	R/W	Description	Reset Value
TLR	RTC_BA+0x0C	R/W	Time Loading Register	0x0000_0000

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

Bits	Descriptions	
[31:22]	**Reserved**	Reserved
[21:20]	**10HR**	10-Hour Time Digit (0~2)
[19:16]	**1HR**	1-Hour Time Digit (0~9)
[15]	**Reserved**	Reserved
[14:12]	**10MIN**	10-Min Time Digit (0~5)
[11:8]	**1MIN**	1-Min Time Digit (0~9)
[7]	**Reserved**	Reserved
[6:4]	**10SEC**	10-Sec Time Digit (0~5)
[3:0]	**1SEC**	1-Sec Time Digit (0~9)

Note:

1. TLR 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 (CLR)

Register	Offset	R/W	Description	Reset Value
CLR	RTC_BA+0x10	R/W	Calendar Loading Register	0x0005_0101

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

Bits	Descriptions	
[31:24]	**Reserved**	Reserved
[23:20]	**10YEAR**	10-Year Calendar Digit (0~9)
[19:16]	**1YEAR**	1-Year Calendar Digit (0~9)
[15:13]	**Reserved**	Reserved
[12]	**10MON**	10-Month Calendar Digit (0~1)
[11:8]	**1MON**	1-Month Calendar Digit (0~9)
[7:6]	**Reserved**	Reserved
[5:4]	**10DAY**	10-Day Calendar Digit (0~3)
[3:0]	**1DAY**	1-Day Calendar Digit (0~9)

Note:

1. CLR 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 (TSSR)

Register	Offset	R/W	Description	Reset Value
TSSR	RTC_BA+0x14	R/W	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							**24H_12H**

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[31:1]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[0]</td><td>24H_12H</td><td>24-Hour / 12-Hour Time Scale Selection
It indicate that TLR and TAR are in 24-hour time scale or 12-hour time scale
1 = select 24-hour time scale
0 = select 12-hour time scale with AM and PM indication
<table>
<tr><th>24-hour time scale</th><th>12-hour time scale</th><th>24-hour time scale</th><th>12-hour time scale (PM time + 20)</th></tr>
<tr><td>00</td><td>12(AM12)</td><td>12</td><td>32(PM12)</td></tr>
<tr><td>01</td><td>01 (AM01)</td><td>13</td><td>21 (PM01)</td></tr>
<tr><td>02</td><td>02(AM02)</td><td>14</td><td>22(PM02)</td></tr>
<tr><td>03</td><td>03(AM03)</td><td>15</td><td>23(PM03)</td></tr>
<tr><td>04</td><td>04 (AM04)</td><td>16</td><td>24 (PM04)</td></tr>
<tr><td>05</td><td>05(AM05)</td><td>17</td><td>25(PM05)</td></tr>
<tr><td>06</td><td>06(AM06)</td><td>18</td><td>26(PM06)</td></tr>
<tr><td>07</td><td>07(AM07)</td><td>19</td><td>27(PM07)</td></tr>
<tr><td>08</td><td>08(AM08)</td><td>20</td><td>28(PM08)</td></tr>
<tr><td>09</td><td>09(AM09)</td><td>21</td><td>29(PM09)</td></tr>
<tr><td>10</td><td>10 (AM10)</td><td>22</td><td>30 (PM10)</td></tr>
<tr><td>11</td><td>11 (AM11)</td><td>23</td><td>31 (PM11)</td></tr>
</table>
</td></tr>
</table>

RTC Day of the Week Register (DWR)

Register	**Offset**	**R/W**	**Description**	**Reset Value**
DWR	RTC_BA+0x18	R/W	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					**DWR**		

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[31:3]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[2:0]</td><td>DWR</td><td>Day of the Week Register
<table>
<tr><th>Value</th><th>Day of the Week</th></tr>
<tr><td>0</td><td>Sunday</td></tr>
<tr><td>1</td><td>Monday</td></tr>
<tr><td>2</td><td>Tuesday</td></tr>
<tr><td>3</td><td>Wednesday</td></tr>
<tr><td>4</td><td>Thursday</td></tr>
<tr><td>5</td><td>Friday</td></tr>
<tr><td>6</td><td>Saturday</td></tr>
</table></td></tr>
</table>

RTC Time Alarm Register (TAR)

Register	Offset	R/W	Description	Reset Value
TAR	RTC_BA+0x1C	R/W	Time Alarm Register	0x0000_0000

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

Bits	Descriptions	
[31:22]	**Reserved**	Reserved
[21:20]	**10HR**	10-Hour Time Digit of Alarm Setting (0~2)
[19:16]	**1HR**	1-Hour Time Digit of Alarm Setting (0~9)
[15]	**Reserved**	Reserved
[14:12]	**10MIN**	10-Min Time Digit of Alarm Setting (0~5)
[11:8]	**1MIN**	1-Min Time Digit of Alarm Setting (0~9)
[7]	**Reserved**	Reserved
[6:4]	**10SEC**	10-Sec Time Digit of Alarm Setting (0~5)
[3:0]	**1SEC**	1-Sec Time Digit of Alarm Setting (0~9)

Note:

1. TAR 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 ENF (AER[16]) is active.

RTC Calendar Alarm Register (CAR)

Register	Offset	R/W	Description	Reset Value
CAR	RTC_BA+0x20	R/W	Calendar Alarm Register	0x0000_0000

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

Bits	Descriptions	
[31:24]	**Reserved**	Reserved
[23:20]	**10YEAR**	10-Year Calendar Digit of Alarm Setting (0~9)
[19:16]	**1YEAR**	1-Year Calendar Digit of Alarm Setting (0~9)
[15:13]	**Reserved**	Reserved
[12]	**10MON**	10-Month Calendar Digit of Alarm Setting (0~1)
[11:8]	**1MON**	1-Month Calendar Digit of Alarm Setting (0~9)
[7:6]	**Reserved**	Reserved
[5:4]	**10DAY**	10-Day Calendar Digit of Alarm Setting (0~3)
[3:0]	**1DAY**	1-Day Calendar Digit of Alarm Setting (0~9)

Note:

1. CAR 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 ENF (AER[16]) is active.

RTC Leap year Indication Register (LIR)

Register	Offset	R/W	Description	Reset Value
LIR	RTC_BA+0x24	R	RTC Leap year Indication 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							**LIR**

Bits	Descriptions	
[31:1]	**Reserved**	Reserved
[0]	**LIR**	**Leap Year Indication REGISTER (Real only).** 1 = It indicate that this year is leap year 0 = It indicate that this year is not a leap year

RTC Interrupt Enable Register (RIER)

Register	Offset	R/W	Description	Reset Value
RIER	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						**TIER**	**AIER**

Bits	Descriptions	
[31:2]	**Reserved**	Reserved
[1]	**TIER**	**Time Tick Interrupt Enable** 1 = RTC Time Tick Interrupt is enabled 0 = RTC Time Tick Interrupt is disabled
[0]	**AIER**	**Alarm Interrupt Enable** 1 = RTC Alarm Interrupt is enabled 0 = RTC Alarm Interrupt is disabled

RTC Interrupt Indication Register (RIIR)

Register	Offset	R/W	Description	Reset Value
RIIR	RTC_BA+0x2C	R/W	RTC Interrupt Indication 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						**TIF**	**AIF**

Bits	Descriptions	
[31:2]	**Reserved**	Reserved
[1]	**TIF**	**RTC Time Tick Interrupt Flag** When RTC Time Tick Interrupt is enabled (RIER.TIER=1), RTC controller will set TIF to high periodically in the period selected by TTR[2:0]. This bit is software clear by writing 1 to it. 1= Indicates RTC Time Tick Interrupt is requested if RIER.TIER=1 0= Indicates RCT Time Tick Interrupt condition never occurred.
[0]	**AIF**	**RTC Alarm Interrupt Flag** When RTC Alarm Interrupt is enabled (RIER.AIER=1), RTC controller will set AIF to high once the RTC real time counters TLR and CLR reach the alarm setting time registers TAR and CAR. This bit is software clear by writing 1 to it. 1= Indicates RTC Alarm Interrupt is requested if RIER.AIER=1 0= Indicates RCT Alarm Interrupt condition never occurred.

RTC Time Tick Register (TTR)

Register	Offset	R/W	Description	Reset Value
TTR	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				**TWKE**	**TTR[2:0]**		

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[31:4]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[3]</td><td>TWKE</td><td>RTC Timer Wake-up Function Enable
If TWKE is set before chip is in power down mode, chip will be wakened up by RTC controller when a RTC Time Tick occurs, The chip can also be wakened up by alarm match occur.
1 = Enable RTC Timer wake-up function that chip can be waken up from power down mode by Time Tick or Alarm Match.
0 = Disable RTC Timer wake-up function.
Note: Tick timer setting follows TTR[2:0] description.</td></tr>
<tr><td>[2:0]</td><td>TTR</td><td>Time Tick Register
The RTC time tick period for Periodic Time Tick Interrupt request.
<table>
<tr><th>TTR[2:0]</th><th>Time tick (second)</th></tr>
<tr><td>0</td><td>1</td></tr>
<tr><td>1</td><td>1/2</td></tr>
<tr><td>2</td><td>1/4</td></tr>
<tr><td>3</td><td>1/8</td></tr>
<tr><td>4</td><td>1/16</td></tr>
<tr><td>5</td><td>1/32</td></tr>
<tr><td>6</td><td>1/64</td></tr>
<tr><td>7</td><td>1/128</td></tr>
</table>
Note: This register can be read back after the RTC register access enable bit ENF (AER[16]) is active.</td></tr>
</table>

5.9 Serial Peripheral Interface (SPI)

5.9.1 Overview

The Serial Peripheral Interface (SPI) is a synchronous serial data communication protocol which operates in full duplex mode. Devices communicate in master/slave mode with 4-wire bi-direction interface. The NuMicro™ NUC130/NUC140 contains up to four sets of SPI controller 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 set of SPI controller can be set as a master, it also can be configured as a slave device controlled by an off-chip master device.

This controller supports a variable serial clock for special application and it also supports 2-bit transfer mode to connect 2 off-chip slave devices at the same time. The SPI controller also supports PDMA function to access the data buffer.

5.9.2 Features

- Up to four sets of SPI controller
- Support master or slave mode operation
- Support 1-bit or 2-bit transfer mode
- Configurable bit length up to 32-bit of a transfer word and configurable word numbers up to 2 of a transaction, so the maximum bit length is 64-bit for each data transfer
- Provide burst mode operation, transmit/receive can be transferred up to two times word transaction in one transfer
- Support MSB or LSB first transfer
- 2 device/slave select lines in master mode, but 1 device/slave select line in slave mode
- Support byte reorder in data register
- Support byte or word suspend mode
- Variable output serial clock frequency in master mode
- Support two programmable serial clock frequencies in master mode
- Support two channel PDMA request, one for transmitter and another for receiver
- Support FIFO mode
- Support three wire, no slave select signal, bi-direction interface
- The SPI clock rate can be configured to equal the system clock rate

5.9.3 Block Diagram

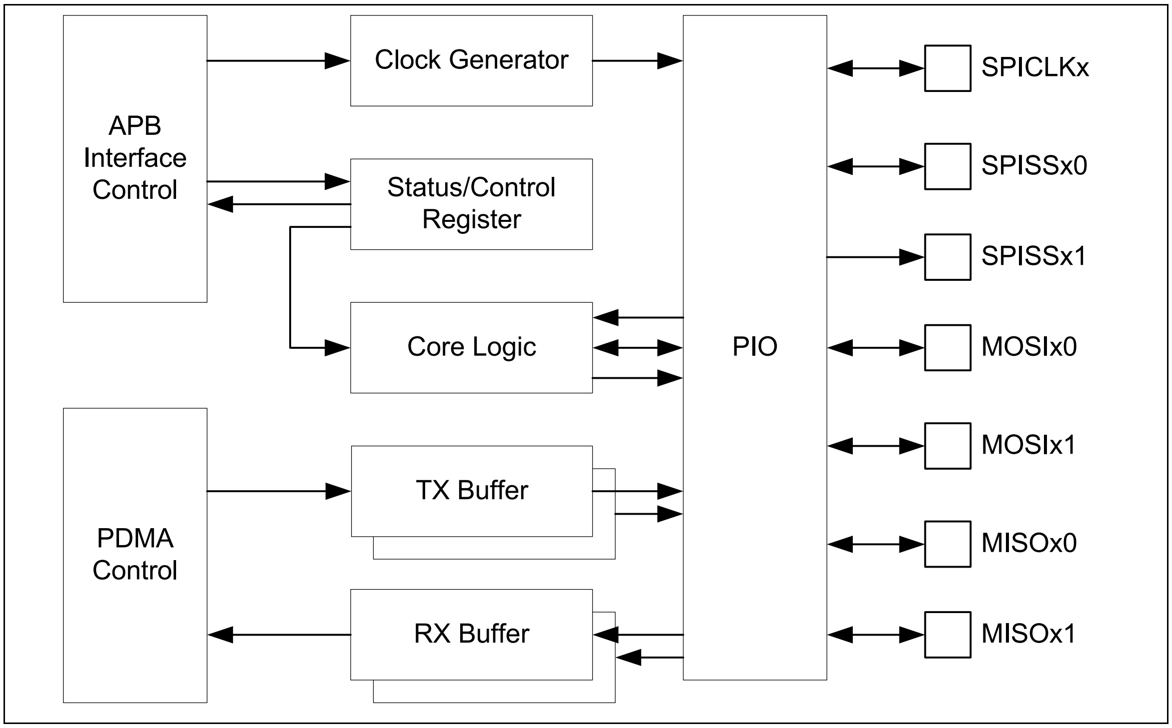


Figure 5-50 SPI Block Diagram

5.9.4 Function Description

Master/Slave Mode

This SPI controller can be set as master or slave mode by setting the SLAVE bit (SPI_CNTRL[18]) to communicate with the off-chip SPI slave or master device. The application block diagrams in master and slave mode are shown as below.

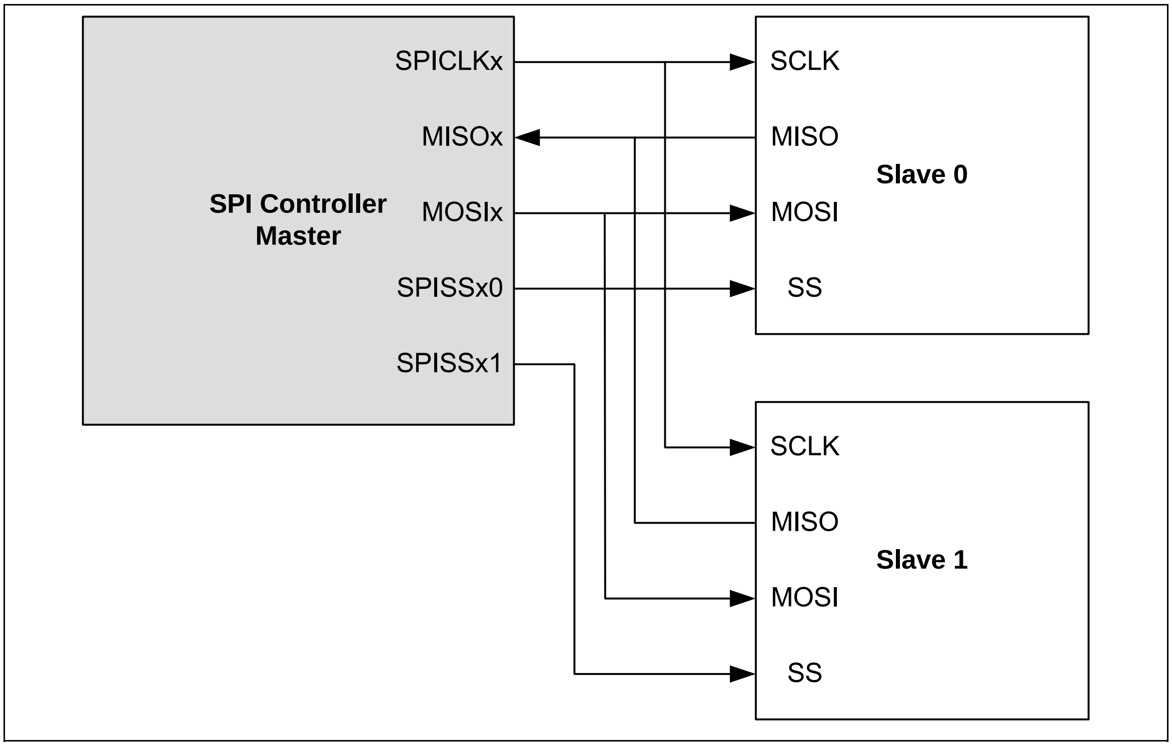


Figure 5-51 SPI Master Mode Application Block Diagram

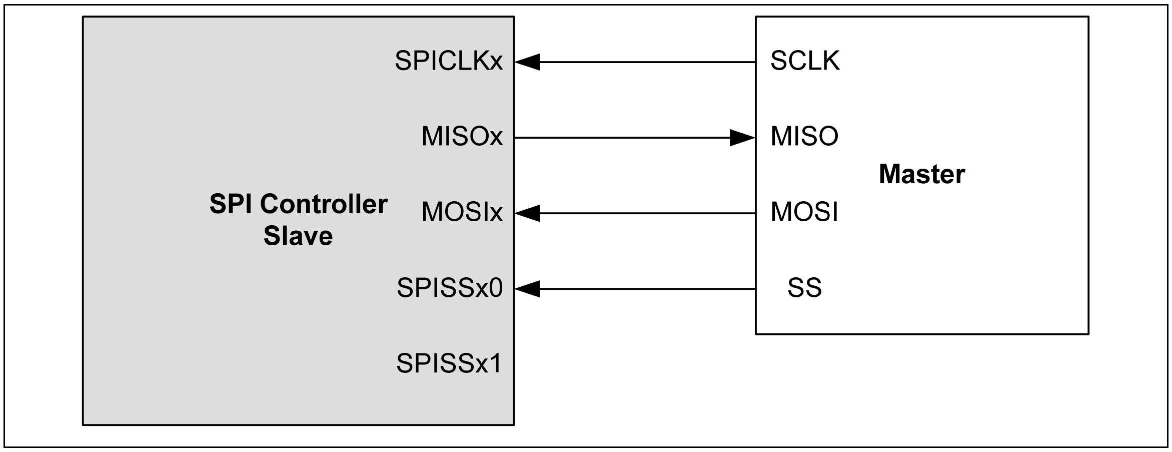


Figure 5-52 SPI Slave Mode Application Block Diagram

Slave Select

In master mode, this SPI controller can drive up to two off-chip slave devices through the slave select output pins SPISSx0 and SPISSx1. In slave mode, the off-chip master device drives the slave select signal from the SPISSx0 input port to this SPI controller. In master/slave mode, the active state of slave select signal can be programmed to low active or high active in SS_LVL bit (SPI_SSR[2]), and the SS_LTRIG bit (SPI_SSR[4]) defines the slave select signal SPISSx0/1 is level trigger or edge trigger. The selection of trigger condition depends on what type of peripheral slave/master device is connected.

In slave mode, if the SS_LTRIG bit is configured as level trigger, the LTRIG_FLAG bit (SPI_SSR[5]) is used to indicate if both the received number and received bits met the requirement which defines in TX_NUM and TX_BIT_LEN among one transaction done (the transaction done means the slave select has deactivated or the SPI controller has finished one data transfer.)

Level-trigger / Edge-trigger

In slave mode, the slave select signal can be configured as level-trigger or edge-trigger. In edge-trigger, the data transfer starts from an active edge and ends on an inactive edge. If master does not send an inactive edge to slave, the transfer procedure will not be completed and the interrupt flag of slave will not be set. In level-trigger, the following two conditions will terminate the transfer procedure and the interrupt flag of slave will be set. The first condition is that if the number of transferred bits matches the settings of TX_NUM and TX_BIT_LEN, the interrupt flag of slave will be set. The second condition, if master set the slave select pin to inactive level during the transfer is in progress, it will force slave device to terminate the current transfer no matter how many bits have been transferred and the interrupt flag will be set. User can read the status of LTRIG_FLAG bit to check if the data has been completely transferred.

Automatic Slave Select

In master mode, if the bit AUTOSS (SPI_SSR[3]) is set, the slave select signals will be generated automatically and output to SPISSx0 and SPISSx1 pins according to SSR[0] (SPI_SSR[0]) and SSR[1] (SPI_SSR[1]) whether be enabled or not. It means that the slave select signals, which are selected in SSR[1:0], will be asserted by the SPI controller when transmit/receive is started by setting the GO_BUSY bit (SPI_CNTRL[0]) and will be de-asserted after the data transfer is finished. If the AUTOSS bit is cleared, the slave select output signals will be asserted/de-asserted by manual setting/clearing the related bits of SPI_SSR[1:0]. The active state of the slave select output signals is specified in SS_LVL bit (SPI_SSR[2]).

Serial Clock

In master mode, set the DIVIDER1 bits (SPI_DIVIDER[15:0]) to program the output frequency of serial clock to the SPICLK output port. It also supports a variable serial clock if the VARCLK_EN bit (SPI_CTL[23]) is enabled. In this case, the output frequency of serial clock can be programmed as one of the two different frequencies which depend on the value of DIVIDER1 (SPI_DIVIDER[15:0]) and DIVIDER2 (SPI_DIVIDER[31:16]). The serial clock rate of each cycle is depended on the setting of the SPI_VARCLK register.

In slave mode, the off-chip master device drives the serial clock through the SPICLK input port to this SPI controller.

Variable Serial Clock Frequency

In master mode, the output of serial clock can be programmed as variable frequency pattern if the Variable Clock Enable bit VARCLK_EN (SPI_CNTRL[23]) is enabled. The frequency pattern format is defined in VARCLK (SPI_VARCLK[31:0]) register. If the bit content of VARCLK is '0' the output frequency is according with the DIVIDER (SPI_DIVIDER[15:0]) and if the bit content of VARCLK is '1', the output frequency is according to the DIVIDER2 (SPI_DIVIDER[31:16]). Figure 5-53 is the timing relationship among the serial clock (SPICLK), the VARCLK, the DIVIDER and the DIVIDER2 registers. A two-bit combination in the VARCLK defines one clock cycle. The bit field VARCLK[31:30] defines the first clock cycle of SPICLK. The bit field VARCLK[29:28] defines the second clock cycle of SPICLK and so on. The clock source selections are defined in VARCLK and it must be set 1 cycle before the next clock option. For example, if there are 5 CLK1 cycle in SPICLK, the VARCLK shall set 9 '0' in the MSB of VARCLK. The 10th shall be set as '1' in order to switch the next clock source is CLK2. Note that when enable the VARCLK_EN bit, the setting of TX_BIT_LEN must be programmed as 0x10 (16-bit mode only).

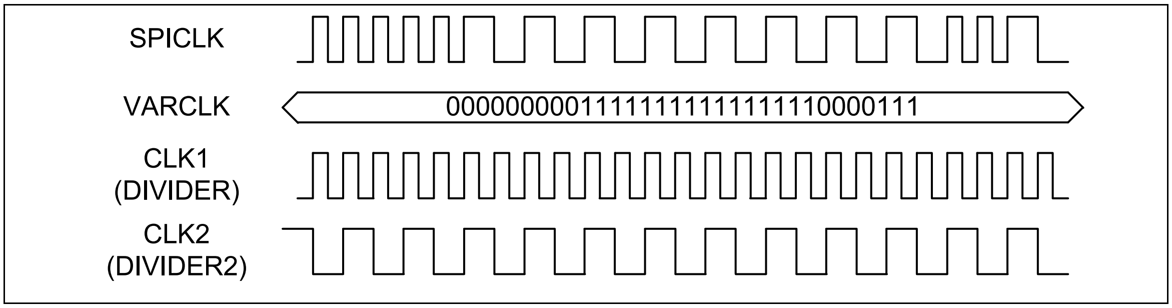


Figure 5-53 Variable Serial Clock Frequency

Clock Polarity

The CLKP bit (SPI_CTL[11]) defines the serial clock idle state. If CLKP = 1, the output SPICLK is idle at high state, otherwise it is at low state if CLKP = 0.

Transmit/Receive Bit Length

The bit length of a transaction word is defined in TX_BIT_LEN bit field (SPI_CNTRL[7:3]). It can be configured up to 32-bit length in a transaction word for transmitting and receiving.

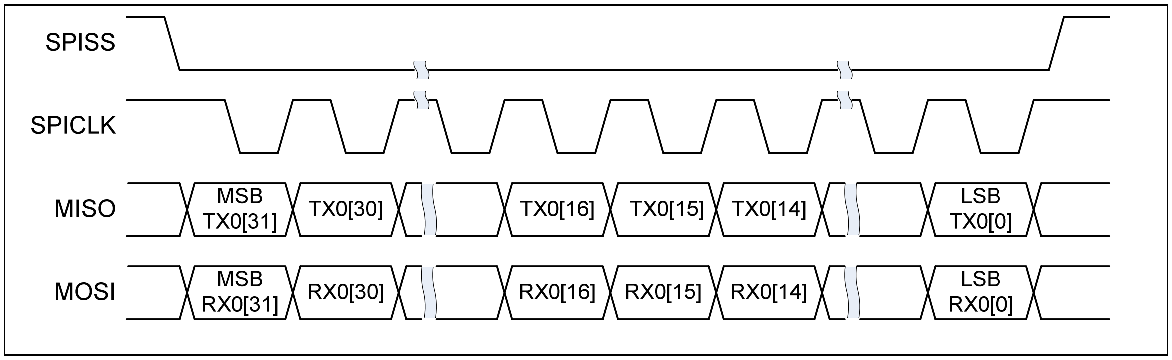


Figure 5-54 32-Bit in one Transaction

Burst Mode

SPI controller can switch to burst mode by setting TX_NUM bit field (SPI_CNTRL[9:8]) to 0x01. In burst mode, SPI can transmit/receive two transactions in one transfer. The SPI burst mode waveform is showed below:

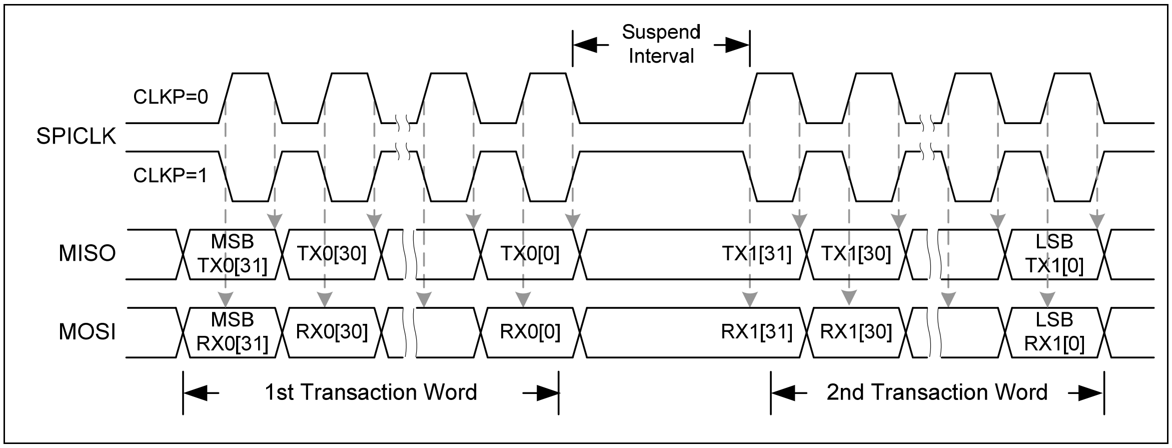


Figure 5-55 Two Transactions in One Transfer (Burst Mode)

LSB First

The LSB bit (SPI_CNTRL[10]) defines the data transmission either from LSB or MSB firstly to start to transmit/receive data.

Transmit Edge

The TX_NEG bit (SPI_CNTRL[2]) defines the data transmitted out either at negative edge or at positive edge of serial clock SPICLK.

Receive Edge

The Rx_NEG bit (SPI_CNTRL[1]) defines the data received in either at negative edge or at positive edge of serial clock SPICLK.

Note: the settings of TX_NEG and RX_NEG are mutual exclusive. In other words, don’t transmit and receive data at the same clock edge.

Word Suspend

These four bits field of SP_CYCLE (SPI_CNTRL[15:12]) provide a configurable suspend interval 2 ~ 17 serial clock periods between two successive transaction words in master mode. The suspend interval is from the last falling clock edge of the preceding transaction word to the first rising clock edge of the following transaction word if CLKP = 0. If CLKP = 1, the interval is from the rising clock edge of the preceding transaction word to the falling clock edge of the following transaction word. The default value of SP_CYCLE is 0x0 (2 serial clock cycles), but set these bits field has no any effects on data transaction process if TX_NUM = 0x00.

Byte Reorder

When the transfer is set as MSB first (LSB = 0) and the REORDER is enabled, the data stored in the TX buffer and RX buffer will be rearranged in the order as [BYTE0, BYTE1, BYTE2, BYTE3] in TX_BIT_LEN = 32-bit mode, and the sequence of transmitted/received data will be BYTE0, BYTE1, BYTE2, and then BYTE3. If the TX_BIT_LEN is set as 24-bit 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 TX_BIT_LEN is configured as 16, 24, and 32 bits.

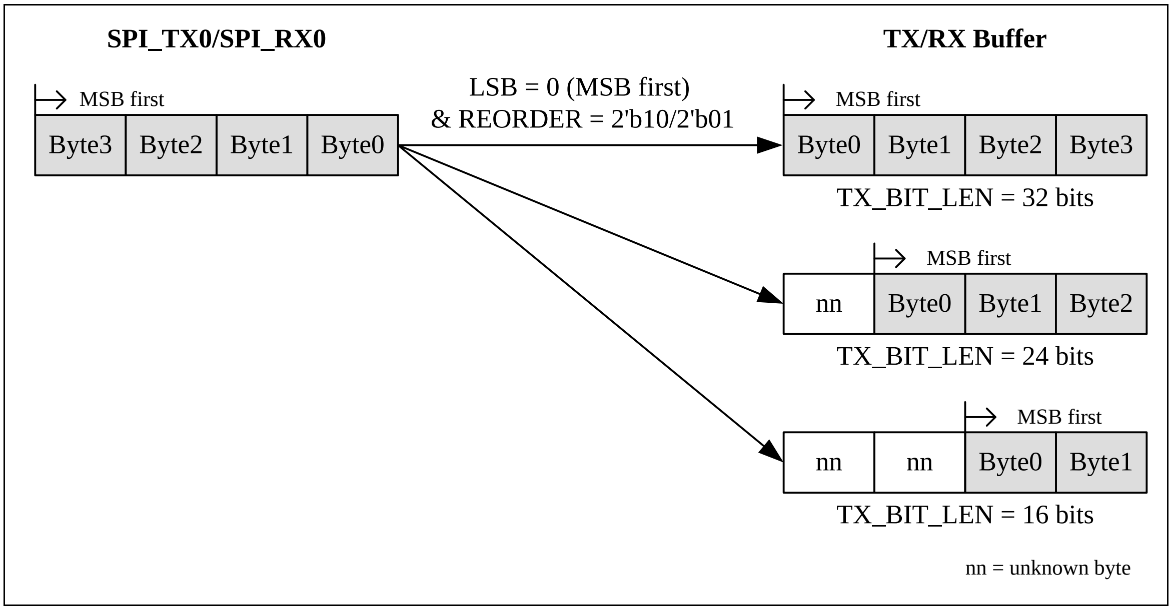


Figure 5-56 Byte Reorder

Byte Suspend

In master mode, if SPI_CNTRL[19] is set to 1, the hardware will insert a suspend interval 2 ~ 17 serial clock periods between two successive bytes in a transaction word. Both settings of byte suspend and word suspend are configured in SP_CYCLE. Note that when enable the byte suspend function, the setting of TX_BIT_LEN must be programmed as 0x00 only (32-bit per transaction word).

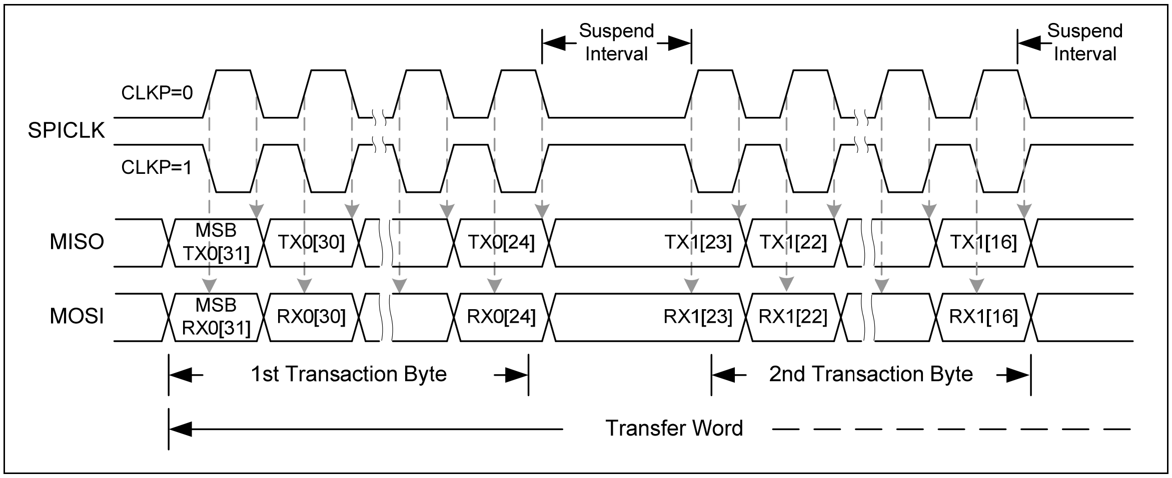


Figure 5-57 Timing Waveform for Byte Suspend

REORDER	Description
00	Disable both byte reorder function and byte suspend interval.
01	Enable byte reorder function and insert a byte suspend internal (2~17 SPICLK) among each byte. The setting of TX_BIT_LEN must be configured as 0x00 (32 bits/ word)
10	Enable byte reorder function but disable byte suspend function
11	Disable byte reorder function, but insert a suspend interval (2~17 SPICLK) among each byte. The setting of TX_BIT_LEN must be configured as 0x00 (32 bits/ word)

Table 5-6 Byte Order and Byte Suspend Conditions

No Slave Select Mode (3-WIRE Mode)

This is used to ignore the slave select signal in slave mode. The SPI controller can work on no slave select mode (3-WIRE mode) interface including SPICLK, SPI_MISO, and SPI_MOSI when it is set as a slave device. When the NOSLVSEL bit is set as 1, the controller will start to transmit/receive data after the GO_BUSY bit is set to 1 and the serial clock appears. In no slave select signal mode, the SS_LTRIG, SPI_SSR[4], shall be set as 1.

Interrupt

Each SPI controller can generates an individual interrupt when data transfer is finished and the respective interrupt event flag IF (SPI_CNTRL[16]) will be set. The interrupt event flag will generates an interrupt to CPU if the interrupt enable bit IE (SPI_CNTRL[17]) is set. The interrupt event flag **IF** can be cleared only by writing 1 to it.

In 3-WIRE mode, the interrupt flag in SLV_START_INTSTS will be set when the transfer has start and there is also interrupt event when the received data meet the required bits which define in TX_BIT_LEN and TX_NUM. If the received bits are less than the requirement and there is no more serial clock input over the time period which is defined by the user in slave mode with no slave select, the user can set the SLV_ABORT bit to force the current transfer done and then the user can get a transfer done interrupt event.

Two Bit Transfer Mode

This SPI controller also supports two-bit transfer mode when set the TWOB bit (SPI_CNTRL[22]) to 1. When the TWOB bit is enabled, it can transmit and receives two-bit serial data simultaneously.

For example, in master mode, the data stored at SPI_TX0 register and SPI_TX1 register will be transmitted through the MOSIx0 pin and MOSIx1 pin respectively. In the meanwhile, the SPI_RX0 register and SPI_RX1 register will store the data received from MISOx0 pin and MISOx1 pin respectively.

In slave mode, the data stored at SPI_TX0 register and SPI_TX1 register will be transmitted through the MISOx0 pin and MISOx1 pin respectively. In the meanwhile, the SPI_RX0 register and SPI_RX1 register will store the data received from MOSIx0 pin and MOSIx1 pin respectively.

Note that when enable the TWOB bit, the setting of TX_NUM must be programmed as 0x00 only.

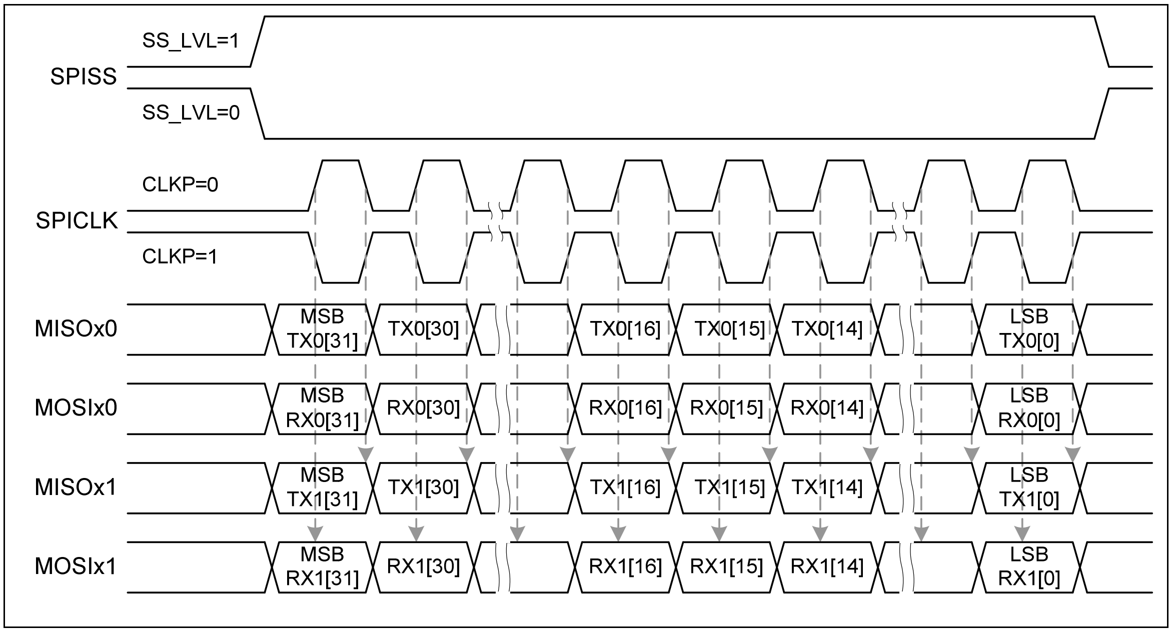


Figure 5-58 Two Bits Transfer Mode (slave mode)

FIFO Mode

The SPI controller supports FIFO mode when the FIFO in SPI_CNTRL[21] is set as 1. If the FIFO mode is disabled, the user can only update the SPI_TX0 buffer after the current transfer is done. In FIFO mode, the next one transmitted data can be written into the SPI_TX0 buffer in advance. When the SPI controller operates with FIFO mode, the GO_BUSY bit of SPI_CNTRL register will be controlled by hardware. It may result in wrong transfer If users modify the GO_BUSY bit.

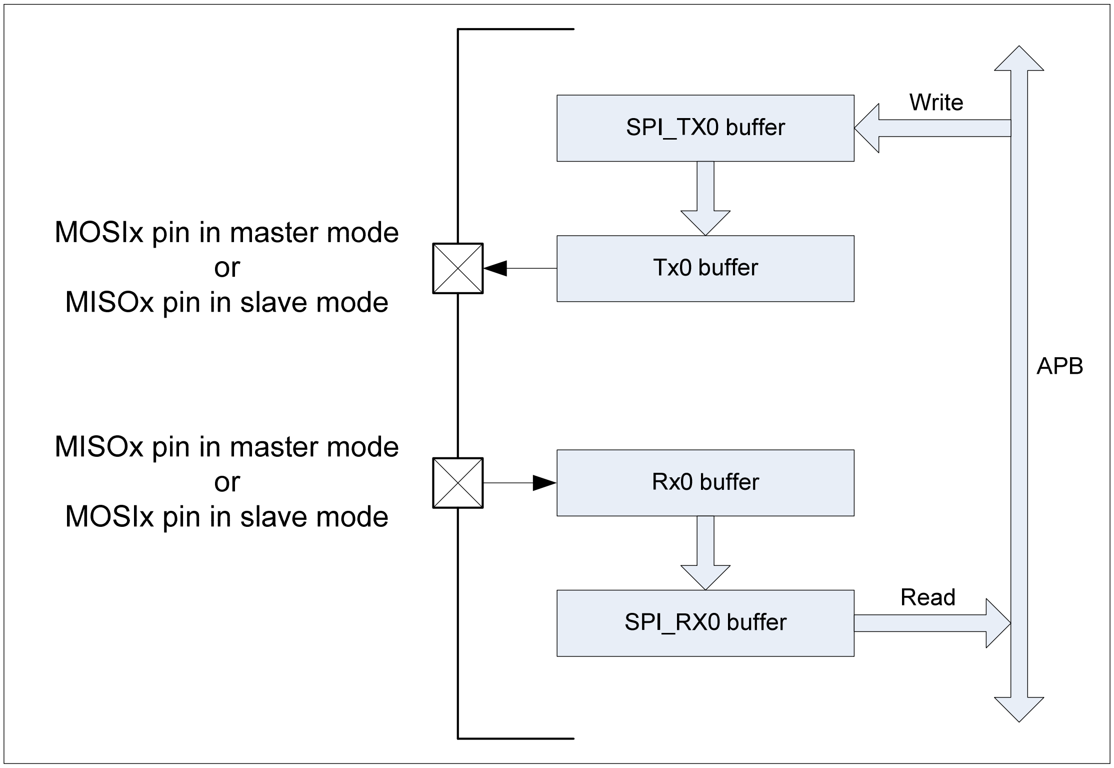


Figure 5-59 FIFO mode Block Diagram

In master mode transmission operation, before the FIFO bit is set, users can write 1st data into SPI_TX0 buffer. The TX_EMPTY flag will be cleared to 0. When the FIFO bit is set to 1, the data stored at SPI_TX0 buffer will be loaded into the Tx0 buffer and the transmission starts immediately. Users can write the 2nd data into SPI_TX0 buffer again and it will be loaded into the transmitted bus buffer after the 1st transfer done. The SPI controller will insert a suspend interval between two successive transactions in FIFO mode and the period of suspend interval is decided by the setting of SP_CYCLE (SPI_CNTRL [15:12]). Users can write the transmitted data into SPI_TX0 buffer whenever the TX_FULL flag is 0.

The transfer will be triggered automatically when the next transmitted data is updated in time. If the SPI_TX0 buffer isn’t updated after all data transfer is done, the transfer will stop.

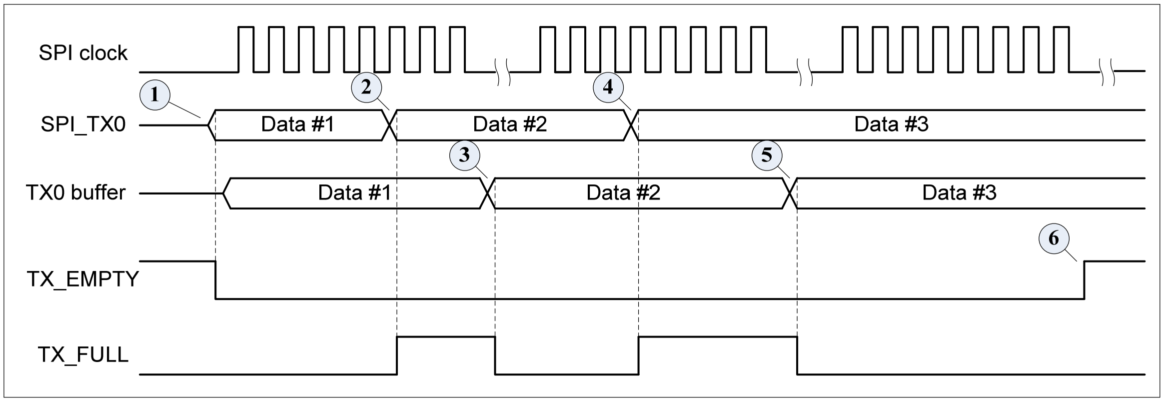
In master mode reception operation, the serial data is received from MISOx pin and stored at Rx0 buffer. When a data transfer is done and the RX_EMPTY flag is 1, the received data stored at Rx0 buffer will be loaded into SPI_RX0 buffer and the RX_EMPTY flag will be cleared to 0. Otherwise, when a data transfer is done and the RX_EMPTY flag is 0, the TX_FULL flag will be set to 1, and the data stored at SPI_RX0 buffer will not be replaced by Rx0 buffer until software read the SPI_RX0 buffer. Users can read the received data from SPI_RX0 buffer whenever the RX_EMPTY flag is 0.

In slave mode, when the FIFO bit is set as 1, the GO_BUSY bit will be set as 1 by hardware automatically. If users want to stop the slave mode SPI data transfer, both the FIFO bit and GO_BUSY bit must be cleared to 0 by software.

In slave mode transmission operation, when software writes data to SPI_TX0 buffer, the data will be loaded into Tx0 buffer if TX_EMPTY flag is 1. After a write operation, the TX_EMPTY flag will be cleared to 0. The transmission will start when the slave device receives clock signal from master. Users can write data to SPI_TX0 buffer whenever TX_FULL flag is 0. As a transfer is done and the IE bit of SPI_CNTRL register is set to 1, SPI interrupt will be generated. The data stored at SPI_TX0 buffer will be loaded to Tx0 buffer automatically if software has updated the SPI_TX0 buffer. After all data have been transferred and software doesn't update the SPI_TX0 buffer, the TX_EMPTY flag will be set to 1.

In slave mode reception operation, the serial data is received from MOSIx pin and stored at Rx0 buffer. The reception mechanism is similar to master mode.

The following is an example of FIFO mode operations. The SPI controller is configured as a master with 8-bit data length in each transfer.



1) Software writes the first data to SPI_TX0 register. The first data will be loaded to TX0 buffer if the TX_EMPTY flag is 1. And the TX_EMPTY flag will be cleared to 0 by hardware. The first data transfer sequence starts.

2) As long as the TX_FULL flag is 0, software can write the next data to SPI register in advance. In this example, software writes the second data to SPI_TX0 register when the first data transfer sequence is in progress. The TX_FULL flag will be set to 1 by hardware.

3) As the first data transfer is finished and the TX_EMPTY flag is not 1, the data stored in SPI_TX0 will be loaded to TX0 buffer and the TX_FULL flag will be cleared to 0 by hardware. The second data transfer sequence will start automatically after a user-specified suspending interval.

4) Software writes the 3rd data to SPI_TX0 register. The TX_FULL flag will be set to 1 again.

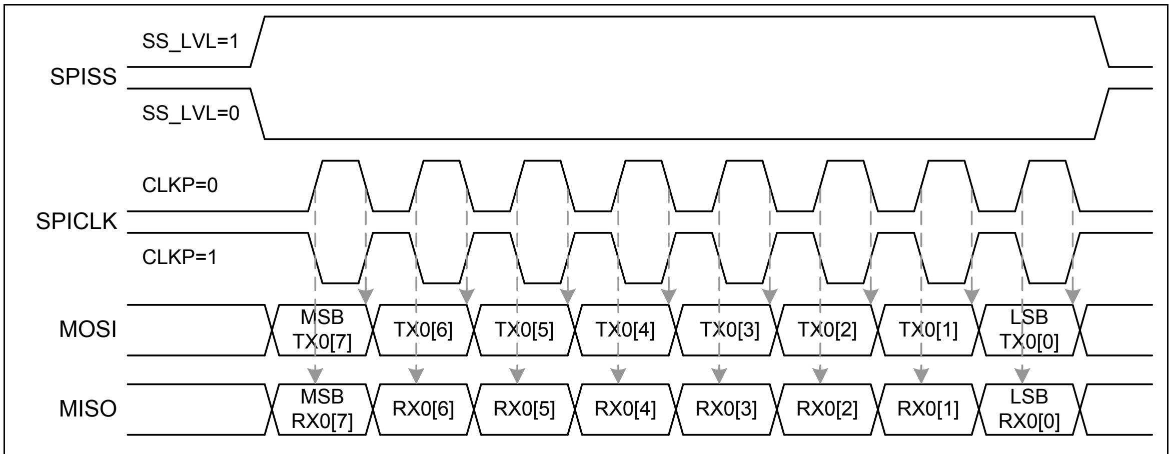
5) The second data transfer sequence is finished. The data stored in SPI_TX0 will be loaded to TX0 buffer and the TX_FULL flag will be cleared to 0 by hardware. The 3rd data transfer sequence will

start automatically after a user-specified suspending interval.

If software does not update the SPI_TX0 register, the TX_EMPTY flag will be set to 1 after the 3rd data transfer sequence is finished

5.9.5 Timing Diagram

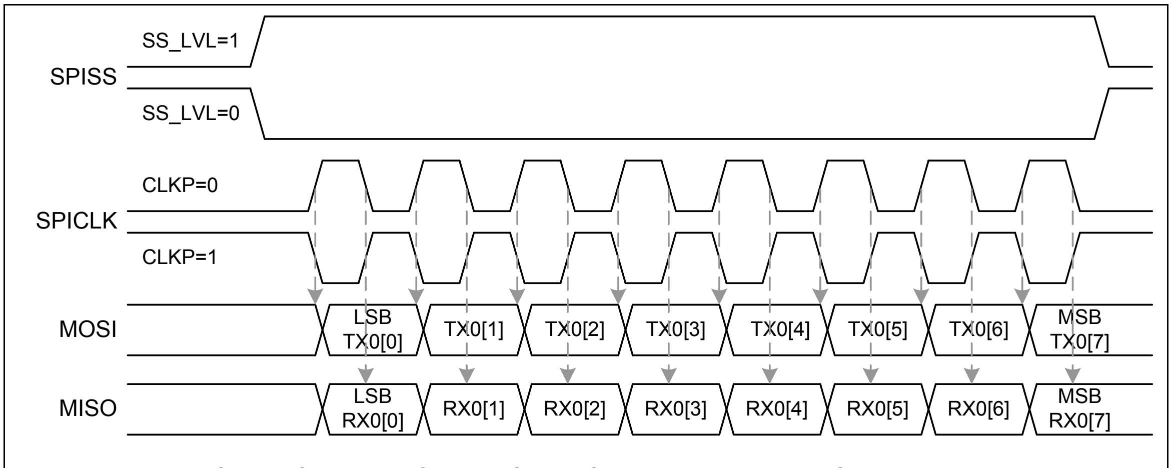
The active state of slave select signal can be defined by the settings of SS_LVL bit (SPI_SSR[2]) and SS_LTRIG bit (SPI_SSR[4]). The serial clock (SPICLK) idle state can be configured as high state or low state by setting the CLKP bit (SPI_CNTRL[11]). It also provides the bit length of a transaction word in TX_BIT_LEN (SPI_CNTRL[7:3]), the transfer number in TX_NUM (SPI_CNTRL[8]), and transmit/receive data from MSB or LSB first in LSB bit (SPI_CNTRL[10]). Users also can select which edge of serial clock to transmit/receive data in TX_NEG/RX_NEG (SPI_CNTRL[2:1]) registers. Four SPI timing diagrams for master/slave operations and the related settings are shown as below.



Master Mode: CNTRL[SLVAE]=0, CNTRL[LSB]=0, CNTRL[TX_NUM]=0x0, CNTRL[TX_BIT_LEN]=0x08

1. CNTRL[CLKP]=0, CNTRL[TX_NEG]=1, CNTRL[RX_NEG]=0 or
2. CNTRL[CLKP]=1, CNTRL[TX_NEG]=0, CNTRL[RX_NEG]=1

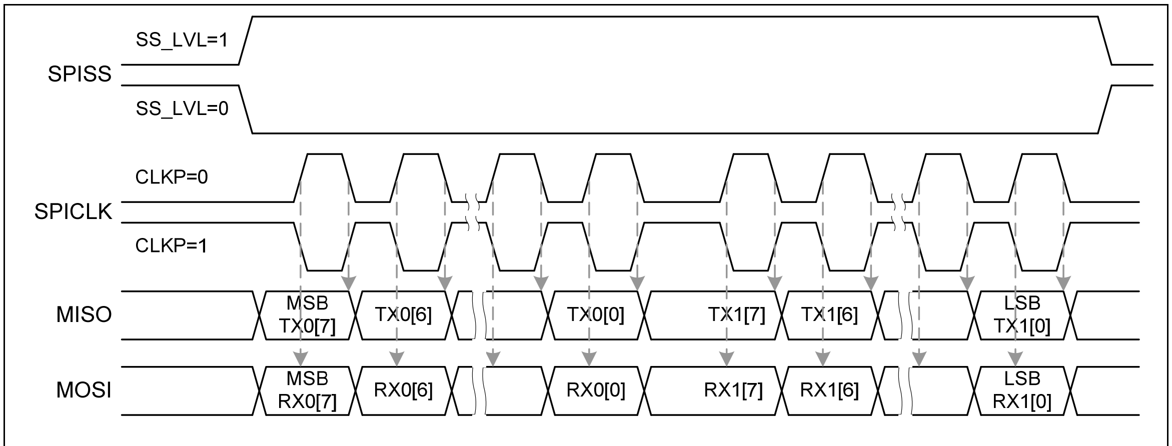
Figure 5-60 SPI Timing in Master Mode



Master Mode: CNTRL[SLVAE]=0, CNTRL[LSB]=1, CNTRL[TX_NUM]=0x0, CNTRL[TX_BIT_LEN]=0x08

1. CNTRL[CLKP]=0, CNTRL[TX_NEG]=0, CNTRL[RX_NEG]=1 or
2. CNTRL[CLKP]=1, CNTRL[TX_NEG]=1, CNTRL[RX_NEG]=0

Figure 5-61 SPI Timing in Master Mode (Alternate Phase of SPICLK)



Slave Mode: CNTRL[SLVAE]=1, CNTRL[LSB]=0, CNTRL[TX_NUM]=0x01, CNTRL[TX_BIT_LEN]=0x08

1. CNTRL[CLKP]=0, CNTRL[TX_NEG]=1, CNTRL[RX_NEG]=0 or
2. CNTRL[CLKP]=1, CNTRL[TX_NEG]=0, CNTRL[RX_NEG]=1

Figure 5-62 SPI Timing in Slave Mode

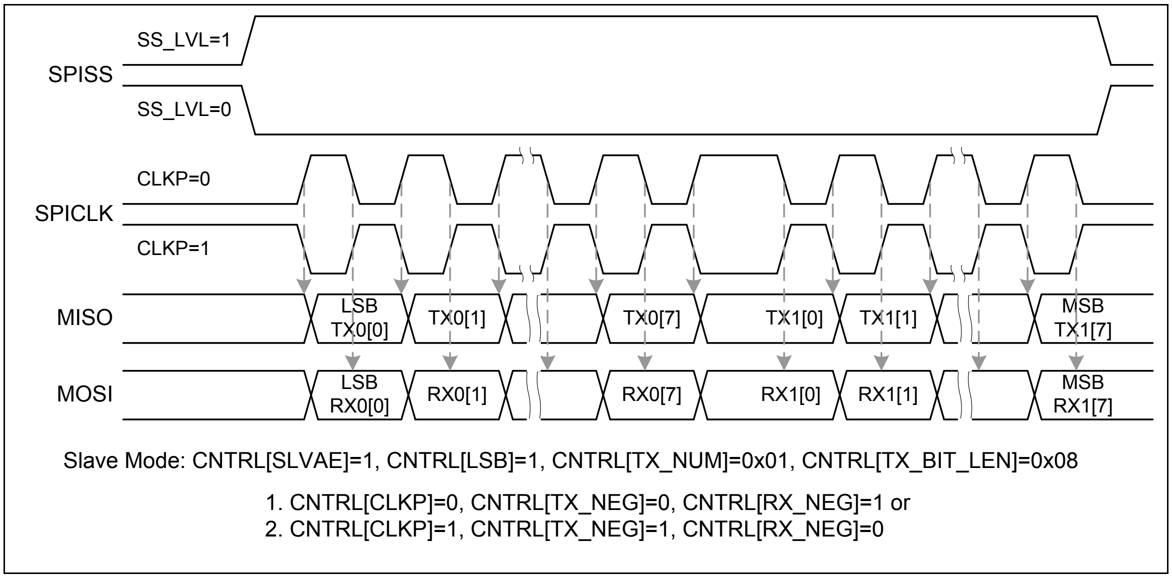


Figure 5-63 SPI Timing in Slave Mode (Alternate Phase of SPICLK)

5.9.6 Programming Examples

Example 1, SPI controller is set as a master to access an off-chip slave device with following specifications:

- Data bit is latched on positive edge of serial clock
- Data bit is driven on negative edge of serial clock
- Data is transferred from MSB first
- SPICLK is idle at low state
- Only one byte of data to be transmitted/received in a transaction
- Use the first SPI slave select pin to connect with an off-chip slave device. Slave select signal is active low

The operation flow is as follows.

1) Set the DIVIDER (SPI_DIVIDER [15:0]) register to determine the output frequency of serial clock.

2) Write the SPI_SSR register a proper value for the related settings of master mode

   1. Disable the Automatic Slave Select bit AUTOSS(SPI_SSR[3] = 0)

      Select low level trigger output of slave select signal in the Slave Select Active Level bit SS_LVL (SPI_SSR[2] = 0)

   2. Select slave select signal to be output active at the IO pin by setting the Slave Select Register bits SSR[0] (SPI_SSR[0]) to active the off-chip slave devices

3) Write the related settings into the SPI_CNTRL register to control this SPI master actions

   1. Set this SPI controller as master device in SLAVE bit (SPI_CNTRL[18] = 0)

2. Force the serial clock idle state at low in CLKP bit (SPI_CNTRL[11] = 0)
3. Select data transmitted at negative edge of serial clock in TX_NEG bit (SPI_CNTRL[2] = 1)
4. Select data latched at positive edge of serial clock in RX_NEG bit (SPI_CNTRL[1] = 0)
5. Set the bit length of word transfer as 8-bit in TX_BIT_LEN bit field (SPI_CNTRL[7:3] = 0x08)
6. Set only one time of word transfer in TX_NUM (SPI_CNTRL[9:8] = 0x0)
7. Set MSB transfer first in MSB bit (SPI_CNTRL[10] = 0), and don't care the SP_CYCLE bit field (SPI_CNTRL[15:12]) due to it's not in burst mode in this case

4) If this SPI master will transmits (writes) one byte data to the off-chip slave device, write the byte data that will be transmitted into the TX0[7:0] (SPI_TX0[7:0]) register.

5) If this SPI master just only receives (reads) one byte data from the off-chip slave device, you don't need to care what data will be transmitted and just write 0xFF into the SPI_TX0[7:0] register.

6) Enable the GO_BUSY bit (SPI_CNTRL [0] = 1) to start the data transfer at the SPI interface.

7) Waiting for SPI interrupt occurred (if the Interrupt Enable IE bit is set) or just polling the GO_BUSY bit till it is cleared to 0 by hardware automatically.

8) Read out the received one byte data from RX0 [7:0] (SPI_RX0[7:0]) register.

9) Go to 4) to continue another data transfer or set SSR [0] to 0 to inactivate the off-chip slave devices.

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 serial clock
- Data bit is driven on negative edge of serial clock
- Data is transferred from LSB first
- SPICLK is idle at high state
- Only one byte of data to be transmitted/received in a transaction
- Slave select signal is high level trigger

The operation flow is as follows.

1) Write the SPI_SSR register a proper value for the related settings of slave mode

Select high level and level trigger for the input of slave select signal by setting the Slave Select Active Level bit SS_LVL (SPI_SSR[2] = 1) and the Slave Select Level Trigger bit SS_LTRIG (SPI_SSR[4] = 1).

2) Write the related settings into the SPI_CNTRL register to control this SPI slave actions

1. Set this SPI controller as slave device in SLAVE bit (SPI_CNTRL[18] = 1)
2. Select the serial clock idle state at high in CLKP bit (SPI_CNTRL[11] = 1)
3. Select data transmitted at negative edge of serial clock in TX_NEG bit (SPI_CNTRL[2] = 1)

4. Select data latched at positive edge of serial clock in RX_NEG bit (SPI_CNTRL[1] = 0)
5. Set the bit length of word transfer as 8-bit in TX_BIT_LEN bit field (SPI_CNTRL[7:3] = 0x08)
6. Set only one time of word transfer in TX_NUM (SPI_CNTRL[9:8] = 0x0)
7. Set LSB transfer first in LSB bit (SPI_CNTRL[10] = 1), and don't care the SP_CYCLE bit field (SPI_CNTRL[15:12]) due to not burst mode in this case.

3) If this SPI slave will transmits (be read) one byte data to the off-chip master device, write the byte data that will be transmitted into the TX0 [7:0] (SPI_TX0[7:0]) register.

4) If this SPI slave just only receives (be written) one byte data from the off-chip master device, you don't care what data will be transmitted and just write 0xFF into the SPI_TX0[7:0] register.

5) Enable the GO_BUSY bit (SPI_CNTRL[0] = 1) to wait for the slave select trigger input and serial clock input from the off-chip master device to start the data transfer at the SPI interface.

6) Waiting for SPI interrupt occurred (if the Interrupt Enable IE bit is set), or just polling the GO_BUSY bit till it is cleared to 0 by hardware automatically.

7) Read out the received one byte data from RX[7:0] (SPI_RX0[7:0]) register.

Go to 3) to continue another data transfer or disable the GO_BUSY bit to stop data transfer.

5.9.7 Register Map

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

Register	Offset	R/W	Description	Reset Value
SPI0_BA = 0x4003_0000 **SPI1_BA = 0x4003_4000** **SPI2_BA = 0x4013_0000** **SPI3_BA = 0x4013_4000**				
SPI_CNTRL	SPIx_BA+0x00	R/W	Control and Status Register	0x0500_0004
SPI_DIVIDER	SPIx_BA+0x04	R/W	Clock Divider Register	0x0000_0000
SPI_SSR	SPIx_BA+0x08	R/W	Slave Select Register	0x0000_0000
SPI_RX0	SPIx_BA+0x10	R	Data Receive Register 0	0x0000_0000
SPI_RX1	SPIx_BA+0x14	R	Data Receive Register 1	0x0000_0000
SPI_TX0	SPIx_BA+0x20	W	Data Transmit Register 0	0x0000_0000
SPI_TX1	SPIx_BA+0x24	W	Data Transmit Register 1	0x0000_0000
SPI_VARCLK	SPIx_BA+0x34	R/W	Variable Clock Pattern Register	0x007F_FF87
SPI_DMA	SPIx_BA+0x38	R/W	SPI DMA Control Register	0x0000_0000
SPI_CNTRL2	SPIx_BA+0x3C	R/W	Control and Status Register 2	0x0000_0000
SPI_FIFO_CTL	SPIx_BA+0x40	R/W	FIFO Control Register	0x0000_0000

Note: When software programs CNTRL, the GO_BUSY bit should be written last.

5.9.8 Register Description

SPI Control and Status Register (SPI_CNTRL)

Register	Offset	R/W	Description	Reset Value
SPI_CNTRL	SPIx_BA+0x00	R/W	Control and Status Register	0x0500_0004

31	30	29	28	27	26	25	24
Reserved				TX_FULL	TX_EMPTY	RX_FULL	RX_EMPTY
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
VARCLK_EN	TWOB	FIFO	REORDER		SLAVE	IE	IF
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SP_CYCLE				CLKP	LSB	TX_NUM	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TX_BIT_LEN					TX_NEG	RX_NEG	GO_BUSY

Bits	Descriptions	
[31:28]	**Reserved**	Reserved
[27]	**TX_FULL**	**Transmitted FIFO_FULL STATUS** (read only) 1 = Indicates that the transmitted data buffer is full on the FIFO 0 = Indicates that the transmitted data buffer is not full
[26]	**TX_EMPTY**	**Transmitted FIFO_EMPTY STATUS** (read only) 1 = Indicates that the transmitted data buffer is empty 0 = Indicates that the transmitted data buffer is not empty
[25]	**RX_FULL**	**Received FIFO_FULL STATUS** (read only) 1 = Indicates that the receive data buffer is full on the FIFO 0 = Indicates that the receive data buffer is not full
[24]	**RX_EMPTY**	**Received FIFO_EMPTY STATUS** (read only) 1 = Indicates that the receive data buffer is empty 0 = Indicates that the receive data buffer is not empty
[23]	**VARCLK_EN**	**Variable Clock Enable** (Master Only) 1 = The serial clock output frequency is variable. The output frequency is decided by the value of VARCLK, DIVIDER, and DIVIDER2. 0 = The serial clock output frequency is fixed and decided only by the value of DIVIDER. Note that when enable this VARCLK_EN bit, the setting of TX_BIT_LEN must be programmed as 0x10 (16-bit mode)
[22]	**TWOB**	**Two Bits Transfer Mode Active**

		1 = Enable two-bit transfer mode. 0 = Disable two-bit transfer mode. Note that when enable TWOB, the serial transmitted 2-bit data output are from SPI_TX1/0, and the received 2-bit data input are put in SPI_RX1/0. Note that when enable TWOB, the setting of TX_NUM must be programmed as 0x00
[21]	**FIFO**	**FIFO Mode** 1 = Enable FIFO Mode 0 = Disable FIFO Mode Note: 1. Before enabling FIFO mode, the other related settings should be set in advance. 2. In slave mode with level-trigger configuration, the slave select pin must be kept at active state during the successive data transfer. 3. In FIFO mode, both the REORDER field and the TX_NUM field must be configured as 0. In other words, the byte-reorder function, byte suspend function and burst mode must be disable.
[20:19]	**REORDER**	**Reorder Mode Select** 00 = Disable both byte reorder and byte suspend functions. 01 = Enable byte reorder function and insert a byte suspend interval (2~17 SPICLK cycles) among each byte. The setting of TX_BIT_LEN must be configured as 0x00. (32 bits/word) 10 = Enable byte reorder function, but disable byte suspend function. 11 = Disable byte reorder function, but insert a suspend interval (2~17 SPICLK cycles) among each byte. The setting of TX_BIT_LEN must be configured as 0x00. (32 bits/word) Note: 1. Byte reorder function is only available if TX_BIT_LEN is defined as 16, 24, and 32 bits. 2. In slave mode with level-trigger configuration, if the byte suspend function is enabled, the slave select pin must be kept at active state during the successive four bytes transfer.
[18]	**SLAVE**	**Slave Mode Indication** 1 = Slave mode 0 = Master mode
[17]	**IE**	**Interrupt Enable** 1 = Enable SPI Interrupt 0 = Disable SPI Interrupt
[16]	**IF**	**Interrupt Flag** 1 = It indicates that the transfer is done. 0 = It indicates that the transfer dose not finish yet. Note: This bit will be cleared by writing 1 to itself.
[15:12]	**SP_CYCLE**	**Suspend Interval** (Master Only) These four bits provide configurable suspend interval between two successive transmit/receive transaction in a transfer. The suspend interval is from the last falling clock edge of the current transaction to the first rising clock edge of the successive transaction if CLKP = 0. If CLKP = 1, the interval is from the rising clock edge to the falling clock edge. The default value is 0x0. When TX_NUM = 00b, setting this field has no effect on transfer. The desired suspend interval is obtained according to the following equation:

		■ For byte suspend interval and burst mode suspend interval: (SP_CYCLE[3:0] + 2) * period of SPICLK + 1 system clock cycle Ex: SP_CYCLE = 0x0 … 2 SPICLK clock cycle + 1 system clock cycle SP_CYCLE = 0x1 … 3 SPICLK clock cycle + 1 system clock cycle …… SP_CYCLE = 0xE … 16 SPICLK clock cycle + 1 system clock cycle SP_CYCLE = 0xF … 17 SPICLK clock cycle + 1 system clock cycle If the SPI clock rate equals system clock rate, that is to say, the DIV_ONE feature is enabled, the burst mode suspend interval period is (SP_CYCLE[3:0] * 2 + 3.5) * period of system clock ■ For FIFO mode suspend interval: (SP_CYCLE can't be set as 1) In case of disabling the DIV_ONE feature, if SP_CYCLE = 2~15, suspend interval period is (SP_CYCLE[3:0] + 3) * system clock cycle + 1 SPICLK clock cycle Ex: SP_CYCLE = 0x2 … 5 system clock cycles + 1 SPICLK clock cycle …… SP_CYCLE = 0xE … 17 system clock cycles + 1 SPICLK clock cycle SP_CYCLE = 0xF … 18 system clock cycles + 1 SPICLK clock cycle if SP_CYCLE = 0, suspend interval period is 35 * system clock cycle + 1 SPICLK clock cycle In case of enabling the DIV_ONE feature, if SP_CYCLE = 2~15, suspend interval period is (SP_CYCLE[3:0] + 3.5) * system clock period Ex: SP_CYCLE = 0x2 … 5.5 system clock periods …… SP_CYCLE = 0xE … 17.5 system clock periods SP_CYCLE = 0xF … 18.5 system clock periods if SP_CYCLE = 0, suspend interval period is 35.5 * system clock period
[11]	**CLKP**	**Clock Polarity** 1 = SPICLK idle high 0 = SPICLK idle low
[10]	**LSB**	**LSB First** 1 = The LSB is sent first on the line (bit 0 of SPI_TX0/1), and the first bit received from

		the line will be put in the LSB position in the RX register (bit 0 of SPI_RX0/1). 0 = The MSB is transmitted/received first (which bit in SPI_TX0/1 and SPI_RX0/1 register that is depends on the TX_BIT_LEN field).
[9:8]	**TX_NUM**	**Numbers of Transmit/Receive Word** This field specifies how many transmit/receive word numbers should be executed in one transfer. 00 = Only one transmit/receive word will be executed in one transfer. 01 = Two successive transmit/receive words will be executed in one transfer. (burst mode) 10 = Reserved. 11 = Reserved. Note: in slave mode with level-trigger configuration, if TX_NUM is set to 01, the slave select pin must be kept at active state during the successive data transfer.
[7:3]	**TX_BIT_LEN**	**Transmit Bit Length** This field specifies how many bits are transmitted in one transaction. Up to 32 bits can be transmitted. TX_BIT_LEN = 0x01 … 1 bit TX_BIT_LEN = 0x02 … 2 bits …… TX_BIT_LEN = 0x1F … 31 bits TX_BIT_LEN = 0x00 … 32 bits
[2]	**TX_NEG**	**Transmit At Negative Edge** 1 = The transmitted data output signal is changed at the falling edge of SPICLK 0 = The transmitted data output signal is changed at the rising edge of SPICLK
[1]	**RX_NEG**	**Receive At Negative Edge** 1 = The received data input signal is latched at the falling edge of SPICLK 0 = The received data input signal is latched at the rising edge of SPICLK
[0]	**GO_BUSY**	**Go and Busy Status** 1 = In master mode, writing 1 to this bit to start the SPI data transfer; in slave mode, writing 1 to this bit indicates that the slave is ready to communicate with a master. 0 = Writing 0 to this bit to stop data transfer if SPI is transferring. During the data transfer, this bit keeps the value of 1. As the transfer is finished, this bit will be cleared automatically. Note: 1. All registers should be set before writing 1 to this GO_BUSY bit. 2. In FIFO mode, this bit will be controlled by hardware. Software should not modify this bit.

SPI Divider Register (SPI_DIVIDER)

Register	Offset	R/W	Description	Reset Value
SPI_DIVIDER	SPIx_BA+0x04	R/W	Clock Divider Register (Master Only)	0x0000_0000

31	30	29	28	27	26	25	24
DIVIDER2[15:8]							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
DIVIDER2[7:0]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
DIVIDER[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DIVIDER[7:0]							

Bits	Descriptions	
[31:16]	**DIVIDER2**	**Clock Divider 2 Register** (master only) The value in this field is the 2nd frequency divider for generating the serial clock on the output SPICLK. The desired frequency is obtained according to the following equation: $f_{sclk} = \frac{f_{pclk}}{(DIVIDER2+1)*2}$ If VARCLK_EN is cleared to 0, this setting is unmeaning.
[15:0]	**DIVIDER**	**Clock Divider Register** (master only) The value in this field is the frequency divider for generating the serial clock on the output SPICLK. The desired frequency is obtained according to the following equation: $f_{sclk} = \frac{f_{pclk}}{(DIVIDER+1)*2}$ In slave mode, the period of SPI clock driven by a master shall equal or over 5 times the period of PCLK. In other words, the maximum frequency of SPI clock is the fifth of the frequency of slave's PCLK.

SPI Slave Select Register (SPI_SSR)

Register	Offset	R/W	Description	Reset Value
SPI_SSR	SPI0_BA+0x08	R/W	Slave 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		LTRIG_FLAG	SS_LTRIG	AUTOSS	SS_LVL	SSR	

Bits	Descriptions	
[31:6]	**Reserved**	Reserved
[5]	**LTRIG_FLAG**	**Level Trigger Flag** When the SS_LTRIG bit is set in slave mode, this bit can be read to indicate the received bit number is met the requirement or not. 1 = The transaction number and the transferred bit length met the specified requirements which defined in TX_NUM and TX_BIT_LEN. 0 = The transaction number or the transferred bit length of one transaction doesn't meet the specified requirements. Note: This bit is READ only
[4]	**SS_LTRIG**	**Slave Select Level Trigger (Slave only)** 1 = The slave select signal will be level-trigger. It depends on SS_LVL to decide the signal is active low or active high. 0 = The input slave select signal is edge-trigger. This is the default value. It depends on SS_LVL to decide the signal is active at falling-edge or rising-edge
[3]	**AUTOSS**	**Automatic Slave Select (Master only)** 1 = If this bit is set, SPISSx0/1 signals will be generated automatically. It means that device/slave select signal, which is set in SSR[1:0], will be asserted by the SPI controller when transmit/receive is started by setting GO_BUSY, and will be de-asserted after each transmit/receive is finished. 0 = If this bit is cleared, slave select signals will be asserted/de-asserted by setting /clearing related bits in SSR[1:0].
[2]	**SS_LVL**	**Slave Select Active Level** It defines the active status of slave select signal (SPISSx0/1). 1 = The slave select signal SPISSx0/1 is active at high-level/rising-edge. 0 = The slave select signal SPISSx0/1 is active at low-level/falling-edge.
[1:0]	**SSR**	**Slave Select Register (Master only)**

		If AUTOSS bit is cleared, writing 1 to any bit location of this field sets the proper SPISSx0/1 line to an active state and writing 0 sets the line back to inactive state. If AUTOSS bit is set, writing 0 to any bit location of this field will keep the corresponding SPISSx0/1 line at inactive state; writing 1 to any bit location of this field will select the corresponding SPISSx0/1 line to be automatically driven to active state for the duration of the transmit/receive, and will be driven to inactive state for the rest of the time. The active state of SPISSx0/1 is specified in SS_LVL. Note: SPISSx0 is also defined as slave select input in slave mode.

SPI Data Receive Register (SPI_RX)

Register	Offset	R/W	Description	Reset Value
SPI_RX0	SPIx_BA+0x10	R	Data Receive Register 0	0x0000_0000
SPI_RX1	SPIx_BA+0x14	R	Data Receive Register 1	0x0000_0000

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

Bits	Descriptions	
[31:0]	**RX**	**Data Receive Register** The Data Receive Registers hold the value of received data of the last executed transfer. Valid bits depend on the transmit bit length field in the SPI_CNTRL register. For example, if TX_BIT_LEN is set to 0x08 and TX_NUM is set to 0x0, bit RX0[7:0] holds the received data. The values of the other bits are unknown. Note: The Data Receive Registers are read only registers.

SPI Data Transmit Register (SPI_TX)

Register	Offset	R/W	Description	Reset Value
SPI_TX0	SPIx_BA+0x20	W	Data Transmit Register 0	0x0000_0000
SPI_TX1	SPIx_BA+0x24	W	Data Transmit Register 1	0x0000_0000

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

Bits	Descriptions	
[31:0]	**TX**	**Data Transmit Register** The Data Transmit Registers hold the data to be transmitted in the next transfer. Valid bits depend on the transmit bit length field in the CNTRL register. For example, if TX_BIT_LEN is set to 0x08 and the TX_NUM is set to 0x0, the bit TX0[7:0] will be transmitted in next transfer. If TX_BIT_LEN is set to 0x00 and TX_NUM is set to 0x1, the SPI controller will perform two 32-bit transmit/receive successive using the same setting. The transmission sequence is TX0[31:0] first and then TX1[31:0].

SPI Variable Clock Pattern Register (SPI_VARCLK)

Register	Offset	R/W	Description	Reset Value
SPI_VARCLK	SPIx_BA+0x34	R/W	Variable Clock Pattern Register	0x007F_FF87

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

Bits	Descriptions	
[31:0]	**VARCLK**	**Variable Clock Pattern** The value in this field is the frequency patterns of the SPI clock. If the bit pattern of VARCLK is ‘0’, the output frequency of SPICLK is according the value of DIVIDER. If the bit patterns of VARCLK are ‘1’, the output frequency of SPICLK is according the value of DIVIDER2. Refer to register SPI_DIVIDER. Refer to Variable Serial Clock Frequency paragraph for more detail description.

DMA Control Register (DMACTL)

Register	Offset	R/W	Description	Reset Value
SPI_DMA	SPIx_BA+0x38	R/W	SPI DMA 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**
Reserved						**RX_DMA_GO**	**TX_DMA_GO**

Bits	Descriptions	
[31:2]	**Reserved**	Reserved
[1]	**RX_DMA_GO**	**Receive DMA Start** Set this bit to 1 will start the receive PDMA process. SPI controller will issue request to PDMA controller automatically. Hardware will clear this bit to 0 automatically after PDMA transfer done.
[0]	**TX_DMA_GO**	**Transmit DMA Start** Set this bit to 1 will start the transmit PDMA process. SPI controller will issue request to PDMA controller automatically. If using PDMA mode to transfer data, remember not to set GO_BUSY bit of SPI_CNTRL register. The DMA controller inside SPI controller will set it automatically whenever necessary. Hardware will clear this bit to 0 automatically after PDMA transfer done. Note: In DMA mode, the burst mode and FIFO mode are not supported.

SPI Control and Status Register 2 (SPI_CNTRL2)

Register	Offset	R/W	Description	Reset Value
SPI_CNTRL2	SPIx_BA+0x3C	R/W	The second Control and 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				**SLV_START_INTSTS**	**SSTA_INTEN**	**SLV_ABORT**	**NOSLVSEL**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**DIV_ONE**

Bits	Descriptions	
[31:12]	**Reserved**	Reserved
[11]	**SLV_START_INTSTS**	**Slave Start Interrupt Status** It is used to dedicate that the transfer has start in slave mode with no slave select. 1 = It indicates that the transfer start in slave mode with no slave select. It is auto clear by transfer done or writing one clear. 0 = It indicates that the slave start transfer no active.
[10]	**SSTA_INTEN**	**Slave Start Interrupt Enable** It is used to enable interrupt when the transfer has start in slave mode with no slave select. If there is no transfer done interrupt over the time period which is defined by user after the transfer start, the user can set the SLV_ABORT bit to force the transfer done. 1 = Enable the transaction start interrupt. It is clear by the current transfer done or the SLV_START_INTSTS bit be clear (write one clear). 0 = Disable the transfer start interrupt.
[9]	**SLV_ABORT**	**Abort in Slave Mode with No Slave Select** In normal operation, there is interrupt event when the received data meet the required bits which define in TX_BIT_LEN and TX_NUM. If the received bits are less than the requirement and there is no more serial clock input over the one transfer time in slave mode with no slave select, the user can set this bit to force the current transfer done and then the user can get a transfer done interrupt event. Note: It is auto clear to 0 by hardware when the abort event is active.

[8]	**NOSLVSEL**	**No Slave Select in Slave Mode** This is used to ignore the slave select signal in slave mode. The SPI controller can work on 3 wire interface including SPICLK, SPI_MISO, and SPI_MOSI when it is set as a slave device. 0 = The controller is 4-wire bi-direction interface. 1 = The controller is 3-wire bi-direction interface in slave mode. When this bit is set as 1, the controller start to transmit/receive data after the GO_BUSY bit active and the serial clock input. Note: In no slave select signal mode, the SS_LTRIG, SPI_SSR[4], shall be set as 1.
[7:1]	**Reserved**	Reserved
[0]	**DIV_ONE**	**SPI clock divider control** 0 = The SPI clock rate is determined by the setting of SPI_DIVIDER register. 1 = Enable the DIV_ONE feature. The SPI clock rate equals the system clock rate. Note: 1. When this bit is set to 1, both the REORDER field and the VARCLK_EN field must be configured as 0. In other words, the byte-reorder function, byte suspend function and variable clock function must be disable. 2. When this bit is set to 1, the TX_BIT_LEN can't be set as 1.

SPI FIFO Control Register (SPI_FIFO_CTL)

Register	Offset	R/W	Description	Reset Value
SPI_FIFO_CTL	SPIx_BA+0x40	R/W	SPI FIFO 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						**TX_CLR**	**RX_CLR**

Bits	Descriptions	
[31:2]	**Reserved**	Reserved
[1]	**TX_CLR**	**Clear Tx FIFO** 0 = No effect. 1 = Clear Tx FIFO. The TX_FULL flag will be cleared to 0 and the TX_EMPTY flag will be set to 1. This bit will be cleared to 0 by hardware after software sets it to 1.
[0]	**RX_CLR**	**Clear Rx FIFO** 0 = No effect. 1 = Clear Rx FIFO. The RX_FULL flag will be cleared to 0 and the RX_EMPTY flag will be set to 1. This bit will be cleared to 0 by hardware after software sets it to 1.

5.10 Timer Controller (TMR)

5.10.1 Overview

The timer controller includes four 32-bit timers, TIMER0~TIMER3, which allows user to easily implement a timer control for applications. The timer can perform functions like frequency measurement, event counting, interval measurement, clock generation, delay timing, and so on. The timer can generate an interrupt signal upon timeout, or provide the current value during operation.

5.10.2 Features

- 4 sets of 32-bit timers with 24-bit up-timer and one 8-bit pre-scale counter
- Independent clock source for each timer
- Provides one-shot, periodic, toggle and continuous counting operation modes
- Time out period = (Period of timer clock input) * (8-bit pre-scale counter + 1) * (24-bit TCMP)
- Maximum counting cycle time = (1 / T MHz) * (2^8) * (2^{24}), T is the period of timer clock
- 24-bit timer value is readable through TDR (Timer Data Register)
- Support event counting function to count the event from external pin
- Support input capture function to capture or reset counter value

5.10.3 Block Diagram

Each channel is equipped with an 8-bit pre-scale counter, a 24-bit up-timer, a 24-bit compare register and an interrupt request signal. Refer to Figure 5-64 for the timer controller block diagram. There are four options of clock sources for each channel. Figure 5-65 illustrates the clock source control function. Software can program the 8-bit pre-scale counter to decide the clock period to 24-bit up timer.

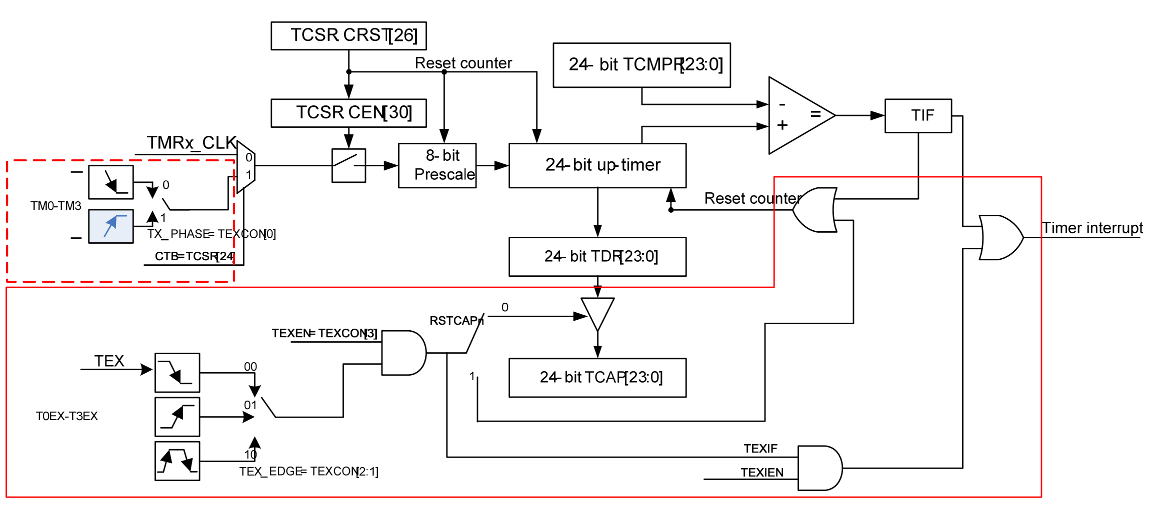


Figure 5-64 Timer Controller Block Diagram

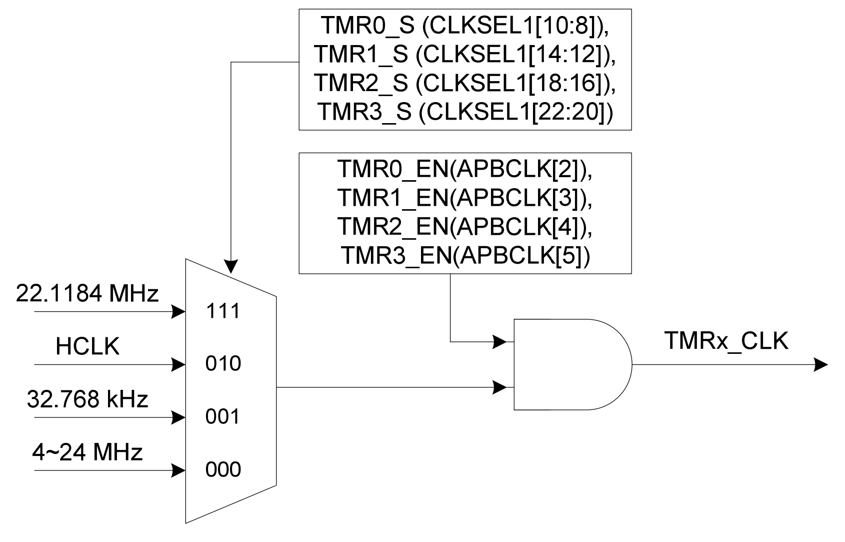


Figure 5-65 Clock Source of Timer Controller

5.10.4 Function Description

Timer controller provides one-shot, period, toggle and continuous counting operation modes. It also provides the event counting function to count the event from external pin and input capture function to capture or reset timer counter value. Each operating function mode is shown as following:

5.10.4.1 One –Shot Mode

If timer is operated at one-shot mode and CEN (TCSR[30] timer enable bit) is set to 1, the timer counter starts up counting. Once the timer counter value reaches timer compare register (TCMPR) value, if IE (TCSR[29] interrupt enable bit) is set to 1, then the timer interrupt flag is set and the interrupt signal is generated and sent to NVIC to inform CPU. It indicates that the timer counting overflow happens. If IE (TCSR[29] interrupt enable bit) is set to 0, no interrupt signal is generated. In this operating mode, once the timer counter value reaches timer compare register (TCMPR) value, the timer counter value goes back to counting initial value and CEN (timer enable bit) is cleared to 0 by timer controller. Timer counting operation stops, once the timer counter value reaches timer compare register (TCMPR) value. That is to say, timer operates timer counting and compares with TCMPR value function only one time after programming the timer compare register (TCMPR) value and CEN (timer enable bit) is set to 1. So, this operating mode is called One-Shot mode.

5.10.4.2 Periodic Mode

If timer is operated at period mode and CEN (TCSR[30] timer enable bit) is set to 1, the timer counter starts up counting. Once the timer counter value reaches timer compare register (TCMPR) value, if IE (TCSR[29] interrupt enable bit) is set to 1, then the timer interrupt flag is set and the interrupt signal is generated and sent to NVIC to inform CPU. It indicates that the timer counting overflow happens. If IE (TCSR[29] interrupt enable bit) is set to 0, no interrupt signal is generated. In this operating mode, once the timer counter value reaches timer compare register (TCMPR) value, the timer counter value goes back to counting initial value and CEN is kept at 1 (counting enable continuously). The timer counter operates up counting again. If the interrupt flag is cleared by software, once the timer counter value reaches timer compare register (TCMPR) value and IE (interrupt enable bit) is set to 1'b1, then the timer interrupt flag is set and the interrupt signal is generated and sent to NVIC to inform CPU again. That is to say, timer operates timer counting and compares with TCMPR value function periodically. The timer counting operation doesn't stop until the CEN is set to 0. The interrupt signal is also generated periodically. So, this operating mode is called Periodic mode.

5.10.4.3 Toggle Mode

If timer is operated at toggle mode and CEN (TCSR[30] timer enable bit) is set to 1, the timer counter starts up counting. Once the timer counter value reaches timer compare register (TCMPR) value, if IE (TCSR[29] interrupt enable bit) is set to 1, then the timer interrupt flag is set and the interrupt signal is generated and sent to NVIC to inform CPU. It indicates that the timer counting overflow happens. The associated toggle output (tout) signal is set to 1. In this operating mode, once the timer counter value reaches timer compare register (TCMPR) value, the timer counter value goes back to counting initial value and CEN is kept at 1 (counting enable continuously). The timer counter operates up counting again. If the interrupt flag is cleared by software, once the timer counter value reaches timer compare register (TCMPR) value and IE (interrupt enable bit) is set to 1, then the timer interrupt flag is set and the interrupt signal is generated and sent to NVIC to inform CPU again. The associated toggle output (tout) signal is set to 0. The timer counting operation doesn't stop until the CEN is set to 0. Thus, the toggle output (tout) signal is changing back and forth with 50% duty cycle. So, this operating mode is called Toggle mode.

5.10.4.4 Continuous Counting Mode

If the timer is operated at continuous counting mode and CEN (TCSR[30] timer enable bit) is set to 1, the associated interrupt signal is generated depending on TDR = TCMPR if IE (TCSR[29] interrupt enable bit) is enabled. User can change different TCMPR value immediately without disabling timer counting and restarting timer counting. For example, TCMPR is set as 80, first. (The TCMPR should be less than 2^{24} and be greater than 1). The timer generates the interrupt if IE is enabled and TIF (timer interrupt flag) will set to 1 then the interrupt signal is generated and sent to NVIC to inform CPU when TDR value is equal to 80. But the CEN is kept at 1 (counting enable continuously) and TDR 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 TCMPR as 200 and the TIF is cleared to 0, then timer interrupt occurred and TIF is set to 1, then the interrupt signal is generated and sent to NVIC to inform CPU again when TDR value reaches to 200. At last, user programs TCMPR as 500 and clears TIF to 0 again, then timer interrupt occurred and TIF sets to 1 then the interrupt signal is generated and sent to NVIC to inform CPU when TDR value reaches to 500. From application view, the interrupt is generated depending on TCMPR. In this mode, the timer counting is continuous. So, this operation mode is called as continuous counting mode.

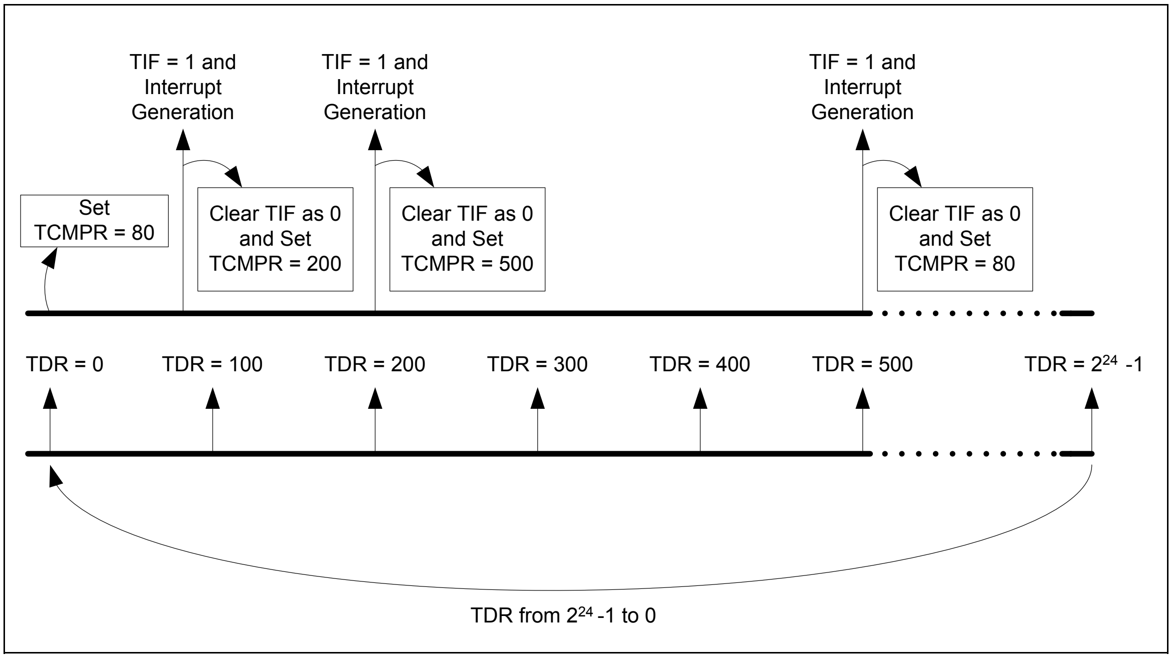


Figure 5-66 Continuous Counting Mode

5.10.4.5 Event Counting Function

It also provides an application which can count the event from TM0~TM3 pins. It is called as event counting function. In event counting function, the clock source of timer controller, TMRx_CLK, in Figure 5-65 should be set as HCLK. It provides TM0~TM3 enabled or disabled de-bounce function by TEXCONx[7] and TM0~TM3 falling or rising phase counting setting by TEXCONx[0]. And, the event count source operating frequency should be less than 1/3 HCLK frequency if disable counting de-bounce or less than 1/8 HCLK frequency if enable counting de-bounce. Otherwise, the returned TDR value is incorrect.

5.10.4.6 Input Capture Function

It also provides input capture function to capture or reset timer counter value. If TEXEN (Timer External Pin Enable) is set to 1 and RSTCAPn is set to 0, the timer counter value (TDR) will be captured into TCAP register when TEX (Timer External Pin) pin trigger condition occurred. There are four TEX sources form specified pins, T0EX~T3EX pins. If TEXEN is set to 1 and RSTCAPn is set to 1, the TDR will be reset to 0 when TEX pin trigger condition happened. The TEX trigger edge can choose by TEX_EDGE. When TEX trigger occurred, TEXIF (Timer External Interrupt Flag) is set to 1, and if enabled TEXIEN (Timer External Interrupt Enable Bit) to 1, the interrupt signal is generated then sent to NVIC to inform CPU. It also provides T0EX~T3EX enabled or disabled capture de-bounce function by TEXCONx[6]. And, the TEX source operating frequency should be less than 1/3 HCLK frequency if disable TEX de-bounce or less than 1/8 HCLK frequency if enable TEX de-bounce.

5.10.5 Register Map

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

Register	Offset	R/W	Description	Reset Value
TMR_BA01 = 0x4001_0000 **TMR_BA23 = 0x4011_0000**				
TCSR0	TMR_BA01+0x00	R/W	Timer0 Control and Status Register	0x0000_0005
TCMPR0	TMR_BA01+0x04	R/W	Timer0 Compare Register	0x0000_0000
TISR0	TMR_BA01+0x08	R/W	Timer0 Interrupt Status Register	0x0000_0000
TDR0	TMR_BA01+0x0C	R	Timer0 Data Register	0x0000_0000
TCAP0	TMR_BA01+0x10	R	Timer0 Capture Data Register	0x0000_0000
TEXCON0	TMR_BA01+0x14	R/W	Timer0 External Control Register	0x0000_0000
TEXISR0	TMR_BA01+0x18	R/W	Timer0 External Interrupt Status Register	0x0000_0000
TCSR1	TMR_BA01+0x20	R/W	Timer1 Control and Status Register	0x0000_0005
TCMPR1	TMR_BA01+0x24	R/W	Timer1 Compare Register	0x0000_0000
TISR1	TMR_BA01+0x28	R/W	Timer1 Interrupt Status Register	0x0000_0000
TDR1	TMR_BA01+0x2C	R	Timer1 Data Register	0x0000_0000
TCAP1	TMR_BA01+0x30	R	Timer1 Capture Data Register	0x0000_0000
TEXCON1	TMR_BA01+0x34	R/W	Timer1 External Control Register	0x0000_0000
TEXISR1	TMR_BA01+0x38	R/W	Timer1 External Interrupt Status Register	0x0000_0000
TCSR2	TMR_BA23+0x00	R/W	Timer2 Control and Status Register	0x0000_0005
TCMPR2	TMR_BA23+0x04	R/W	Timer2 Compare Register	0x0000_0000
TISR2	TMR_BA23+0x08	R/W	Timer2 Interrupt Status Register	0x0000_0000
TDR2	TMR_BA23+0x0C	R	Timer2 Data Register	0x0000_0000
TCAP2	TMR_BA23+0x10	R	Timer2 Capture Data Register	0x0000_0000
TEXCON2	TMR_BA23+0x14	R/W	Timer2 External Control Register	0x0000_0000
TEXISR2	TMR_BA23+0x18	R/W	Timer2 External Interrupt Status Register	0x0000_0000
TCSR3	TMR_BA23+0x20	R/W	Timer3 Control and Status Register	0x0000_0005
TCMPR3	TMR_BA23+0x24	R/W	Timer3 Compare Register	0x0000_0000
TISR3	TMR_BA23+0x28	R/W	Timer3 Interrupt Status Register	0x0000_0000

TDR3	TMR_BA23+0x2C	R	Timer3 Data Register	0x0000_0000
TCAP3	TMR_BA23+0x30	R	Timer3 Capture Data Register	0x0000_0000
TEXCON3	TMR_BA23+0x34	R/W	Timer3 External Control Register	0x0000_0000
TEXISR3	TMR_BA23+0x38	R/W	Timer3 External Interrupt Status Register	0x0000_0000

5.10.6 Register Description

Timer Control Register (TCSR)

Register	Offset	R/W	Description	Reset Value
TCSR0	TMR_BA01+0x00	R/W	Timer0 Control and Status Register	0x0000_0005
TCSR1	TMR_BA01+0x20	R/W	Timer1 Control and Status Register	0x0000_0005
TCSR2	TMR_BA23+0x00	R/W	Timer2 Control and Status Register	0x0000_0005
TCSR3	TMR_BA23+0x20	R/W	Timer3 Control and Status Register	0x0000_0005

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**
DBGACK_TMR	**CEN**	**IE**	**MODE[1:0]**		**CRST**	**CACT**	**CTB**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							**TDR_EN**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRESCALE[7:0]							

Bits	Descriptions	
[31]	**DBGACK_TMR**	**ICE debug mode acknowledge Disable** (write-protection bit) 0 = ICE debug mode acknowledgement effects TIMER counting. TIMER counter will be held while ICE debug mode acknowledged. 1 = ICE debug mode acknowledgement disabled. TIMER counter will keep going no matter ICE debug mode acknowledged or not.
[30]	**CEN**	**Timer Enable Bit** 1 = Starts counting 0 = Stops/Suspends counting Note1: In stop status, and then set CEN to 1 will enables the 24-bit up-timer keeps up counting from the last stop counting value. Note2: This bit is auto-cleared by hardware in one-shot mode (MODE [28:27] =00) when the associated timer interrupt is generated (IE [29] =1).
[29]	**IE**	**Interrupt Enable Bit** 1 = Enable timer Interrupt 0 = Disable timer Interrupt If timer interrupt is enabled, the timer asserts its interrupt signal when the associated

		up-timer value is equal to TCMPR.
[28:27]	**MODE**	**Timer Operating Mode** <table><tr><th>MODE</th><th>Timer Operating Mode</th></tr><tr><td>00</td><td>The timer is operating at the one-shot mode. The associated interrupt signal is generated once (if IE is enabled) and CEN is automatically cleared by hardware.</td></tr><tr><td>01</td><td>The timer is operating at the periodic mode. The associated interrupt signal is generated periodically (if IE is enabled).</td></tr><tr><td>10</td><td>The timer is operating at the toggle mode. The interrupt signal is generated periodically (if IE is enabled). And the associated signal (tout) is changing back and forth with 50% duty cycle.</td></tr><tr><td>11</td><td>The timer is operating at continuous counting mode. The associated interrupt signal is generated when TDR = TCMPR (if IE is enabled). However, the 24-bit up-timer counts continuously. Please refer 5.10.4.4 for detail description about continuous counting mode operation.</td></tr></table>
[26]	**CRST**	**Timer Reset Bit** Set this bit will reset the 24-bit up-timer, 8-bit pre-scale counter and also force CEN to 0. 0 = No effect 1 = Reset Timer’s 8-bit pre-scale counter, internal 24-bit up-timer and CEN bit
[25]	**CACT**	**Timer Active Status Bit (Read only)** This bit indicates the up-timer status. 0 = Timer is not active 1 = Timer is active
[24]	**CTB**	**Counter Mode Enable Bit** This bit is the counter mode enable bit. When Timer is used as an event counter, this bit should be set to 1 and Timer will work as an event counter. The counter detect phase can be selected as rising/falling edge of external pin by TX_PHASE field. 1 = Enable counter mode 0 = Disable counter mode
[23:17]	**Reserved**	Reserved
[16]	**TDR_EN**	**Data Load Enable** When TDR_EN is set, TDR (Timer Data Register) will be updated continuously with the 24-bit up-timer value as the timer is counting. 1 = Timer Data Register update enable 0 = Timer Data Register update disable
[15:8]	**Reserved**	Reserved
[7:0]	**PRESCALE**	**Pre-scale Counter** Clock input is divided by PRESCALE+1 before it is fed to the counter. If PRESCALE = 0, then there is no scaling.

Timer Compare Register (TCMPR)

Register	Offset	R/W	Description	Reset Value
TCMPR0	TMR_BA01+0x04	R/W	Timer0 Compare Register	0x0000_0000
TCMPR1	TMR_BA01+0x24	R/W	Timer1 Compare Register	0x0000_0000
TCMPR2	TMR_BA23+0x04	R/W	Timer2 Compare Register	0x0000_0000
TCMPR3	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**
TCMP [23:16]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TCMP [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TCMP [7:0]							

Bits	Descriptions	
[31:24]	**Reserved**	Reserved
[23:0]	**TCMP**	**Timer Compared Value** TCMP is a 24-bit compared register. When the internal 24-bit up-timer counts and its value is equal to TCMP value, a Timer Interrupt is requested if the timer interrupt is enabled with TCSR.IE[29]=1. The TCMP value defines the timer counting cycle time. Time out period = (Period of timer clock input) * (8-bit PRESCALE + 1) * (24-bit TCMP) Note1: Never write 0x0 or 0x1 in TCMP, or the core will run into unknown state. Note2: When timer is operating at continuous counting mode, the 24-bit up-timer will count continuously if software writes a new value into TCMP. If timer is operating at other modes, the 24-bit up-timer will restart counting and using newest TCMP value to be the compared value if software writes a new value into TCMP.

Timer Interrupt Status Register (TISR)

Register	Offset	R/W	Description	Reset Value
TISR0	TMR_BA01+0x08	R/W	Timer0 Interrupt Status Register	0x0000_0000
TISR1	TMR_BA01+0x28	R/W	Timer1 Interrupt Status Register	0x0000_0000
TISR2	TMR_BA23+0x08	R/W	Timer2 Interrupt Status Register	0x0000_0000
TISR3	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							**TIF**

Bits	Descriptions	
[31:1]	**Reserved**	Reserved
[0]	**TIF**	**Timer Interrupt Flag** This bit indicates the interrupt status of Timer. TIF bit is set by hardware when the up counting value of internal 24-bit up-timer matches the timer compared value (TCMP). It is cleared by writing 1 to this bit.

Timer Data Register (TDR)

Register	Offset	R/W	Description	Reset Value
TDR0	TMR_BA01+0x0C	R/W	Timer0 Data Register	0x0000_0000
TDR1	TMR_BA01+0x2C	R/W	Timer1 Data Register	0x0000_0000
TDR2	TMR_BA23+0x0C	R/W	Timer2 Data Register	0x0000_0000
TDR3	TMR_BA23+0x2C	R/W	Timer3 Data Register	0x0000_0000

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

Bits	Descriptions	
[31:24]	**Reserved**	Reserved
[23:0]	**TDR**	**Timer Data Register** 1. CTB (TCSR[24]) = 0 : TDR is 24- bits up timer value. User can read TDR for getting current 24- bits up timer value if TCSR[24] = is set to 0 2. CTB (TCSR[24]) = 1 : TDR is 24- bits up event counter value. User can read TDR for getting current 24- bits up event counter value if TCSR[24] is 1

Timer Capture Data Register (TCAP)

Register	Offset	R/W	Description	Reset Value
TCAP0	TMR_BA01+0x10	R/W	Timer0 Capture Data Register	0x0000_0000
TCAP1	TMR_BA01+0x30	R/W	Timer1 Capture Data Register	0x0000_0000
TCAP2	TMR_BA23+0x10	R/W	Timer2 Capture Data Register	0x0000_0000
TCAP3	TMR_BA23+0x30	R/W	Timer3 Capture Data Register	0x0000_0000

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

Bits	Descriptions	
[31:24]	**Reserved**	Reserved
[23:0]	**TCAP**	**Timer Capture Data Register** When TEXEN (TEXCON[3]) is set, RSTCAPn(TTXCON[4]) is 0, and the transition on the TEX pins associated TEX_EDGE(TEXCON[2:1]) setting is occurred, the internal 24-bit up-timer value will be loaded into TCAP. User can read this register for the counter value.

Timer External Control Register (TEXCON)

Register	Offset	R/W	Description	Reset Value
TEXCON0	TMR_BA01+0x14	R/W	Timer0 External Control Register	0x0000_0000
TEXCON1	TMR_BA01+0x34	R/W	Timer1 External Control Register	0x0000_0000
TEXCON2	TMR_BA23+0x14	R/W	Timer2 External Control Register	0x0000_0000
TEXCON3	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**
TCDB	TEXDB	TEXIEN	RSTCAPn	TEXEN	TEX_EDGE		TX_PHASE

Bits	Descriptions	
[31:8]	**Reserved**	Reserved
[7]	**TCDB**	**Timer Counter pin De-bounce enable bit** 1 = Enable De-bounce 0 = Disable De-bounce If this bit is enabled, the edge of TM0~TM3 pin is detected with de-bounce circuit.
[6]	**TEXDB**	**Timer External Capture pin De-bounce enable bit** 1 = Enable De-bounce 0 = Disable De-bounce If this bit is enabled, the edge of T0EX~T3EX pin is detected with de-bounce circuit.
[5]	**TEXIEN**	**Timer External interrupt Enable Bit** 1 = Enable timer External Interrupt 0 = Disable timer External Interrupt If timer external interrupt is enabled, the timer asserts its external interrupt signal and sent to NVIC to inform CPU when the transition on the TEX pins associated with TEX_EDGE(TEXCON[2:1]) setting is happened. For example, while TEXIEN = 1, TEXEN = 1, and TEX_EDGE = 00, a 1 to 0 transition on the TEX pin will cause the TEXIF(TEXISR[0]) interrupt flag to be set then the interrupt signal is generated and sent to NVIC to inform CPU.
[4]	**RSTCAPn**	**Timer External Reset Counter / Capture mode select**

		1 = TEX transition is using as the timer counter reset function. 0 = TEX transition is using as the timer capture function.
[3]	**TEXEN**	**Timer External Pin Enable.** This bit enables the reset/capture function on the TEX pin. 1 = The transition detected on the TEX pin will result in capture or reset of timer counter. 0 = The TEX pin will be ignored.
[2:1]	**TEX_EDGE**	**Timer External Pin Edge Detect** 00 = a 1 to 0 transition on TEX will be detected. 01 = a 0 to 1 transition on TEX will be detected. 10 = either 1 to 0 or 0 to 1 transition on TEX will be detected. 11 = Reserved.
[0]	**TX_PHASE**	**Timer External Count Phase** This bit indicates the external count pin phase. 1 = A rising edge of external count pin will be counted. 0 = A falling edge of external count pin will be counted.

Timer External Interrupt Status Register (TISR)

Register	Offset	R/W	Description	Reset Value
TEXISR0	TMR_BA01+0x18	R/W	Timer0 External Interrupt Status Register	0x0000_0000
TEXISR1	TMR_BA01+0x38	R/W	Timer1 External Interrupt Status Register	0x0000_0000
TEXISR2	TMR_BA23+0x18	R/W	Timer2 External Interrupt Status Register	0x0000_0000
TEXISR3	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							TEXIF

Bits	Descriptions	
[31:1]	**Reserved**	Reserved
[0]	**TEXIF**	**Timer External Interrupt Flag** This bit indicates the external interrupt status of Timer. This bit is set by hardware when TEXEN (TEXCON[3]) is to 1, and the transition on the TEX pins associated with TEX_EDGE (TEXCON[2:1]) setting is occurred. It is cleared by writing 1 to this bit. For example, while TEXEN = 1, and TEX_EDGE = 00, a 1 to 0 transition on the TEX pin causes the TEXIF to be set.

5.11 Watchdog Timer (WDT)

5.11.1 Overview

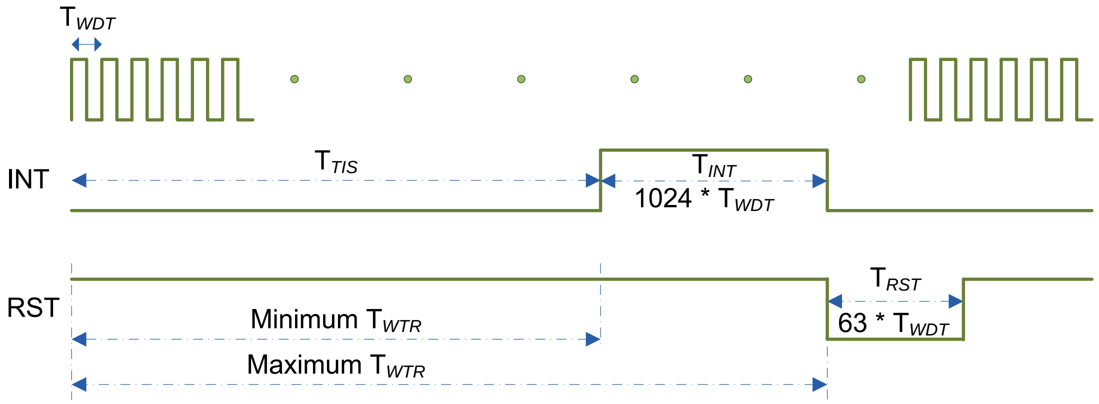
The purpose of Watchdog Timer is to perform a system reset when system runs into an unknown state. This prevents system from hanging for an infinite period of time. Besides, this Watchdog Timer supports another function to wake-up chip from power down mode. The watchdog timer includes an 18-bit free running counter with programmable time-out intervals. Table 5-7 show the watchdog timeout interval selection and Figure 5-64 shows the timing of watchdog interrupt signal and reset signal.

Setting WTE (WDTCR [7]) enables the watchdog timer and the WDT counter starts counting up. When the counter reaches the selected time-out interval, Watchdog timer interrupt flag WTIF will be set immediately to request a WDT interrupt if the watchdog timer interrupt enable bit WTIE is set, in the meanwhile, a specified delay time (1024 * T_{WDT}) follows the time-out event. User must set WTR (WDTCR [0]) (Watchdog timer reset) high to reset the 18-bit WDT counter to avoid chip from Watchdog timer reset before the delay time expires. WTR bit is cleared automatically by hardware after WDT counter is reset. There are eight time-out intervals with specific delay time which are selected by Watchdog timer interval select bits WTIS (WDTCR [10:8]). If the WDT counter has not been cleared after the specific delay time expires, the watchdog timer will set Watchdog Timer Reset Flag (WTRF) high and reset chip. This reset will last 63 WDT clocks (T_{RST}) then chip restarts executing program from reset vector (0x0000_0000). WTRF will not be cleared by Watchdog reset. User may poll WTRF by software to recognize the reset source. WDT also provides wake-up function. When chip is powered down and the Watchdog Timer Wake-up Function Enable bit (WDTR[4]) is set, if the WDT counter reaches the specific time interval defined by WTIS (WDTCR [10:8]) , the chip is waken up from power down state. First example, if WTIS is set as 000, the specific time interval for chip to wake up from power down state is 2^4 * T_{WDT}. When power down command is set by software, then, chip enters power down state. After 2^4 * T_{WDT} time is elapsed, chip is waken up from power down state. Second example, if WTIS (WDTCR [10:8]) is set as 111, the specific time interval for chip to wake up from power down state is 2^{18} * T_{WDT}. If power down command is set by software, then, chip enters power down state. After 2^{18} * T_{WDT} time is elapsed, chip is waken up from power down state. Notice if WTRE (WDTCR [1]) is set to 1, after chip is waken up, software should clear the Watchdog Timer counter by setting WTR(WDTCR [0]) to 1 as soon as possible. Otherwise, if the Watchdog Timer counter is not cleared by setting WTR (WDTCR [0]) to 1 before time starting from waking up to software clearing Watchdog Timer counter is over 1024 * T_{WDT} , the chip is reset by Watchdog Timer.

WTIS	Timeout Interval Selection T_{TIS}	Interrupt Period T_{INT}	WTR Timeout startingInterval (WDT_CLK=10 kHz) MIN. T_{WTR} ~ Max. T_{WTR}
000	2^4 * T_{WDT}	1024 * T_{WDT}	1.6 ms ~ 104 ms
001	2^6 * T_{WDT}	1024 * T_{WDT}	6.4 ms ~ 108.8 ms
010	2^8 * T_{WDT}	1024 * T_{WDT}	25.6 ms ~ 128 ms
011	2^{10} * T_{WDT}	1024 * T_{WDT}	102.4 ms ~ 204.8 ms
100	2^{12} * T_{WDT}	1024 * T_{WDT}	409.6 ms ~ 512 ms
101	2^{14} * T_{WDT}	1024 * T_{WDT}	1.6384 s ~ 1.7408 s

110	2^{16} * T_{WDT}	1024 * T_{WDT}	6.5536 s ~ 6.656 s
111	2^{18} * T_{WDT}	1024 * T_{WDT}	26.2144 s ~ 26.3168 s

Table 5-7 Watchdog Timeout Interval Selection



- T_{WDT} : *Watchdog Engine Clock Time Period*
- T_{TIS} : *Watchdog Timeout Interval Selection Period*
- T_{INT} : *Watchdog Interrupt Period*
- T_{RST} : *Watchdog Reset Period*
- T_{WTR} : *Watchdog Timeout Interval Period*

Figure 5-67 Timing of Interrupt and Reset Signal

5.11.2 Features

- 18-bit free running counter to avoid chip from Watchdog timer reset before the delay time expires.
- Selectable time-out interval (2^4 ~ 2^18) and the time out interval is 104 ms ~ 26.3168 s (if WDT_CLK = 10 kHz).
- Reset period = (1 / 10 kHz) * 63, if WDT_CLK = 10 kHz.

5.11.3 Block Diagram

The Watchdog Timer clock control and block diagram are shown as following.

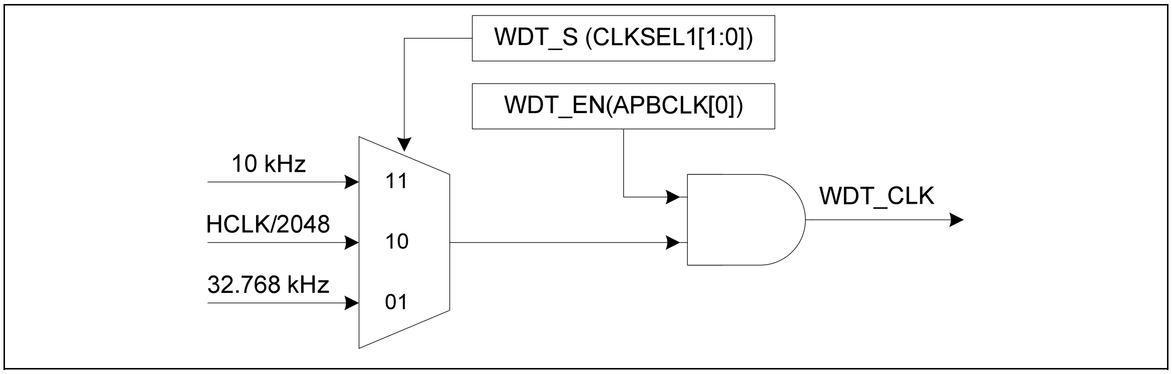
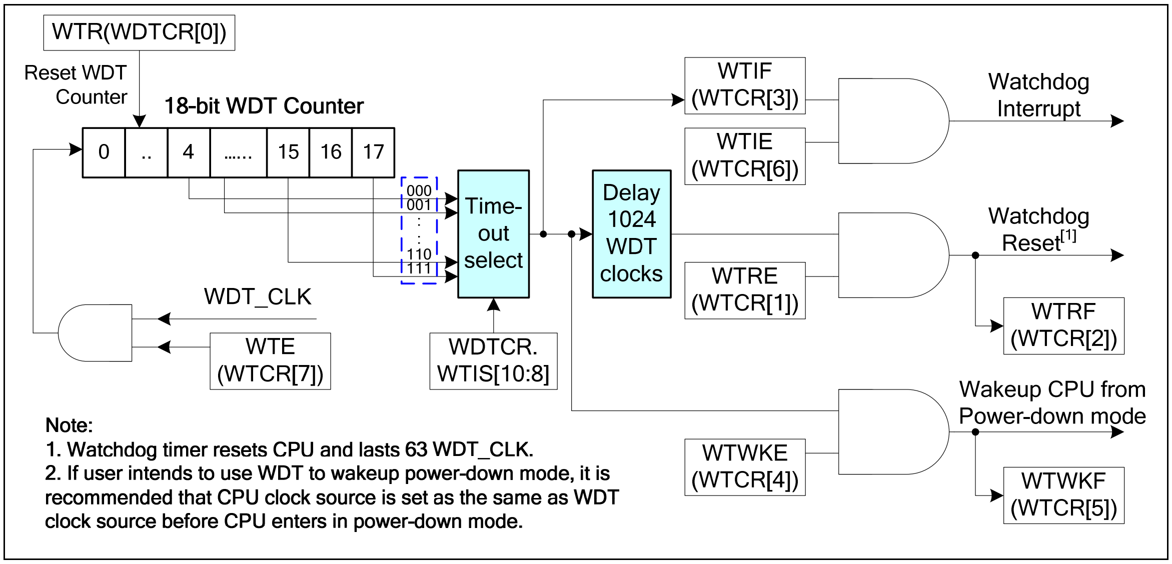


Figure 5-68 Watchdog Timer Clock Control



Note:
1. Watchdog timer resets CPU and lasts 63 WDT_CLK.
2. If user intends to use WDT to wakeup power-down mode, it is recommended that CPU clock source is set as the same as WDT clock source before CPU enters in power-down mode.

Figure 5-69 Watchdog Timer Block Diagram

5.11.4 Register Map

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

Register	Offset	R/W	Description	Reset Value
WDT_BA = 0x4000_4000				
WTCR	WDT_BA+0x00	R/W	Watchdog Timer Control Register	0x0000_0700

5.11.5 Register Description

Watchdog Timer Control Register (WTCR)

Register	Offset	R/W	Description	Reset Value
WTCR	WDT_BA+0x00	R/W	Watchdog Timer Control Register	0x0000_0700

Note: All bits can be write in this register are write-protected. To program it needs to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Reference the register REGWRPROT at address GCR_BA+0x100.

31	30	29	28	27	26	25	24
DBGACK_WDT	**Reserved**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**WTIS**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
WTE	**WTIE**	**WTWKF**	**WTWKE**	**WTIF**	**WTRF**	**WTRE**	**WTR**

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[31]</td><td>DBGACK_WDT</td><td>ICE debug mode acknowledge Disable (write-protection bit)
0 = ICE debug mode acknowledgement effects Watchdog Timer counting.
Watchdog Timer counter will be held while ICE debug mode acknowledged.
1 = ICE debug mode acknowledgement disabled.
Watchdog Timer counter will keep going no matter ICE debug mode acknowledged or not.</td></tr>
<tr><td>[30:11]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[10:8]</td><td>WTIS</td><td>Watchdog Timer Interval Select (write-protection bits)
These three bits select the timeout interval for the Watchdog timer.
<table>
<tr><th>WTIS</th><th>Timeout Interval Selection</th><th>Interrupt Period</th><th>WTR Timeout Interval (WDT_CLK=10 kHz)</th></tr>
<tr><td>000</td><td>$2^4 * T_{WDT}$</td><td>$(2^4 + 1024) * T_{WDT}$</td><td>1.6 ms ~ 104 ms</td></tr>
<tr><td>001</td><td>$2^6 * T_{WDT}$</td><td>$(2^6 + 1024) * T_{WDT}$</td><td>6.4 ms ~ 108.8 ms</td></tr>
<tr><td>010</td><td>$2^8 * T_{WDT}$</td><td>$(2^8 + 1024) * T_{WDT}$</td><td>25.6 ms ~ 128 ms</td></tr>
<tr><td>011</td><td>$2^{10} * T_{WDT}$</td><td>$(2^{10} + 1024) * T_{WDT}$</td><td>102.4 ms ~ 204.8 ms</td></tr>
<tr><td>100</td><td>$2^{12} * T_{WDT}$</td><td>$(2^{12} + 1024) * T_{WDT}$</td><td>409.6 ms ~ 512 ms</td></tr>
</table></td></tr>
</table>

101	$2^{14} * T_{WDT}$	$(2^{14} + 1024) * T_{WDT}$	1.6384 s ~ 1.7408 s
110	$2^{16} * T_{WDT}$	$(2^{16} + 1024) * T_{WDT}$	6.5536 s ~ 6.656 s
111	$2^{18} * T_{WDT}$	$(2^{18} + 1024) * T_{WDT}$	26.2144 s ~ 26.3168 s

[7]	**WTE**	**Watchdog Timer Enable** (write-protection bit) 0 = Disable the Watchdog timer (This action will reset the internal counter) 1 = Enable the Watchdog timer
[6]	**WTIE**	**Watchdog Timer Interrupt Enable** (write-protection bit) 0 = Disable the Watchdog timer interrupt 1 = Enable the Watchdog timer interrupt
[5]	**WTWKF**	**Watchdog Timer Wake-up Flag** If Watchdog timer causes chip wakes up from power down mode, this bit will be set to high. It must be cleared by software with a write 1 to this bit. 0 = Watchdog timer does not cause chip wake-up. 1 = Chip wake-up from idle or power down mode by Watchdog timeout.
[4]	**WTWKE**	**Watchdog Timer Wake-up Function Enable bit** (write-protection bit) 0 = Disable Watchdog timer Wake-up chip function. 1 = Enable the Wake-up function that Watchdog timer timeout can wake-up chip from power down mode. Note: Chip can wake-up by WDT only if WDT clock source select RC10K
[3]	**WTIF**	**Watchdog Timer Interrupt Flag** If the Watchdog timer interrupt is enabled, then the hardware will set this bit to indicate that the Watchdog timer interrupt has occurred. 0 = Watchdog timer interrupt did not occur 1 = Watchdog timer interrupt occurs Note: This bit is cleared by writing 1 to this bit.
[2]	**WTRF**	**Watchdog Timer Reset Flag** When the Watchdog timer initiates a reset, the hardware will set this bit. This flag can be read by software to determine the source of reset. Software is responsible to clear it manually by writing 1 to it. If WTRE is disabled, then the Watchdog timer has no effect on this bit. 0 = Watchdog timer reset did not occur 1 = Watchdog timer reset occurs Note: This bit is cleared by writing 1 to this bit.
[1]	**WTRE**	**Watchdog Timer Reset Enable** (write-protection bit) Setting this bit will enable the Watchdog timer reset function. 0 = Disable Watchdog timer reset function 1 = Enable Watchdog timer reset function
[0]	**WTR**	**Clear Watchdog Timer** (write-protection bit) Set this bit will clear the Watchdog timer. 0 = Writing 0 to this bit has no effect 1 = Reset the contents of the Watchdog timer

		Note: This bit will be auto cleared by hardware

5.12 UART Interface Controller (UART)

NuMicro™ NUC130/NUC140 provides up to three channels of Universal Asynchronous Receiver/Transmitters (UART). UART0 supports High Speed UART and UART1~2 perform Normal Speed UART, besides, only UART0 and UART1 support flow control function.

5.12.1 Overview

The Universal Asynchronous Receiver/Transmitter (UART) performs a serial-to-parallel conversion on data received from the peripheral, and a parallel-to-serial conversion on data transmitted from the CPU. The UART controller also supports IrDA SIR Function, LIN master/slave mode function and RS-485 mode functions. Each UART channel supports seven types of interrupts including transmitter FIFO empty interrupt (INT_THRE), receiver threshold level reaching interrupt (INT_RDA), line status interrupt (parity error or framing error or break interrupt) (INT_RLS), receiver buffer time out interrupt (INT_TOUT), MODEM/Wake-up status interrupt (INT_MODEM), Buffer error interrupt (INT_BUF_ERR) and LIN receiver break field detected interrupt (INT_LIN_RX_BREAK). Interrupts of UART0 and UART2 share the interrupt number 12 (vector number is 28); Interrupt number 13 (vector number is 29) only supports UART1 interrupt. Refer to Nested Vectored Interrupt Controller chapter for System Interrupt Map.

The UART0 is built-in with a 64-byte transmitter FIFO (TX_FIFO) and a 64-byte receiver FIFO (RX_FIFO) that reduces the number of interrupts presented to the CPU and the UART1~2 are equipped 16-byte transmitter FIFO (TX_FIFO) and 16-byte receiver FIFO (RX_FIFO). The CPU can read the status of the UART at any time during the operation. The reported status information includes the type and condition of the transfer operations being performed by the UART, as well as 4 error conditions (parity error, framing error, break interrupt and buffer error) probably occur while receiving data. The UART includes a programmable baud rate generator that is capable of dividing clock input by divisors to produce the serial clock that transmitter and receiver need. The baud rate equation is Baud Rate = UART_CLK / M * [BRD + 2], where M and BRD are defined in Baud Rate Divider Register (UA_BAUD). Table 5-8 lists the equations in the various conditions and Table 5-9 list the UART baud rate setting table.

Mode	DIV_X_EN	DIV_X_ONE	Divider X	BRD	Baud rate equation
0	0	0	B	A	UART_CLK / [16 * (A+2)]
1	1	0	B	A	UART_CLK / [(B+1) * (A+2)] , B must >= 8
2	1	1	Don’t care	A	UART_CLK / (A+2), A must >=3

Table 5-8 UART Baud Rate Equation

System clock = internal 22.1184 MHz high speed oscillator						
Baud rate	Mode0		Mode1		Mode2	
	Parameter	Register	Parameter	Register	Parameter	Register
921600	x	x	A=0,B=11	0x2B00_0000	A=22	0x3000_0016
460800	A=1	0x0000_0001	A=1,B=15 A=2,B=11	0x2F00_0001 0x2B00_0002	A=46	0x3000_002E

230400	A=4	0x0000_0004	A=4,B=15 A=6,B=11	0x2F00_0004 0x2B00_0006	A=94	0x3000_005E
115200	A=10	0x0000_000A	A=10,B=15 A=14,B=11	0x2F00_000A 0x2B00_000E	A=190	0x3000_00BE
57600	A=22	0x0000_0016	A=22,B=15 A=30,B=11	0x2F00_0016 0x2B00_001E	A=382	0x3000_017E
38400	A=34	0x0000_0022	A=62,B=8 A=46,B=11 A=34,B=15	0x2800_003E 0x2B00_002E 0x2F00_0022	A=574	0x3000_023E
19200	A=70	0x0000_0046	A=126,B=8 A=94,B=11 A=70,B=15	0x2800_007E 0x2B00_005E 0x2F00_0046	A=1150	0x3000_047E
9600	A=142	0x0000_008E	A=254,B=8 A=190,B=11 A=142,B=15	0x2800_00FE 0x2B00_00BE 0x2F00_008E	A=2302	0x3000_08FE
4800	A=286	0x0000_011E	A=510,B=8 A=382,B=11 A=286,B=15	0x2800_01FE 0x2B00_017E 0x2F00_011E	A=4606	0x3000_11FE

Table 5-9 UART Baud Rate Setting Table

The UART0 and UART1 controllers support auto-flow control function that uses two low-level signals, /CTS (clear-to-send) and /RTS (request-to-send), to control the flow of data transfer between the UART and external devices (ex: Modem). When auto-flow is enabled, the UART is not allowed to receive data until the UART asserts /RTS to external device. When the number of bytes in the RX FIFO equals the value of RTS_TRI_LEV (UA_FCR [19:16]), the /RTS is de-asserted. The UART sends data out when UART controller detects /CTS is asserted from external device. If a valid asserted /CTS is not detected the UART controller will not send data out.

The UART controllers also provides Serial IrDA (SIR, Serial Infrared) function (User must set IrDA_EN (UA_FUN_SEL [1]) to enable 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 (half duplex). 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 10ms transfer delay between transmission and reception. This delay feature must be implemented by software.

The alternate function of UART controllers is LIN (Local Interconnect Network) function. The LIN mode is selected by setting the LIN_EN bit in UA_FUN_SEL register. In LIN mode, one start bit and 8-bit data format with 1-bit stop bit are required in accordance with the LIN standard.

For NuMicro™ NUC100 Series, another alternate function of UART controllers is RS-485 9-bit mode function, and direction control provided by RTS pin or can program GPIO (PB.2 for RTS0 and PB.6 for RTS1) to implement the function by software. The RS-485 mode is selected by setting the UA_FUN_SEL register to select RS-485 function. The RS-485 driver control is implemented using the RTS control signal from an asynchronous serial port to enable the RS-485 driver. In RS-485 mode, many characteristics of the RX and TX are same as UART.

5.12.2 Features

- Full duplex, asynchronous communications
- Separate receive / transmit 64/16/16 bytes (UART0/UART1/UART2) entry FIFO for data payloads
- Support hardware auto flow control/flow control function (CTS, RTS) and programmable RTS flow control trigger level (UART0 and UART1 support)
- Programmable receiver buffer trigger level
- Support programmable baud-rate generator for each channel individually
- Support CTS wake-up function (UART0 and UART1 support)
- Support 7-bit receiver buffer time out detection function
- UART0/UART1 can be served by the DMA controller
- Programmable transmitting data delay time between the last stop and the next start bit by setting UA_TOR [DLY] register
- Support break error, frame error, parity error and receive / transmit buffer overflow detect 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
- Support IrDA SIR function mode
  - Support for 3-/16-bit duration for normal mode
- Support LIN function mode
  - Support LIN master/slave mode
  - Support programmable break generation function for transmitter
  - Support break detect function for receiver
- Support RS-485 function mode.
  - Support RS-485 9-bit mode
  - Support hardware or software direct enable control provided by RTS pin

5.12.3 Block Diagram

The UART clock control and block diagram are shown as Figure 5-67 and Figure 5-68.

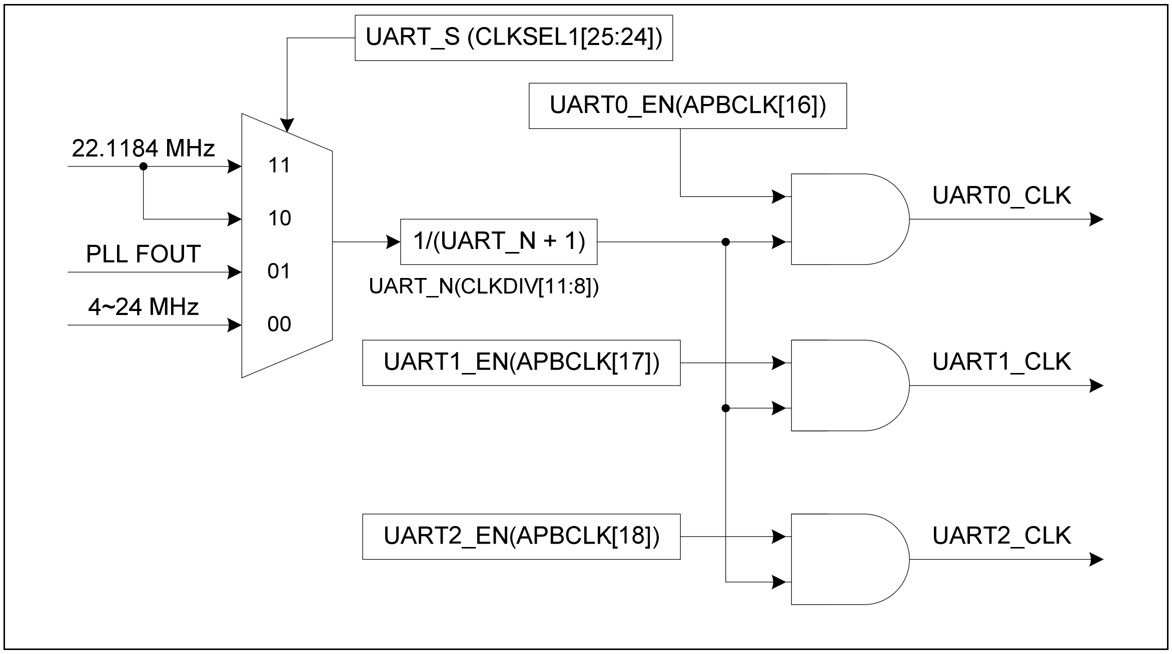


Figure 5-70 UART Clock Control Diagram

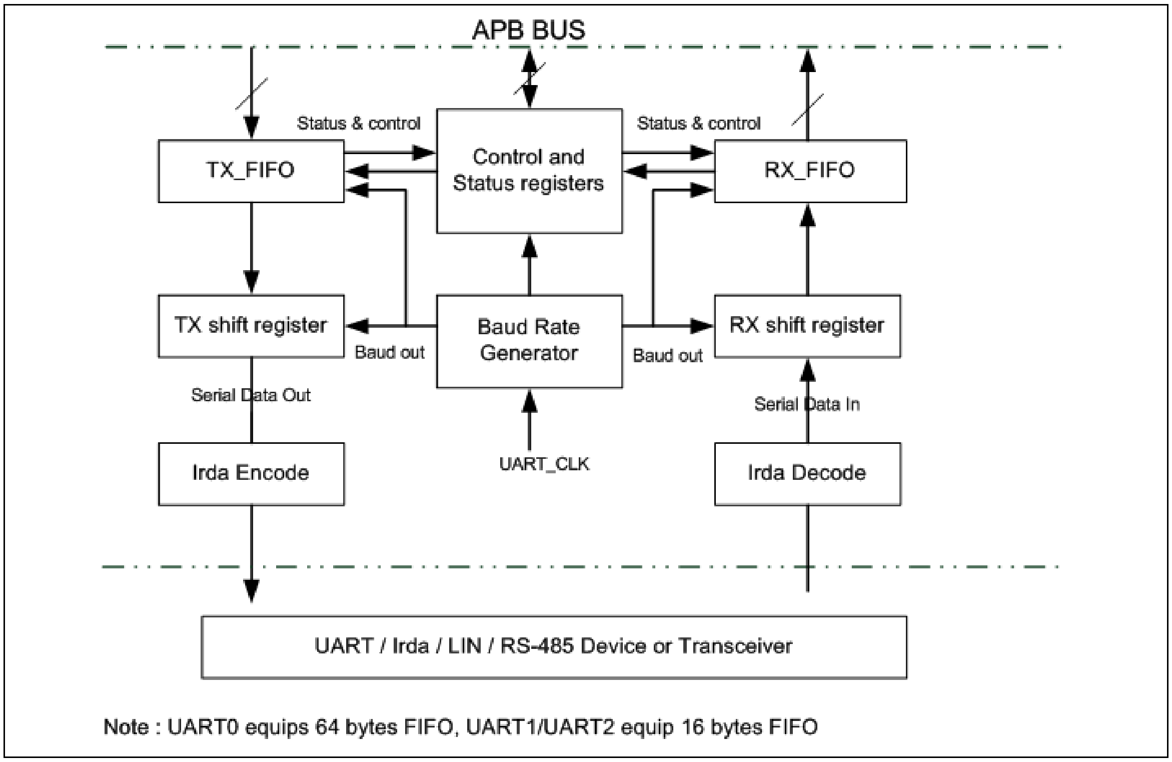


Figure 5-71 UART Block Diagram

TX_FIFO
The transmitter is buffered with a 64/16 byte FIFO to reduce the number of interrupts presented to the CPU.

RX_FIFO
The receiver is buffered with a 64/16 byte FIFO (plus three error bits per byte) to reduce the number of interrupts presented to the CPU.

TX shift Register
This block is the shifting the transmitting data out serially control block.

RX shift Register
This block is the shifting the receiving data in serially control block.

Modem Control Register
This register controls the interface to the MODEM or data set (or a peripheral device emulating a MODEM).

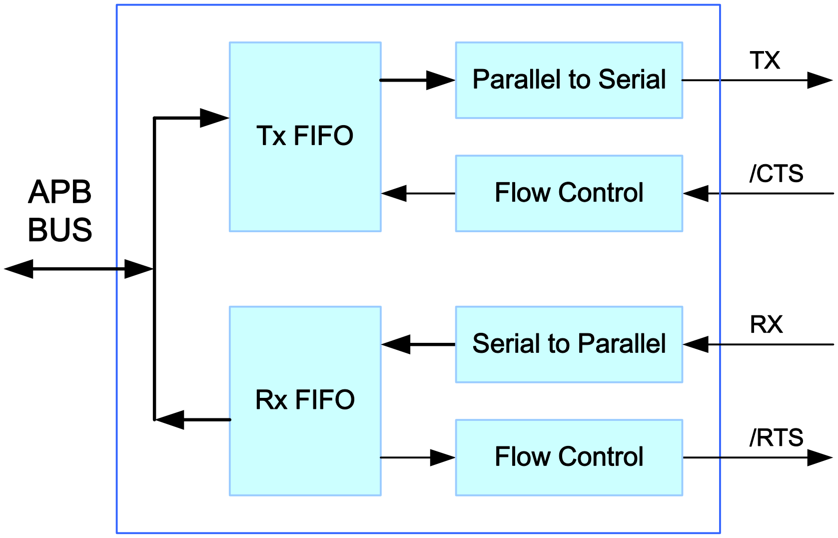
Baud Rate Generator
Divide the external clock by the divisor to get the desired baud rate clock. Refer to baud rate equation.

IrDA Encode
This block is IrDA encode control block.

IrDA Decode
This block is IrDA decode control block.

Control and Status Register
This field is register set that including the FIFO control registers (UA_FCR), FIFO status registers (UA_FSR), and line control register (UA_LCR) for transmitter and receiver. The time out control register (UA_TOR) identifies the condition of time out interrupt. This register set also includes the interrupt enable register (UA_IER) and interrupt status register (UA_ISR) to enable or disable the responding interrupt and to identify the occurrence of the responding interrupt. There are seven types of interrupts, transmitter FIFO empty interrupt(INT_THRE), receiver threshold level reaching interrupt (INT_RDA), line status interrupt (parity error or framing error or break interrupt) (INT_RLS) , time out interrupt (INT_TOUT), MODEM/Wake-up status interrupt (INT_MODEM), Buffer error interrupt (INT_BUF_ERR) and LIN receiver break field detected interrupt (INT_LIN_RX_BREAK).

The following diagram demonstrates the auto-flow control block diagram.



Note: Only supported in UART0 and UART1

Figure 5-72 Auto Flow Control Block Diagram

5.12.4 IrDA Mode

The UART supports IrDA SIR (Serial Infrared) Transmit Encoder and Receive Decoder, and IrDA mode is selected by setting the **IrDA_EN** bit in **UA_FUN_SEL** register.

When in IrDA mode, the UA_BAUD [DIV_X_EN] register must disable.

Baud Rate = Clock / (16 * BRD), where BRD is Baud Rate Divider in UA_BAUD register.

The following diagram demonstrates the IrDA control block diagram.

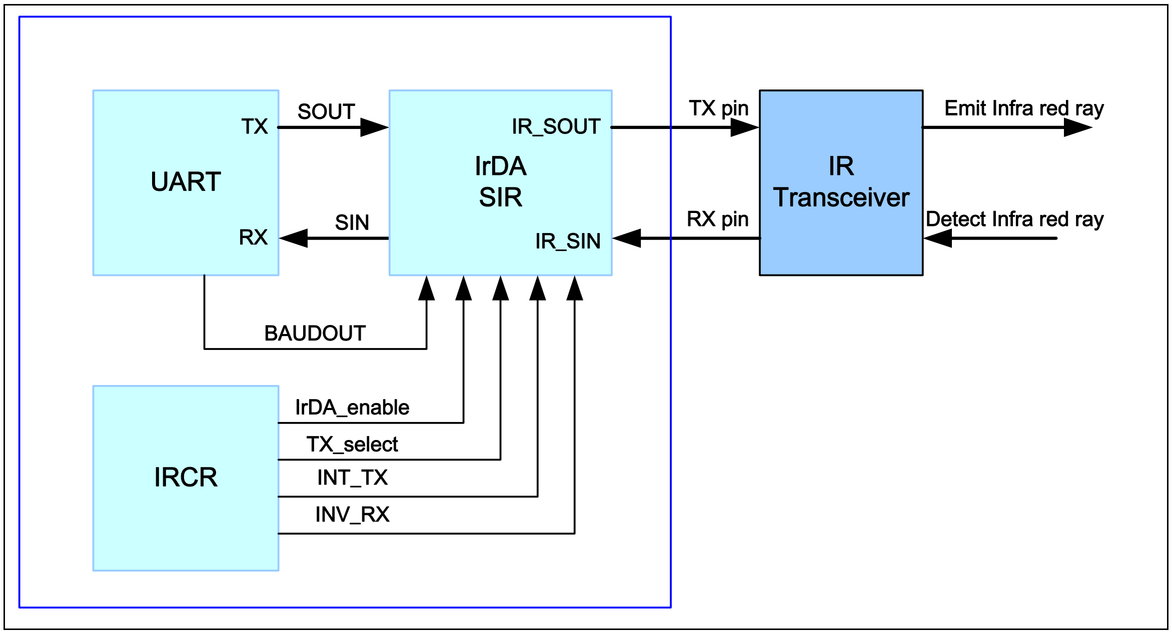


Figure 5-73 IrDA Block Diagram

5.12.4.1 IrDA SIR Transmit Encoder

The IrDA SIR Transmit Encoder modulate Non-Return-to Zero (NRZ) transmit bit stream output from UART. The IrDA SIR physical layer specifies use of Return-to-Zero, Inverted (RZI) modulation scheme which represent logic 0 as an infra light pulse. The modulated output pulse stream is transmitted to an external output driver and infrared Light Emitting Diode.

In normal mode, the transmitted pulse width is specified as 3/16 period of baud rate.

5.12.4.2 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. The decoder input is normally high in the idle state. (Because of this, IRCR bit 6 should be set as 1 by default)

A start bit is detected when the decoder input is LOW

5.12.4.3 IrDA SIR Operation

The IrDA SIR Encoder/decoder provides functionality which converts between UART data stream and half duplex serial SIR interface. The following diagram is IrDA encoder/decoder waveform:

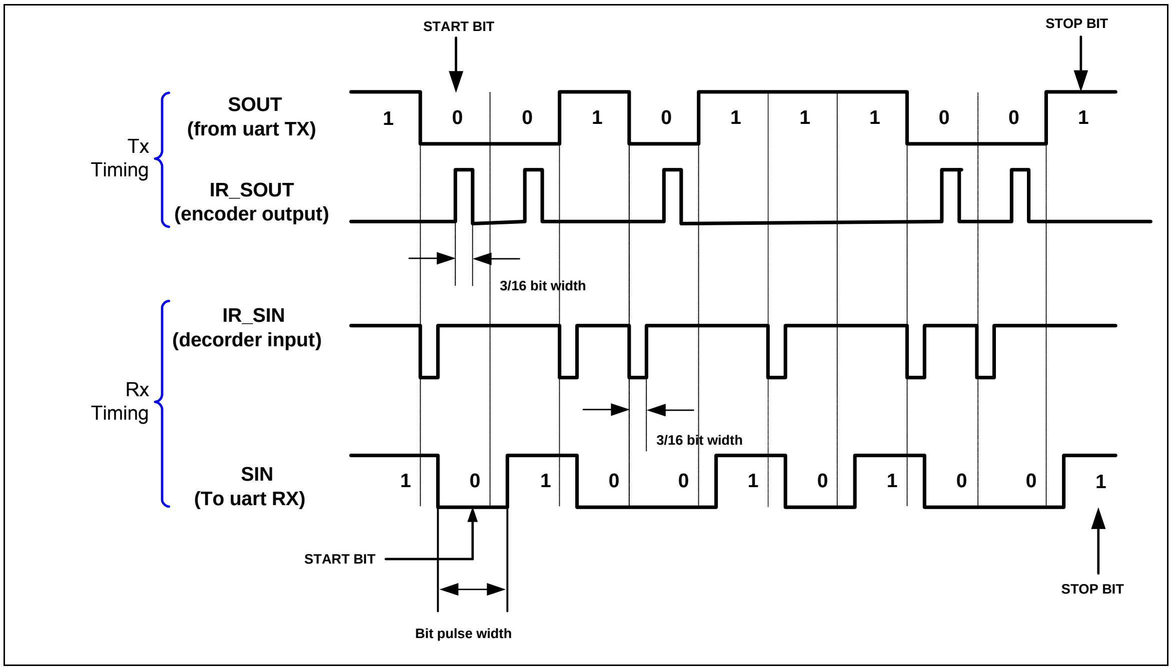


Figure 5-74 IrDA TX/RX Timing Diagram

5.12.5 LIN (Local Interconnection Network) mode

The UART supports LIN function, and LIN mode is selected by setting the **LIN_EN** bit in **UA_FUN_SEL** register. In LIN mode, each byte field is initialed by a start bit with value zero (dominant), followed by 8 data bits (LSB is first) and ended by 1 stop bit with value one (recessive) in accordance with the LIN standard. The following diagram is the structure of LIN function mode:

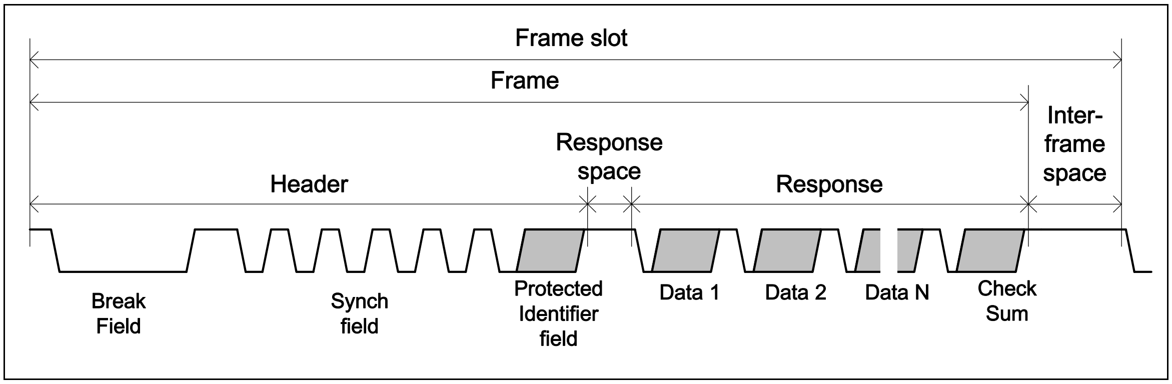


Figure 5-75 Structure of LIN Frame

The program flow of LIN Bus Transmit transfer (TX) is shown as following

1. Setting the LIN_EN bit in UA_FUN_SEL register to enable LIN Bus mode.
2. Fill UA_LIN_BKFL in UA_LIN_BCNT to choose break field length. (The break field length is UA_LIN_BKFL + 2).
3. Setting the LIN_TX_EN bit in UA_LIN_BCNT register to start transmission (When break filed operation is finished, LIN_TX_EN will be cleared automatically).
4. Fill 0x55 to UA_THR to request synch field transmission.
5. Request Identifier Field transmission by writing the protected identifier value in the UA_THR
6. When the STOP bit of the last byte THR has been sent to bus, hardware will set flag TE_FLAG in UA_FSR to 1.
7. Fill N bytes data and Checksum to UA_THR then repeat step 5 and 6 to transmit the data.

The program flow of LIN Bus Receiver transfer (RX) is show as following

1. Setting the LIN_EN bit in UA_FUN_SEL register to enable LIN Bus mode.
2. Setting the LIN_RX_EN bit in UA_LIN_BCNT register to enable LIN RX mode.
3. Waiting for the flag LIN_RX_BREAK_IF in UA_ISR to check RX received Break field or not.
4. Waiting for the flag RDA_IF in UA_ISR and read back the UR_RBR register.

5.12.6 RS-485 function mode

The UART support **RS-485 9-bit mode function**. The RS-485 mode is selected by setting the UA_FUN_SEL register to select RS-485 function. The RS-485 driver control is implemented using the RTS control signal from an asynchronous serial port to enable the RS-485 driver. In RS-485 mode, many characteristics of the RX and TX are same as UART.

When in RS-485 mode, the controller can configuration of it as an RS-485 addressable slave and the RS-485 master transmitter will identify an address character by setting the parity (9th bit) to 1. For data characters, the parity is set to 0. Software can use UA_LCR 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 support three operation mode that is 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 UA_RS-485_CSR register, and software can driving the transfer delay time between the last stop bit leaving the TX-FIFO and the de-assertion of by setting UA_TOR [DLY] register.

RS-485 Normal Multidrop Operation Mode (NMM)

In RS-485 Normal Multidrop operation mode, in first, software must decided the data which before the address byte be detected will be stored in RX-FIFO or not. If software want to ignore any data before address byte detected, the flow is set UART_FCR[RS485_RX_DIS] then enable UA_RS-485[RS485_NMM] and the receiver will ignore any data until an address byte is detected (bit9 =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 disable UART_FCR [RS485_RX_DIS] then enable UA_RS-485[RS485_NMM] and the receiver will received any data. If an address byte is detected (bit9 =1), it will generator an interrupt to CPU and software can decide whether enable or disable receiver to accept the following data byte by setting UA_RS-485_CSR [RX_DIS]. If the receiver is be enabled, all received byte data will be accepted and stored in the RX-FIFO, and if the receiver is disabled, all received byte data will be ignore until the next address byte be detected. If software disable receiver by setting UA_RS-485_CSR [RX_DIS] register, when a next address byte be detected, the controller will clear the UA_RS-485_CSR [RX_DIS] 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, the receiver will ignore any data until an address byte is detected (bit9 =1) and the address byte data match the UA_RS-485[ADDR_MATCH] 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 and address byte data not match the UA_RS-485[ADDR_MATCH] value.

RS-485 Auto Direction Mode (AUD)

Another option function of RS-485 controllers is **RS-485 auto direction control function**. The RS-485 driver control is implemented using the RTS control signal from an asynchronous serial port to enable the RS-485 driver. The RTS line is connected to the RS-485 driver enable such that setting the RTS line to high (logic 1) enables the RS-485 driver. Setting the RTS line to low (logic 0) puts the driver into the tri-state condition. User can setting LEV_RTS in UA_MCR register to change the RTS driving level.

Program Sequence example:

1. Program FUN_SEL in UA_FUN_SEL to select RS-485 function.
2. Program the RX_DIS bit in UA_FCR register to determine enable or disable RS-485 receiver
3. Program the RS-485_NMM or RS-485_AAD mode.
4. If the RS-485_AAD mode is selected, the ADDR_MATCH is programmed for auto address match value.
5. Determine auto direction control by programming RS-485_AUD.

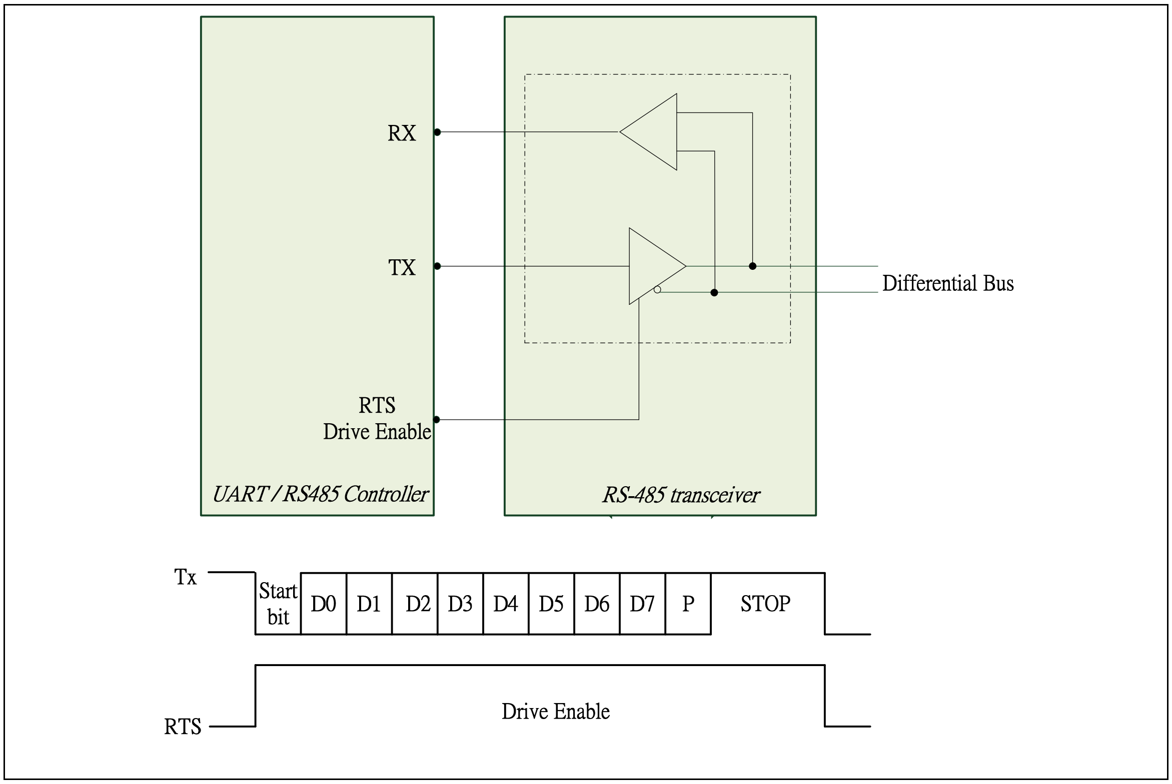


Figure 5-76 Structure of RS-485 Frame

5.12.7 Register Map

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

Register	Offset	R/W	Description	Reset Value
UART Base Address : **Channel0 : UART0_BA (High Speed) = 0x4005_0000** **Channel1 : UART1_BA (Normal Speed)= 0x4015_0000** **Channel2 : UART2_BA (Normal Speed)= 0x4015_4000**				
UA_RBR	UART0_BA+0x00	R	UART0 Receive Buffer Register	Undefined
	UART1_BA+0x00	R	UART1 Receive Buffer Register	Undefined
	UART2_BA+0x00	R	UART2 Receive Buffer Register	Undefined
UA_THR	UART0_BA+0x00	W	UART0 Transmit Holding Register	Undefined
	UART1_BA+0x00	W	UART1 Transmit Holding Register	Undefined
	UART2_BA+0x00	W	UART2 Transmit Holding Register	Undefined
UA_IER	UART0_BA+0x04	R/W	UART0 Interrupt Enable Register	0x0000_0000
	UART1_BA+0x04	R/W	UART1 Interrupt Enable Register	0x0000_0000
	UART2_BA+0x04	R/W	UART2 Interrupt Enable Register	0x0000_0000
UA_FCR	UART0_BA+0x08	R/W	UART0 FIFO Control Register	0x0000_0000
	UART1_BA+0x08	R/W	UART1 FIFO Control Register	0x0000_0000
	UART2_BA+0x08	R/W	UART2 FIFO Control Register	0x0000_0000
UA_LCR	UART0_BA+0x0C	R/W	UART0 Line Control Register	0x0000_0000
	UART1_BA+0x0C	R/W	UART1 Line Control Register	0x0000_0000
	UART2_BA+0x0C	R/W	UART2 Line Control Register	0x0000_0000
UA_MCR	UART0_BA+0x10	R/W	UART0 Modem Control Register	0x0000_0000
	UART1_BA+0x10	R/W	UART1 Modem Control Register	0x0000_0000
	UART2_BA+0x10	R/W	Reserved	0x0000_0000
UA_MSR	UART0_BA+0x14	R/W	UART0 Modem Status Register	0x0000_0000
	UART1_BA+0x14	R/W	UART1 Modem Status Register	0x0000_0000
	UART2_BA+0x14	R/W	Reserved	0x0000_0000
UA_FSR	UART0_BA+0x18	R/W	UART0 FIFO Status Register	0x1040_4000
	UART1_BA+0x18	R/W	UART1 FIFO Status Register	0x1040_4000

	UART2_BA+0x18	R/W	UART2 FIFO Status Register	0x1040_4000
UA_ISR	UART0_BA+0x1C	R/W	UART0 Interrupt Status Register	0x0000_0002
	UART1_BA+0x1C	R/W	UART1 Interrupt Status Register	0x0000_0002
	UART2_BA+0x1C	R/W	UART2 Interrupt Status Register	0x0000_0002
UA_TOR	UART0_BA+0x20	R/W	UART0 Time Out Register	0x0000_0000
	UART1_BA+0x20	R/W	UART1 Time Out Register	0x0000_0000
	UART2_BA+0x20	R/W	UART2 Time Out Register	0x0000_0000
UA_BAUD	UART0_BA+0x24	R/W	UART0 Baud Rate Divisor Register	0x0F00_0000
	UART1_BA+0x24	R/W	UART1 Baud Rate Divisor Register	0x0F00_0000
	UART2_BA+0x24	R/W	UART2 Baud Rate Divisor Register	0x0F00_0000
UA_IRCR	UART0_BA+0x28	R/W	UART0 IrDA Control Register	0x0000_0040
	UART1_BA+0x28	R/W	UART1 IrDA Control Register	0x0000_0040
	UART2_BA+0x28	R/W	UART2 IrDA Control Register	0x0000_0040
UA_ALT_CSR	UART0_BA+0x2C	R/W	UART0 Alternate Control/Status Register	0x0000_0000
	UART1_BA+0x2C	R/W	UART1 Alternate Control/Status Register	0x0000_0000
	UART2_BA+0x2C	R/W	UART2 Alternate Control/Status Register	0x0000_0000
UA_FUN_SEL	UART0_BA+0x30	R/W	UART0 Function Select Register	0x0000_0000
	UART1_BA+0x30	R/W	UART1 Function Select Register	0x0000_0000
	UART2_BA+0x30	R/W	UART2 Function Select Register	0x0000_0000

5.12.8 Register Description

Receive Buffer Register (UA_RBR)

Register	Offset	R/W	Description	Reset Value
UA_RBR	UART0_BA+0x00	R	UART0 Receive Buffer Register	Undefined
	UART1_BA+0x00	R	UART1 Receive Buffer Register	Undefined
	UART2_BA+0x00	R	UART2 Receive 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**
RBR							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved
[7:0]	**RBR**	**Receive Buffer Register (Read Only)** By reading this register, the UART will return an 8-bit data received from RX pin (LSB first).

Transmit Holding Register (UA_THR)

Register	Offset	R/W	Description	Reset Value
UA_THR	UART0_BA+0x00	W	UART0 Transmit Holding Register	Undefined
	UART1_BA+0x00	W	UART1 Transmit Holding Register	Undefined
	UART2_BA+0x00	W	UART2 Transmit Holding 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**
THR							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved
[7:0]	**THR**	**Transmit Holding Register** By writing to this register, the UART will send out an 8-bit data through the TX pin (LSB first).

Interrupt Enable Register (UA_IER)

Register	Offset	R/W	Description	Reset Value
UA_IER	UART0_BA+0x04	R/W	UART0 Interrupt Enable Register	0x0000_0000
	UART1_BA+0x04	R/W	UART1 Interrupt Enable Register	0x0000_0000
	UART2_BA+0x04	R/W	UART2 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**
DMA_RX_EN	**DMA_TX_EN**	**AUTO_CTS_EN**	**AUTO_RTS_EN**	**TIME_OUT_EN**	**Reserved**		**LIN_RX_BRK_IEN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**WAKE_EN**	**BUF_ERR_IEN**	**RTO_IEN**	**MODEM_IEN**	**RLS_IEN**	**THRE_IEN**	**RDA_IEN**

Bits	Descriptions	
[31:16]	**Reserved**	Reserved
[15]	**DMA_RX_EN**	**RX DMA Enable (not available in UART2 channel)** This bit can enable or disable RX DMA service. 1 = Enable RX DMA 0 = Disable RX DMA
[14]	**DMA_TX_EN**	**TX DMA Enable (not available in UART2 channel)** This bit can enable or disable TX DMA service. 1 = Enable TX DMA 0 = Disable TX DMA
[13]	**AUTO_CTS_EN**	**CTS Auto Flow Control Enable (not available in UART2 channel)** 1 = Enable CTS auto flow control 0 = Disable CTS auto flow control When CTS auto-flow is enabled, the UART will send data to external device when CTS input assert (UART will not send data to device until CTS is asserted).
[12]	**AUTO_RTS_EN**	**RTS Auto Flow Control Enable (not available in UART2 channel)** 1 = Enable RTS auto flow control 0 = Disable RTS auto flow control When RTS auto-flow is enabled, if the number of bytes in the RX FIFO equals the UA_FCR [RTS_TRI_LEV], the UART will de-assert RTS signal.

[11]	**TIME_OUT_EN**	**Time Out Counter Enable** 1 = Enable Time-out counter 0 = Disable Time-out counter
[10:9]	**Reserved**	Reserved
[8]	**LIN_RX_BRK_IEN**	**LIN RX Break Field Detected Interrupt Enable** 1 = Enable Lin bus RX break filed interrupt 0 = Mask off Lin bus RX break filed interrupt Note: This field is used for LIN function mode.
[7]	**Reserved**	Reserved
[6]	**WAKE_EN**	**UART Wake-up Function Enable (not available in UART2 channel)** 0 = Disable UART wake-up function 1 = Enable UART wake-up function, when the chip is in power down mode, an external CTS change will wake-up chip from power down mode.
[5]	**BUF_ERR_IEN**	**Buffer Error Interrupt Enable** 1 = Enable INT_BUF_ERR 0 = Mask off INT_BUF_ERR
[4]	**RTO_IEN**	**RX Time Out Interrupt Enable** 1 = Enable INT_TOUT 0 = Mask off INT_TOUT
[3]	**MODEM_IEN**	**Modem Status Interrupt Enable (not available in UART2 channel)** 1 = Enable INT_MODEM 0 = Mask off INT_MODEM
[2]	**RLS_IEN**	**Receive Line Status Interrupt Enable** 1 = Enable INT_RLS 0 = Mask off INT_RLS
[1]	**THRE_IEN**	**Transmit Holding Register Empty Interrupt Enable** 1 = Enable INT_THRE 0 = Mask off INT_THRE
[0]	**RDA_IEN**	**Receive Data Available Interrupt Enable**. 1 = Enable INT_RDA 0 = Mask off INT_RDA

FIFO Control Register (UA_FCR)

Register	Offset	R/W	Description	Reset Value
UA_FCR	UART0_BA+0x08	R/W	UART0 FIFO Control Register	0x0000_0000
	UART1_BA+0x08	R/W	UART1 FIFO Control Register	0x0000_0000
	UART2_BA+0x08	R/W	UART2 FIFO Control Register	0x0000_0000

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

Bits	Descriptions	
[31:20]	**Reserved**	Reserved
[19:16]	**RTS_TRI_LEV**	**RTS Trigger Level for Auto-flow Control Use (not available in UART2 channel)** RTS_TRI_LEV: 0000 = 01; 0001 = 04; 0010 = 08; 0011 = 14; 0100 = 30/14 (High Speed/Normal Speed); 0101 = 46/14 (High Speed/Normal Speed); 0110 = 62/14 (High Speed/Normal Speed); others = 62/14 (High Speed/Normal Speed) Note: This field is used for auto RTS flow control.
[15:9]	**Reserved**	Reserved
[8]	**RX_DIS**	**Receiver Disable register.** The receiver is disabled or not (set 1 is disable receiver) 1 = Disable Receiver 0 = Enable Receiver Note: This field is used for RS-485 Normal Multi-drop mode. It should be programmed before UA_ALT_CSR [RS-485_NMM] is programmed.

RTS_TRI_LEV [19:16]:

RTS_TRI_LEV	Trigger Level (Bytes)
0000	01
0001	04
0010	08
0011	14
0100	30/14 (High Speed/Normal Speed)
0101	46/14 (High Speed/Normal Speed)
0110	62/14 (High Speed/Normal Speed)
others	62/14 (High Speed/Normal Speed)

[7:4]	**RFITL**	**RX FIFO Interrupt (INT_RDA) Trigger Level** When the number of bytes in the receive FIFO equals the RFITL then the RDA_IF will be set (if UA_IER [RDA_IEN] is enable, an interrupt will generated). **RFITL** – **INTR_RDA Trigger Level (Bytes)** 0000 – 01 0001 – 04 0010 – 08 0011 – 14 0100 – 30/14 (High Speed/Normal Speed) 0101 – 46/14 (High Speed/Normal Speed) 0110 – 62/14 (High Speed/Normal Speed) others – 62/14 (High Speed/Normal Speed)
[3]	**Reserved**	Reserved
[2]	**TFR**	**TX Field Software Reset** When TX_RST is set, all the byte in the transmit FIFO and TX internal state machine are cleared. 1 = Writing 1 to this bit will reset the TX internal state machine and pointers. 0 = Writing 0 to this bit has no effect. Note: This bit will auto clear needs at least 3 UART engine clock cycles.
[1]	**RFR**	**RX Field Software Reset** When RX_RST is set, all the byte in the receiver FIFO and RX internal state machine are cleared. 1 = Writing 1 to this bit will reset the RX internal state machine and pointers. 0 = Writing 0 to this bit has no effect. Note: This bit will auto clear needs at least 3 UART engine clock cycles.
[0]	**Reserved**	Reserved

Line Control Register (UA_LCR)

Register	Offset	R/W	Description	Reset Value
UA_LCR	UART0_BA+0x0C	R/W	UART0 Line Control Register	0x0000_0000
	UART1_BA+0x0C	R/W	UART1 Line Control Register	0x0000_0000
	UART2_BA+0x0C	R/W	UART2 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	Descriptions	
[31:7]	**Reserved**	Reserved
[6]	**BCB**	**Break Control Bit** 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 and has no effect on the transmitter logic.
[5]	**SPE**	**Stick Parity Enable** 1 = If bit 3 and 4 are logic 1, the parity bit is transmitted and cheched as logic 0. If bit 3 si 1 and bit 4 is 0 then the parity bit is transmitted and checked as 1 0 = Stick parity disabled
[4]	**EPE**	**Even Parity Enable** 1 = Even number of logic 1's is transmitted and checked in each word 0 = Odd number of logic 1's is transmitted and checked in each word This bit has effect only when bit 3 (parity bit enable) is set.
[3]	**PBE**	**Parity Bit Enable** 1 = Parity bit is generated on each outgoing character and is checked on each incoming data. 0 = No parity bit.
[2]	**NSB**	**Number of "STOP bit"** 1= One and a half " STOP bit" is generated in the transmitted data when 5-bit word length is selected; 0= One " STOP bit" is generated in the transmitted data Two "STOP bit" is generated when 6-, 7- and 8-bit word length is selected.

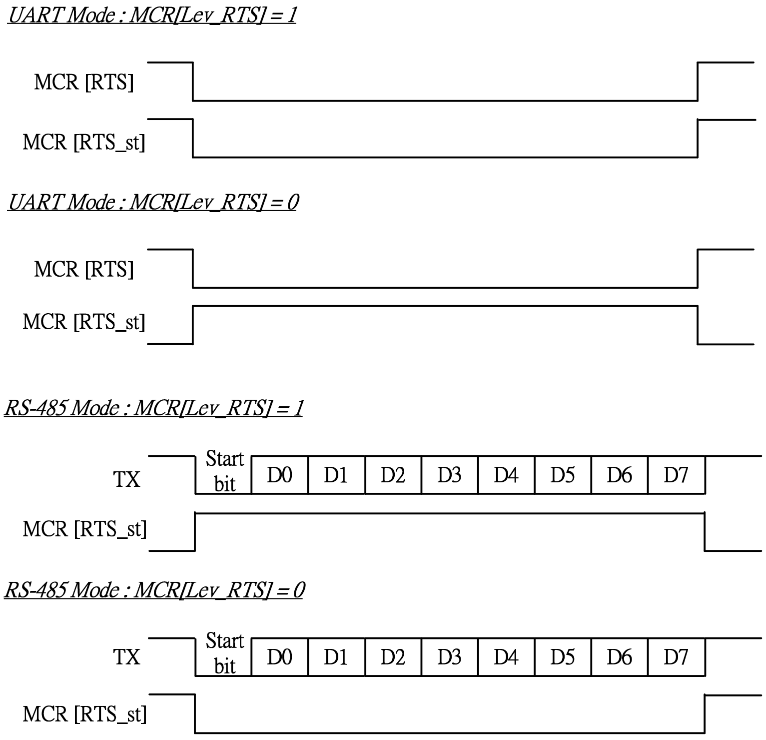
<table>
<tr>
<td>[1:0]</td>
<td>WLS</td>
<td>Word Length Select
<table>
<tr><th>WLS[1:0]</th><th>Character length</th></tr>
<tr><td>00</td><td>5-bit</td></tr>
<tr><td>01</td><td>6-bit</td></tr>
<tr><td>10</td><td>7-bit</td></tr>
<tr><td>11</td><td>8-bit</td></tr>
</table>
</td>
</tr>
</table>

MODEM Control Register (UA_MCR) (not available in UART2 channel)

Register	Offset	R/W	Description	Reset Value
UA_MCR	UART0_BA+0x10	R/W	UART0 Modem Control Register	0x0000_0000
	UART1_BA+0x10	R/W	UART1 Modem Control Register	0x0000_0000
	UART2_BA+0x10	R/W	Reserved	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		RTS_ST	Reserved			LEV_RTS	Reserved
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						RTS	Reserved

Bits	Descriptions	
[31:14]	**Reserved**	Reserved
[13]	**RTS_ST**	**RTS Pin State (Read Only) (not available in UART2 channel)** This bit is the output pin status of RTS.
[12:10]	**Reserved**	Reserved

[9]	**LEV_RTS**	**RTS Trigger Level (not available in UART2 channel)** This bit can change the RTS trigger level. 1= high level triggered 0= low level triggered 
[8:2]	**Reserved**	Reserved
[1]	**RTS**	**RTS (Request-To-Send) Signal (not available in UART2 channel)** 0 = Drive RTS pin to logic 1 (If the **LEV_RTS** set to low level triggered). 1 = Drive RTS pin to logic 0 (If the **LEV_RTS** set to low level triggered). 0 = Drive RTS pin to logic 0 (If the **LEV_RTS** set to high level triggered). 1 = Drive RTS pin to logic 1 (If the **LEV_RTS** set to high level triggered).
[0]	**Reserved**	Reserved

Modem Status Register (UA_MSR) (not available in UART2 channel)

Register	Offset	R/W	Description	Reset Value
UA_MSR	UART0_BA+0x14	R/W	UART0 Modem Status Register	0x0000_0000
	UART1_BA+0x14	R/W	UART1 Modem Status Register	0x0000_0000
	UART2_BA+0x14	R/W	Reserved	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							LEV_CTS
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			CTS_ST	Reserved			DCTSF

Bits	Descriptions	
[31:9]	**Reserved**	Reserved
[8]	**LEV_CTS**	**CTS Trigger Level (not available in UART2 channel)** This bit can change the CTS trigger level. 1= high level triggered 0= low level triggered
[7:5]	**Reserved**	Reserved
[4]	**CTS_ST**	**CTS Pin Status (Read Only) (not available in UART2 channel)** This bit is the pin status of CTS.
[3:1]	**Reserved**	Reserved
[0]	**DCTSF**	**Detect CTS State Change Flag (Read Only) (not available in UART2 channel)** This bit is set whenever CTS input has change state, and it will generate Modem interrupt to CPU when UA_IER [MODEM_IEN] is set to 1. Software can write 1 to clear this bit to zero

FIFO Status Register (UA_FSR)

Register	Offset	R/W	Description	Reset Value
UA_FSR	UART0_BA+0x18	R/W	UART0 FIFO Status Register	0x1040_4000
	UART1_BA+0x18	R/W	UART1 FIFO Status Register	0x1040_4000
	UART2_BA+0x18	R/W	UART2 FIFO Status Register	0x1040_4000

31	30	29	28	27	26	25	24
Reserved			TE_FLAG	Reserved			TX_OVER_IF
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TX_FULL	TX_EMPTY	TX_POINTER					
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RX_FULL	RX_EMPTY	RX_POINTER					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	BIF	FEF	PEF	RS485_ADD_DETF	Reserved		RX_OVER_IF

Bits	Descriptions	
[31:29]	**Reserved**	Reserved
[28]	**TE_FLAG**	**Transmitter Empty Flag (Read Only)** Bit is set by hardware when TX FIFO (UA_THR) is empty and the STOP bit of the last byte has been transmitted. Bit is cleared automatically when TX FIFO is not empty or the last byte transmission has not completed.
[27:25]	**Reserved**	Reserved
[24]	**TX_OVER_IF**	**TX Overflow Error Interrupt Flag (Read Only)** If TX FIFO (UA_THR) is full, an additional write to UA_THR will cause this bit to logic 1. Note: This bit is read only, but can be cleared by writing ‘1’ to it.
[23]	**TX_FULL**	**Transmitter FIFO Full (Read Only)** This bit indicates TX FIFO full or not. This bit is set when TX_POINTER is equal to 64/16(UART0/UART1), otherwise is cleared by hardware.
[22]	**TX_EMPTY**	**Transmitter FIFO Empty (Read Only)** This bit indicates TX FIFO empty or not. 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 THR (TX FIFO not empty).
[21:16]	**TX_POINTER**	**TX FIFO Pointer (Read Only)** This field indicates the TX FIFO Buffer Pointer. When CPU writes one byte into

		UA_THR, TX_POINTER increases one. When one byte of TX FIFO is transferred to Transmitter Shift Register, TX_POINTER decreases one.
[15]	**RX_FULL**	**Receiver FIFO Full (Read Only)** This bit initiates RX FIFO full or not. This bit is set when RX_POINTER is equal to 64/16(UART0/UART1), otherwise is cleared by hardware.
[14]	**RX_EMPTY**	**Receiver FIFO Empty (Read Only)** This bit initiate RX FIFO empty or not. 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]	**RX_POINTER**	**RX FIFO Pointer (Read Only)** This field indicates the RX FIFO Buffer Pointer. When UART receives one byte from external device, RX_POINTER increases one. When one byte of RX FIFO is read by CPU, RX_POINTER decreases one.
[7]	**Reserved**	Reserved
[6]	**BIF**	**Break Interrupt Flag (Read Only)** This bit is set to a 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) and is reset whenever the CPU writes 1 to this bit. 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 a logic 0), and is reset whenever the CPU writes 1 to this bit. 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”, and is reset whenever the CPU writes 1 to this bit. Note: This bit is read only, but can be cleared by writing ‘1’ to it.
[3]	**RS485_ADD_DETF**	**RS-485 Address Byte Detection Flag (Read Only)** This bit is set to logic 1 and set UA_ALT_CSR [RS-485_ADD_EN] whenever in RS-485 mode the receiver detect any address byte received address byte character (bit9 = ‘1’) bit", and it is reset whenever the CPU writes 1 to this bit. Note: This field is used for RS-485 function mode. Note: This bit is read only, but can be cleared by writing ‘1’ to it.
[2:1]	**Reserved**	Reserved
[0]	**RX_OVER_IF**	**RX Overflow Error IF (Read Only)** This bit is set when RX FIFO overflow. If the number of bytes of received data is greater than RX_FIFO (UA_RBR) size, 64/16 bytes of UART0/UART1, this bit will be set. Note: This bit is read only, but can be cleared by writing ‘1’ to it.

Interrupt Status Control Register (UA_ISR)

Register	Offset	R/W	Description	Reset Value
UA_ISR	UART0_BA+0x1C	R/W	UART0 Interrupt Status Register	0x0000_0002
	UART1_BA+0x1C	R/W	UART1 Interrupt Status Register	0x0000_0002
	UART2_BA+0x1C	R/W	UART2 Interrupt Status Register	0x0000_0002

31	30	29	28	27	26	25	24
HW_LIN_RX_BREAK_INT	Reserved	HW_BUF_ERR_INT	HW_TOUT_INT	HW_MODEM_INT	HW_RLS_INT	Reserved	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
HW_LIN_RX_BREAK_IF	Reserved	HW_BUF_ERR_IF	HW_TOUT_IF	HW_MODEM_IF	HW_RLS_IF	Reserved	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
LIN_RX_BREAK_INT	Reserved	BUF_ERR_INT	TOUT_INT	MODEM_INT	RLS_INT	THRE_INT	RDA_INT
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LIN_RX_BREAK_IF	Reserved	BUF_ERR_IF	TOUT_IF	MODEM_IF	RLS_IF	THRE_IF	RDA_IF

Bits	Descriptions	
[31]	**HW_LIN_RX_BREAK_INT**	**In DMA Mode, LIN Bus RX Break Field Detected Interrupt Indicator (Read Only)** This bit is set if LIN_RX_BRK_IEN and HW_LIN_RX_BREAK_IF are both set to 1. 1 = The LIN RX Break interrupt is generated in DMA mode 0 = No LIN RX Break interrupt is generated in DMA mode
[30]	**Reserved**	Reserved
[29]	**HW_BUF_ERR_INT**	**In DMA Mode, Buffer Error Interrupt Indicator (Read Only)** This bit is set if BUF_ERR_IEN and HW_BUF_ERR_IF are both set to 1. 1 = The buffer error interrupt is generated in DMA mode 0 = No buffer error interrupt is generated in DMA mode
[28]	**HW_TOUT_INT**	**In DMA Mode, Time Out Interrupt Indicator (Read Only)** This bit is set if TOUT_IEN and HW_TOUT_IF are both set to 1. 1 = The Tout interrupt is generated in DMA mode 0 = No Tout interrupt is generated in DMA mode
[27]	**HW_MODEM_INT**	**In DMA Mode, MODEM Status Interrupt Indicator (Read Only) (not available in UART2 channel)** This bit is set if MODEM_IEN and HW_MODEM_IF are both set to 1. 1 = The Modem interrupt is generated in DMA mode 0 = No Modem interrupt is generated in DMA mode

[26]	**HW_RLS_INT**	**In DMA Mode, Receive Line Status Interrupt Indicator (Read Only)** This bit is set if RLS_IEN and HW_RLS_IF are both set to 1. 1 = The RLS interrupt is generated in DMA mode 0 = No RLS interrupt is generated in DMA mode
[25:24]	**Reserved**	Reserved
[23]	**HW_LIN_RX_BREAK_IF**	**In DMA Mode, LIN Bus RX Break Field Detect Interrupt Flag (Read Only)** This bit is set when RX received LIN break field. If UA_IER [LIN_RX_BRK_IEN] is enabled the LIN RX break interrupt will be generated. Note: This bit is read only, but can be cleared by writing '1' to UA_ISR [7].
[22]	**Reserved**	Reserved
[21]	**HW_BUF_ERR_IF**	**In DMA Mode, Buffer Error Interrupt Flag (Read Only)** This bit is set when the TX or RX FIFO overflows (TX_OVER_IF or RX_OVER_IF is set). When BUF_ERR_IF is set, the transfer maybe is not correct. If UA_IER [BUF_ERR_IEN] is enabled, the buffer error interrupt will be generated. Note: This bit is cleared when both TX_OVER_IF and RX_OVER_IF are cleared.
[20]	**HW_TOUT_IF**	**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. If UA_IER [TOUT_IEN] is enabled, the Tout interrupt will be generated. Note: This bit is read only and user can read UA_RBR (RX is in active) to clear it.
[19]	**HW_MODEM_IF**	**In DMA Mode, MODEM Interrupt Flag (Read Only) (not available in UART2 channel)** This bit is set when the CTS pin has state change (DCTSF=1). If UA_IER [MODEM_IEN] is enabled, the Modem interrupt will be generated. Note: This bit is read only and reset to 0 when bit DCTSF is cleared by a write 1 on DCTSF.
[18]	**HW_RLS_IF**	**In DMA Mode, Receive Line Status Flag (Read Only)** This bit is set when the RX receive data have parity error, framing error or break error (at least one of 3 bits, BIF, FEF and PEF, is set). If UA_IER [RLS_IEN] is enabled, the RLS interrupt will be generated. Note: When in RS-485 function mode, this field include "receiver detect any address byte received address byte character (bit9 = '1') bit". Note: This bit is read only and reset to 0 when all bits of BIF, FEF and PEF are cleared.
[17:16]	**Reserved**	Reserved
[15]	**LIN_RX_BREAK_INT**	**LIN Bus RX Break Field Detected Interrupt Indicator (Read Only)** This bit is set if LIN_RX_BRK_IEN and LIN_RX_BREAK_IF are both set to 1. 1 = The LIN RX Break interrupt is generated 0 = No LIN RX Break interrupt is generated
[14]	**Reserved**	Reserved
[13]	**BUF_ERR_INT**	**Buffer Error Interrupt Indicator (Read Only)** This bit is set if BUF_ERR_IEN and BUF_ERR_IF are both set to 1. 1 = The buffer error interrupt is generated 0 = No buffer error interrupt is generated

[12]	**TOUT_INT**	**Time Out Interrupt Indicator (Read Only)** This bit is set if TOUT_IEN and TOUT_IF are both set to 1. 1 = The Tout interrupt is generated 0 = No Tout interrupt is generated
[11]	**MODEM_INT**	**MODEM Status Interrupt Indicator (Read Only). (not available in UART2 channel)** This bit is set if MODEM_IEN and MODEM_IF are both set to 1. 1 = The Modem interrupt is generated 0 = No Modem interrupt is generated
[10]	**RLS_INT**	**Receive Line Status Interrupt Indicator (Read Only).** This bit is set if RLS_IEN and RLS_IF are both set to 1. 1 = The RLS interrupt is generated 0 = No RLS interrupt is generated
[9]	**THRE_INT**	**Transmit Holding Register Empty Interrupt Indicator (Read Only).** This bit is set if THRE_IEN and THRE_IF are both set to 1. 1 = The THRE interrupt is generated 0 = No THRE interrupt is generated
[8]	**RDA_INT**	**Receive Data Available Interrupt Indicator (Read Only).** This bit is set if RDA_IEN and RDA_IF are both set to 1. 1 = The RDA interrupt is generated 0 = No RDA interrupt is generated
[7]	**LIN_RX_BREAK_IF**	**LIN Bus RX Break Field Detected Flag (Read Only)** This bit is set when RX received LIN Break Field. If UA_IER [LIN_RX_BRK_IEN] is enabled the LIN RX Break interrupt will be generated. Note: This bit is read only, but can be cleared by writing ‘1’ to it.
[6]	**Reserved**	Reserved
[5]	**BUF_ERR_IF**	**Buffer Error Interrupt Flag (Read Only)** This bit is set when the TX or RX FIFO overflows (TX_OVER_IF or RX_OVER_IF is set). When BUF_ERR_IF is set, the transfer maybe is not correct. If UA_IER [BUF_ERR_IEN] is enabled, the buffer error interrupt will be generated. Note: This bit is cleared when both TX_OVER_IF and RX_OVER_IF are cleared.
[4]	**TOUT_IF**	**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 UA_IER [TOUT_IEN] is enabled, the Tout interrupt will be generated. Note: This bit is read only and user can read UA_RBR (RX is in active) to clear it.
[3]	**MODEM_IF**	**MODEM Interrupt Flag (Read Only) (not available in UART2 channel)** This bit is set when the CTS pin has state change (DCTSF=1). If UA_IER [MODEM_IEN] is enabled, the Modem interrupt will be generated. Note: This bit is read only and reset to 0 when bit DCTSF is cleared by a write 1 on DCTSF.
[2]	**RLS_IF**	**Receive Line Interrupt Flag (Read Only).** This bit is set when the RX receive data have parity error, framing error or break error

		(at least one of 3 bits, BIF, FEF and PEF, is set). If UA_IER [RLS_IEN] is enabled, the RLS interrupt will be generated. Note: When in RS-485 function mode, this field include "receiver detect any address byte received address byte character (bit9 = '1') bit". Note: This bit is read only and reset to 0 when all bits of BIF, FEF and PEF are cleared.
[1]	**THRE_IF**	**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 UA_IER [THRE_IEN] is enabled, the THRE interrupt will be generated. Note: This bit is read only and it will be cleared when writing data into THR (TX FIFO not empty).
[0]	**RDA_IF**	**Receive Data Available Interrupt Flag (Read Only).** When the number of bytes in the RX FIFO equals the RFITL then the RDA_IF will be set. If UA_IER [RDA_IEN] is enabled, the RDA interrupt will be 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 Interrupt Source	Interrupt Enable Bit	Interrupt Indicator to Interrupt Controller	Interrupt Flag	Flag Cleared by
LIN RX Break Field Detected interrupt	LIN_RX_BRK_IEN	HW_LIN_RX_BREAK_INT	**HW_LIN_RX_BREAK_IF**	Write '1' to LIN_RX_BREAK_IF
Buffer Error Interrupt INT_BUF_ERR	BUF_ERR_IEN	HW_BUF_ERR_INT	**HW_BUF_ERR_IF** = (TX_OVER_IF or RX_OVER_IF)	Write '1' to TX_OVER_IF/ RX_OVER_IF
RX Timeout Interrupt INT_TOUT	RTO_IEN	HW_TOUT_INT	**HW_TOUT_IF**	Read UA_RBR
Modem Status Interrupt INT_MODEM	MODEM_IEN	HW_MODEM_INT	**HW_MODEM_IF** = (DCTSF)	Write '1' to DCTSF
Receive Line Status Interrupt INT_RLS	RLS_IEN	HW_RLS_INT	**HW_RLS_IF** = (BIF or FEF or PEF or RS-485_ADD_DETF)	Write '1' to BIF/FEF/PEF/ RS-485_ADD_DETF
Transmit Holding Register Empty Interrupt INT_THRE	THRE_IEN	HW_THRE_INT	**HW_THRE_IF**	Write UA_THR
Receive Data Available Interrupt INT_RDA	RDA_IEN	HW_RDA_INT	**HW_RDA_IF**	Read UA_RBR

Table 5-10 UART Interrupt Sources and Flags Table In DMA Mode

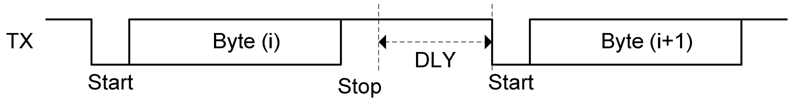
UART Interrupt Source	Interrupt Enable Bit	Interrupt Indicator to Interrupt Controller	Interrupt Flag	Flag Cleared by
LIN RX Break Field Detected interrupt	LIN_RX_BRK_IEN	LIN_RX_BREAK_INT	**LIN_RX_BREAK_IF**	Write '1' to LIN_RX_BREAK_IF
Buffer Error Interrupt INT_BUF_ERR	BUF_ERR_IEN	BUF_ERR_INT	**BUF_ERR_IF** = (TX_OVER_IF or RX_OVER_IF)	Write '1' to TX_OVER_IF/ RX_OVER_IF
RX Timeout Interrupt INT_TOUT	RTO_IEN	TOUT_INT	**TOUT_IF**	Read UA_RBR
Modem Status Interrupt INT_MODEM	MODEM_IEN	MODEM_INT	**MODEM_IF** = (DCTSF)	Write '1' to DCTSF
Receive Line Status Interrupt INT_RLS	RLS_IEN	RLS_INT	**RLS_IF** = (BIF or FEF or PEF)	Write '1' to BIF/FEF/PEF
Transmit Holding Register Empty Interrupt INT_THRE	THRE_IEN	THRE_INT	**THRE_IF**	Write UA_THR
Receive Data Available Interrupt INT_RDA	RDA_IEN	RDA_INT	**RDA_IF**	Read UA_RBR

Table 5-11 UART Interrupt Sources and Flags Table In Software Mode

Time out Register (UA_TOR)

Register	Offset	R/W	Description	Reset Value
UA_TOR	UART0_BA+0x20	R/W	UART0 Time Out Register	0x0000_0000
	UART1_BA+0x20	R/W	UART1 Time Out Register	0x0000_0000
	UART2_BA+0x20	R/W	UART2 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	Descriptions	
[31:16]	**Reserved**	Reserved
[15:8]	**DLY**	**TX Delay time value** This field is use to programming the transfer delay time between the last stop bit and next start bit. 
[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 (TOUT_CNT) is equal to that of time out interrupt comparator (TOIC), a receiver time out interrupt (INT_TOUT) is generated if UA_IER [RTO_IEN]. A new incoming data word or RX FIFO empty clears INT_TOUT. 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.

Baud Rate Divider Register (UA_BAUD)

Register	Offset	R/W	Description	Reset Value
UA_BAUD	UART0_BA+0x24	R/W	UART0 Baud Rate Divisor Register	0x0F00_0000
	UART1_BA+0x24	R/W	UART1 Baud Rate Divisor Register	0x0F00_0000
	UART2_BA+0x24	R/W	UART2 Baud Rate Divisor Register	0x0F00_0000

31	30	29	28	27	26	25	24
Reserved		DIV_X_EN	DIV_X_ONE	DIVIDER_X			
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	Descriptions	
[31:30]	**Reserved**	Reserved
[29]	**DIV_X_EN**	**Divider X Enable** The BRD = Baud Rate Divider, and the baud rate equation is Baud Rate = Clock / [M * (BRD + 2)]; The default value of M is 16. 1 = Enable divider X (the equation of M = X+1, but DIVIDER_X [27:24] must >= 8). 0 = Disable divider X (the equation of M = 16) Refer to the table below for more information. Note: When in IrDA mode, this bit must disable.
[28]	**DIV_X_ONE**	**Divider X equal 1** 1 = Divider M = 1 (the equation of M = 1, but BRD [15:0] must >= 3). 0 = Divider M = X (the equation of M = X+1, but DIVIDER_X[27:24] must >= 8) Refer to the Table 5-12 below for more information.
[27:24]	**DIVIDER_X**	**Divider X** The baud rate divider M = X+1.
[23:16]	**Reserved**	Reserved
[15:0]	**BRD**	**Baud Rate Divider** The field indicated the baud rate divider

Mode	DIV_X_EN	DIV_X_ONE	DIVIDER X	BRD	Baud rate equation
0	Disable	0	B	A	UART_CLK / [16 * (A+2)]
1	Enable	0	B	A	UART_CLK / [(B+1) * (A+2)] , B must >= 8
2	Enable	1	Don't care	A	UART_CLK / (A+2), A must >=3

Table 5-12 Baud rate equation table

IrDA Control Register (IRCR)

Register	Offset	R/W	Description	Reset Value
UA_IRCR	UART0_BA+0x28	R/W	UART0 IrDA Control Register	0x0000_0040
	UART1_BA+0x28	R/W	UART1 IrDA Control Register	0x0000_0040
	UART2_BA+0x28	R/W	UART2 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	**INV_RX**	**INV_TX**	**Reserved**			**TX_SELECT**	**Reserved**

Bits	Descriptions	
[31:7]	**Reserved**	Reserved
[6]	**INV_RX**	**INV_RX** 1= Inverse RX input signal 0= No inversion
[5]	**INV_TX**	**INV_TX** 1= Inverse TX output signal 0= No inversion
[4:2]	**Reserved**	Reserved
[1]	**TX_SELECT**	**TX_SELECT** 1= Enable IrDA transmitter 0= Enable IrDA receiver
[0]	**Reserved**	Reserved

Note: When in IrDA mode, the UA_BAUD [DIV_X_EN] register must disable (the baud equation must be Clock / 16 * (BRD)

UART Alternate Control/Status Register (UA_ALT_CSR)

Register	Offset	R/W	Description	Reset Value
UA_ALT_CSR	UART0_BA+0x2C	R/W	UART0 Alternate Control/Status Register	0x0000_0000
	UART1_BA+0x2C	R/W	UART1 Alternate Control/Status Register	0x0000_0000
	UART2_BA+0x2C	R/W	UART2 Alternate Control/Status Register	0x0000_0000

31	30	29	28	27	26	25	24
ADDR_MATCH							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RS485_ADD_EN	Reserved				RS485_AUD	RS485_AAD	RS485_NMM
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
LIN_TX_EN	LIN_RX_EN	Reserved		UA_LIN_BKFL			

Bits	Descriptions	
[31:24]	**ADDR_MATCH**	**Address match value register** This field contains the RS-485 address match values. Note: This field is used for RS-485 auto address detection mode.
[23:16]	**Reserved**	Reserved
[15]	**RS485_ADD_EN**	**RS-485 Address Detection Enable** This bit is use to enable RS-485 address detection mode. 1 = Enable address detection mode 0 = Disable address detection mode Note: This field is used for RS-485 any operation mode.
[14:11]	**Reserved**	Reserved
[10]	**RS485_AUD**	**RS-485 Auto Direction Mode (AUD)** 1 = Enable RS-485 Auto Direction Operation Mode (AUO) 0 = Disable RS-485 Auto Direction Operation Mode (AUO) Note: It can be active with RS-485_AAD or RS-485_NMM operation mode.
[9]	**RS485_AAD**	**RS-485 Auto Address Detection Operation Mode (AAD)** 1 = Enable RS-485 Auto Address Detection Operation Mode (AAD) 0 = Disable RS-485 Auto Address Detection Operation Mode (AAD) Note: It can't be active with RS-485_NMM operation mode.
[8]	**RS485_NMM**	**RS-485 Normal Multi-drop Operation Mode (NMM)**

		1 = Enable RS-485 Normal Multi-drop Operation Mode (NMM) 0 = Disable RS-485 Normal Multi-drop Operation Mode (NMM) Note: It can't be active with RS-485_AAD operation mode.
[7]	**LIN_TX_EN**	**LIN TX Break Mode Enable** 1 = Enable LIN TX Break Mode. 0 = Disable LIN TX Break Mode. Note: When TX break field transfer operation finished, this bit will be cleared automatically.
[6]	**LIN_RX_EN**	**LIN RX Enable** 1 = Enable LIN RX mode. 0 = Disable LIN RX mode.
[5:4]	**Reserved**	Reserved
[3:0]	**UA_LIN_BKFL**	**UART LIN Break Field Length** This field indicates a 4-bit LIN TX break field count. Note: This break field length is UA_LIN_BKFL + 2

UART Function Select Register (UA_FUN_SEL)

Register	Offset	R/W	Description	Reset Value
UA_FUN_SEL	UART0_BA+0x30	R/W	UART0 Function Select Register	0x0000_0000
	UART1_BA+0x30	R/W	UART1 Function Select Register	0x0000_0000
	UART2_BA+0x30	R/W	UART2 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						FUN_SEL	

Bits	Descriptions	
[31:2]	**Reserved**	Reserved
[1:0]	**FUN_SEL**	**Function Select Enable** 00 = UART Function 01 = Enable LIN Function 10 = Enable IrDA Function 11 = Enable RS-485 Function

5.13 Controller Area Network (CAN)

5.13.1 Overview

The C_CAN consists of the CAN Core, Message RAM, Message Handler, Control Registers and Module Interface (Refer Figure 5-77). 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.

5.13.2 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.
- Support wake-up function

5.13.3 Block Diagram

The C_CAN interfaces with the AMBA APB bus. Figure 5-77 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.

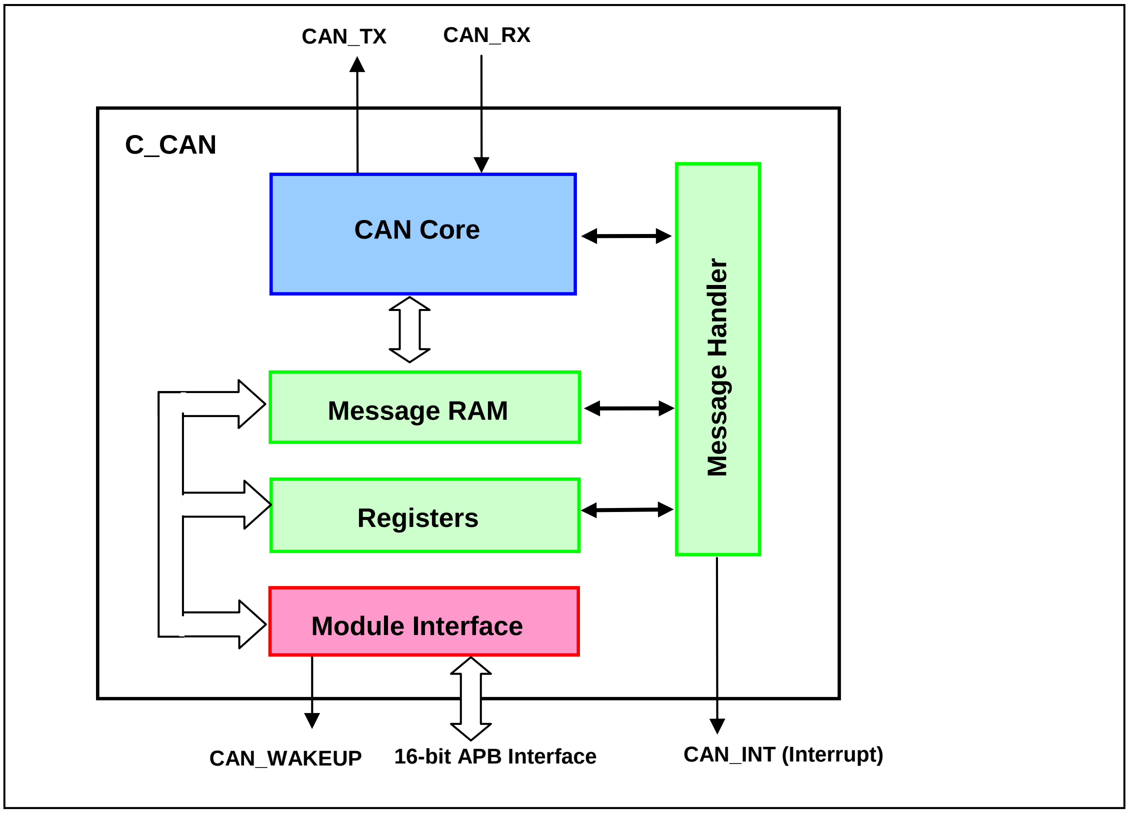


Figure 5-77 CAN Peripheral Block Diagram

5.13.4 Functional Description

5.13.4.1 Software Initialization

The software initialization is started by setting the **Init** bit in the CAN Control Register, 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 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** bits in the CAN Control Register are set.

Resetting the Init bit (by software only) finishes the software initialization. Later, the Bit Stream Processor (BSP) (see Section 5.13.6.10: 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** is set again.

5.13.4.2 CAN Message Transfer

Once the C_CAN is initialized and Init bit 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 and eight data bytes 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 with **NewDat** bit 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.

5.13.4.3 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 in the CAN Control Register to one. In this operation mode, the programmer has to consider the different behavior of bits TxRqst and NewDat in the Control Registers 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.

5.13.5 Test Mode

Test Mode is entered by setting the **Test** bit in the CAN Control Register. In Test Mode, bits **Tx1**, **Tx0**, **LBack**, **Silent** and **Basic** in the Test Register are writeable. Bit **Rx** 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.

5.13.5.1 Silent Mode

The CAN Core can be set in Silent Mode by programming the **Silent** bit in the Test Register 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. Figure 5-78 shows the connection of signals **CAN_TX** and **CAN_RX** to the CAN Core in Silent Mode.

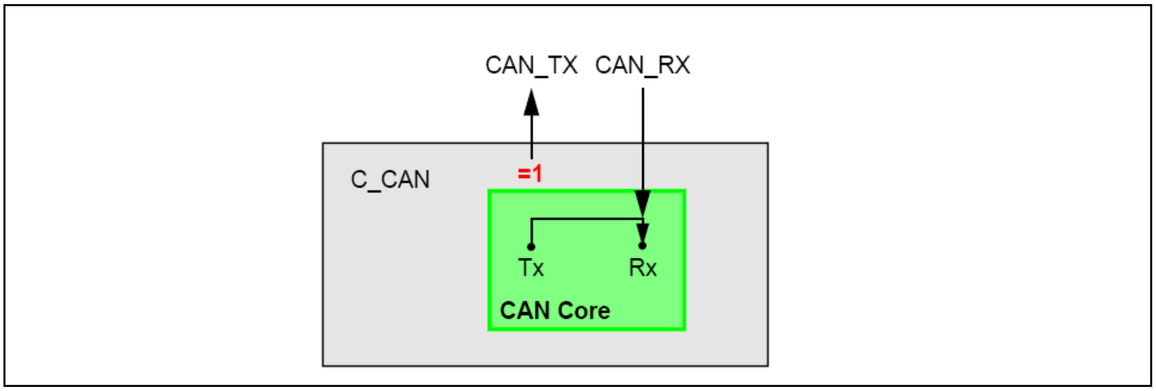


Figure 5-78 CAN Core in Silent Mode

5.13.5.2 Loop Back Mode

The CAN Core can be set in Loop Back Mode by programming the Test Register bit **LBack** to one. In Loop Back Mode, the CAN Core treats its own transmitted messages as received messages and stores them into a Receive Buffer (if they pass acceptance filtering).Figure 5-78 shows the connection of signals, **CAN_TX** and **CAN_RX**, to the CAN Core in Loop Back Mode.

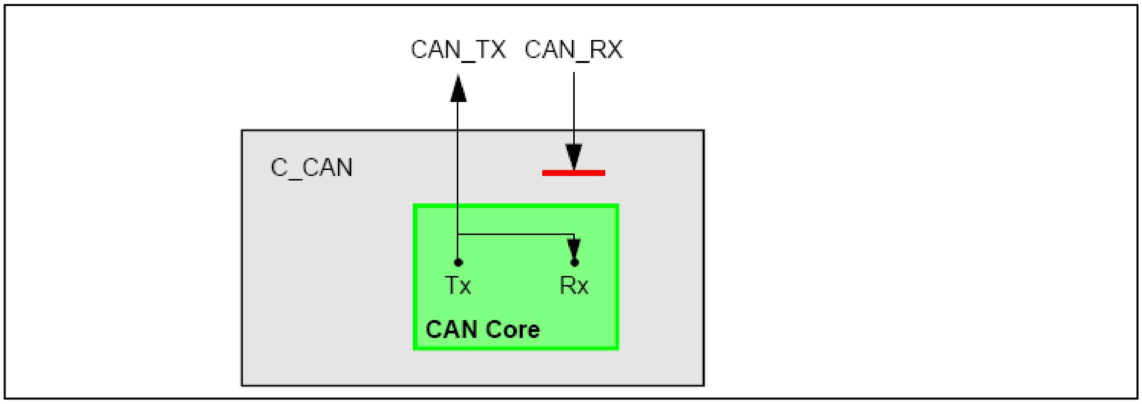


Figure 5-79 CAN Core in Loop Back Mode

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.

5.13.5.3 Loop Back Combined with Silent Mode

It is also possible to combine Loop Back Mode and Silent Mode by programming bits **LBack** and **Silent** 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. Figure 5-80 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.

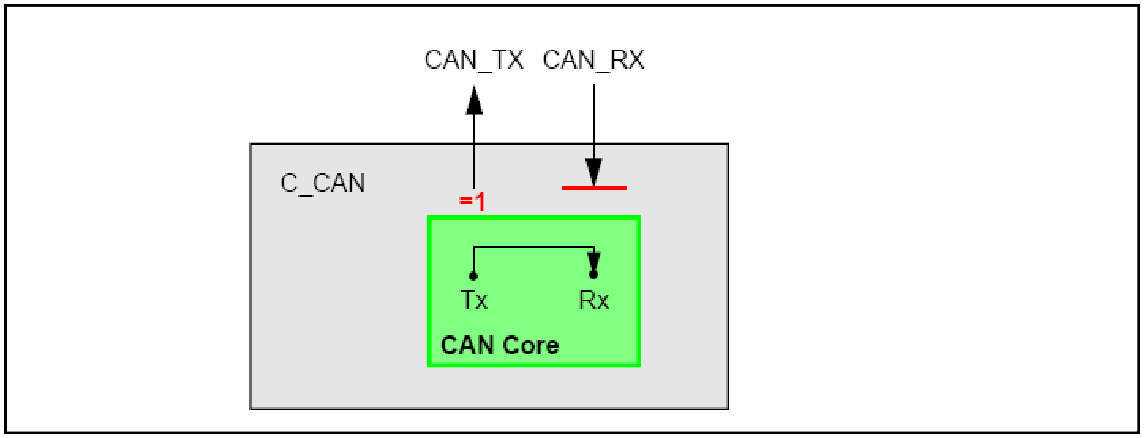


Figure 5-80 CAN Core in Loop Back Mode Combined with Silent Mode

5.13.5.4 Basic Mode

The CAN Core can be set in Basic Mode by programming the Test Register bit **Basic** 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 are requested by writing the Busy bit 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** and **MsgLst** bits in the IF2 Message Control Register retain their function, **DLC3-0** indicates the received DLC, and the other control bits are read as ‘0’.

5.13.5.5 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** and **Tx0** bits of the CAN Test Register.

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.

5.13.6 CAN Communications

5.13.6.1 Managing Message Objects

The configuration of the Message Objects in the Message RAM (with the exception of the bits he CAN Control Regist **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** = ‘0’) and the bit timing must be configured before the application software clears the Init bit in ter.

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 in the CAN Control Register 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.

5.13.6.2 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.

5.13.6.2.1 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 in the respective Command Request Register (CAN_IFn_CRR) to '1'. After the transfer has completed, the Busy bit is again cleared (see Figure 5-81).

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.

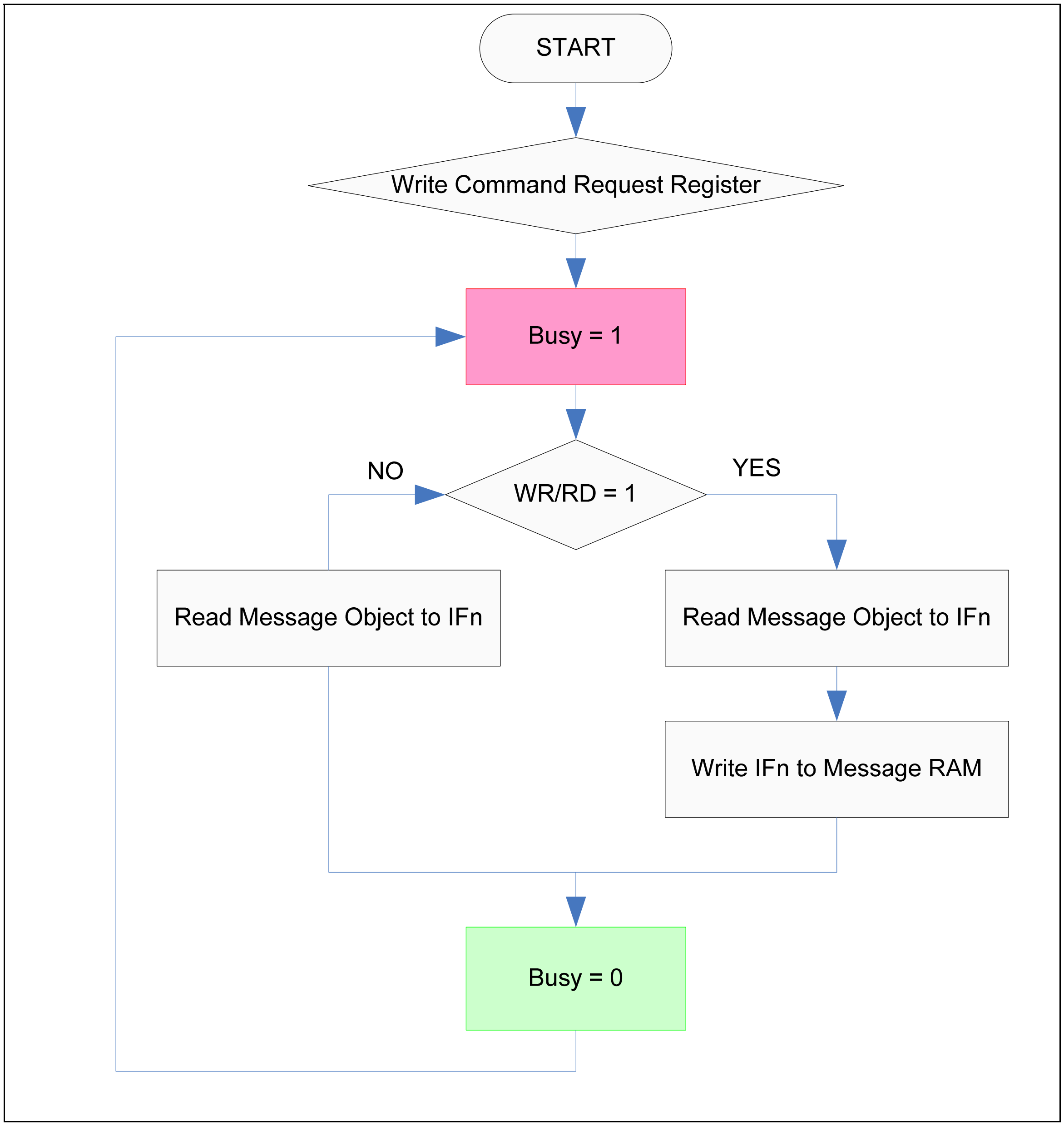


Figure 5-81 Data transfer between IFn Registers and Message

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.

5.13.6.2.2 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** bits in the Message Valid Register and **TxRqst** bits in the Transmission Request Register 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** 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_MCR) will be reset. If TxIE bit of the Message Control register (CAN_IFn_MCR) is set, **IntPnd** bit 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.

5.13.6.2.3 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**, **UMask**, **NewDat**, and **EoB**) 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 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** is set to indicate that the previous data (supposedly not seen by the software) is lost. If the **RxIE** bit is set, the **IntPnd** bit is set, causing the Interrupt Register to point to this Message Object.

The **TxRqst** bit 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 = ‘1’ (direction = transmit), **RmtEn** = ‘1’, **UMask** = ‘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’, **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’, **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 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.

5.13.6.2.4 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

5.13.6.3 Configuring a Transmit Object

Table 5-13 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 5-13 Initialization of a Transmit Object

Note: appl. = application software.

The Arbitration Register values (**ID28-0** and **Xtd** bit) 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 is set, the **IntPnd** bit will be set after a successful transmission of the Message Object.

If the **RmtEn** bit is set, a matching received Remote Frame will cause the **TxRqst** bit to be set; the Remote Frame will autonomously be answered by a Data Frame.

The Data Register values (**DLC3-0**, **Data0-7**) 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**=’1’) to allow groups of Remote Frames with similar identifiers to set the **TxRqst** bit. The Dir bit should not be masked.

5.13.6.4 Updating a Transmit Object

The software may update the data bytes of a Transmit Object any time through the IFn Interface registers, neither **MsgVal** nor **TxRqst** 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** 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.

5.13.6.5 Configuring a Receive Object

Table 5-14 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 5-14 Initialization of a Receive Object

The Arbitration Registers values **(ID28-0** and **Xtd** bit) 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 is set, the **IntPnd** bit will be set when a received Data Frame is accepted and stored in the Message Object.

The Data Length Code (DLC3-0) 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**=’1’) to allow groups of Data Frames with similar identifiers to be accepted. The Dir bit should not be masked in typical applications.

5.13.6.6 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** and **IntPnd** are cleared in the Message RAM (not in the Message Buffer).

If the Message Object uses masks for acceptance filtering, the arbitration bits shows 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** 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 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.

5.13.6.7 Configuring a FIFO Buffer

With the exception of the EoB bit, the configuration of Receive Objects belonging to a FIFO Buffer is the same as the configuration of a (single) Receive Object, see Section 5.13.6.5: 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** bits of the last Message Object of a FIFO Buffer is set to one, configuring it as the End of the Block.

5.13.6.8 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 of this Message Object is set. By setting **NewDat** while **EoB** 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 previous messages.

5.13.6.8.1 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** and **IntPnd** are reset to zero (**TxRqst**/**NewDat** = '1' and **ClrIntPnd** = '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.

Figure 5-82 shows how a set of Message Objects which are concatenated to a FIFO Buffer can be handled by the application software.

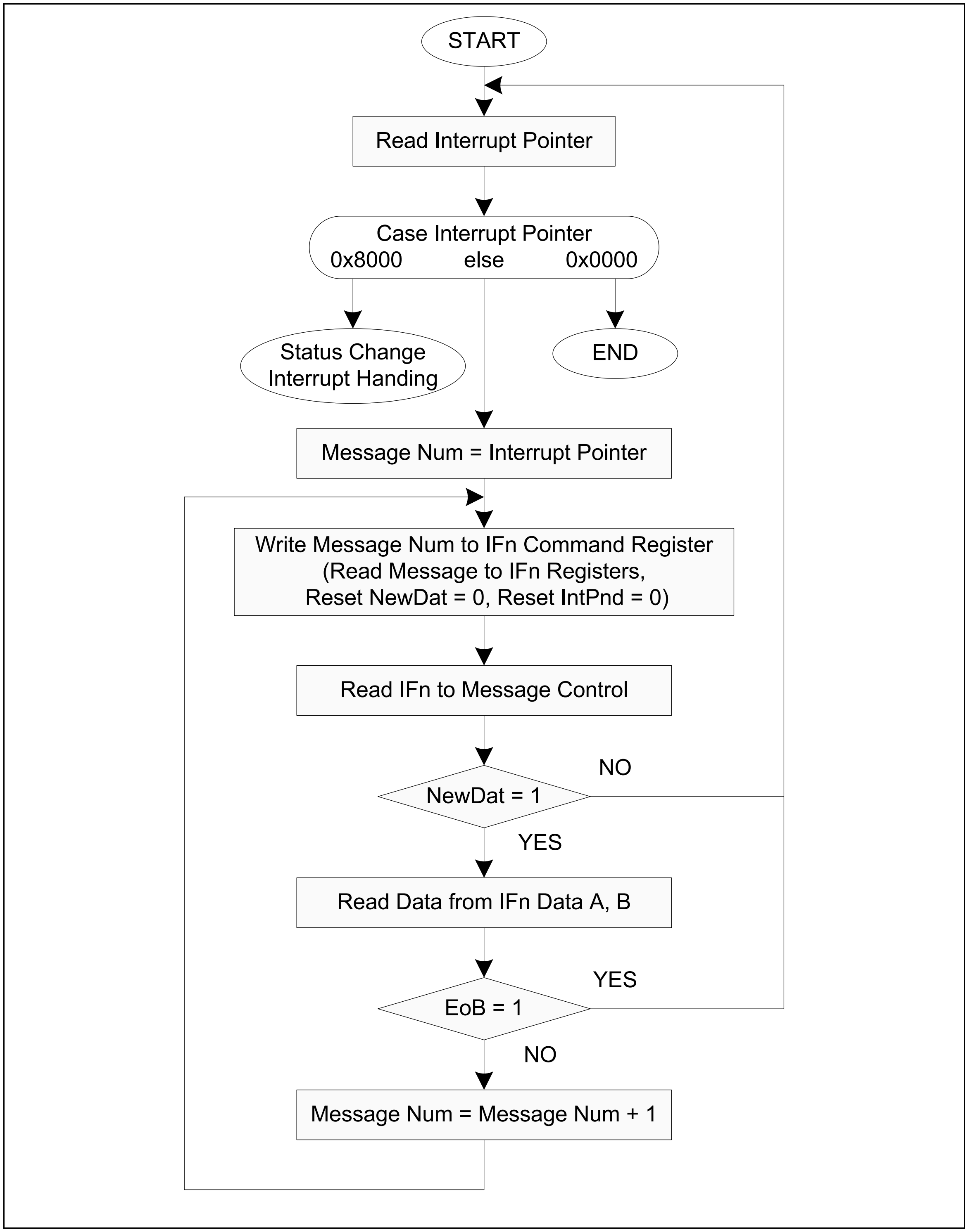


Figure 5-82 Application Software Handling of a FIFO Buffer

5.13.6.9 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 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 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**, **TxOk** and LEC, 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 and SIE in the CAN Control Register) 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** in the Command Mask Register). When **IntPnd** is cleared, the Interrupt Register will point to the next Message Object with a pending interrupt.

5.13.6.10 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.

5.13.6.10.1 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 (fosc) 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 (df), 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 Figure 5-83). 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 5-15). The length of the time quantum (tq), which is the basic time unit of the bit time, is defined by the CAN controller's APB clock fAPB and the BRP bit of the Bit Timing Register (CAN_BTR): tq = BRP / fAPB.

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 times 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.

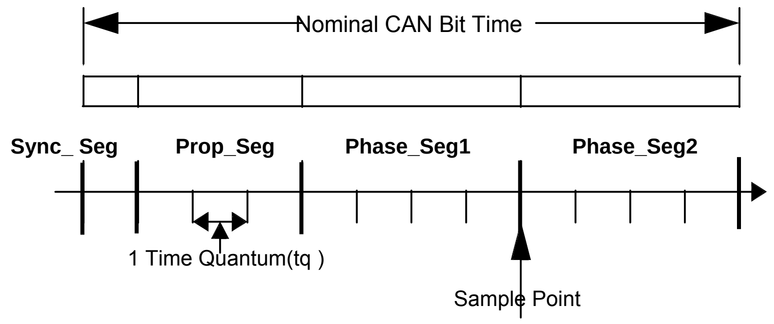


Figure 5-83 Bit Timing

Parameter	Range	Remark
BRP	[1 .. 32]	defines the length of the time quantum tq
Sync_Seg	1 tq	fixed length, synchronization of bus input to APB clock
Prop_Seg	[1.. 8] tq	compensates for the physical delay times
Phase_Seg1	[1..8] tq	may be lengthened temporarily by synchronization
Phase_Seg2	[1.. 8] tq	may be shortened temporarily by synchronization
SJW	[1 .. 4] tq	may not be longer than either Phase Buffer Segment
This table describes the minimum programmable ranges required by the CAN protocol		

Table 5-15 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 times and the oscillator's tolerance range have to be considered.

5.13.6.10.2 Propagation Time Segment

This part of the bit time is used to compensate physical delay times within the network. These delay times 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 Figure 5-84 shows the phase shift and propagation times between two CAN nodes.

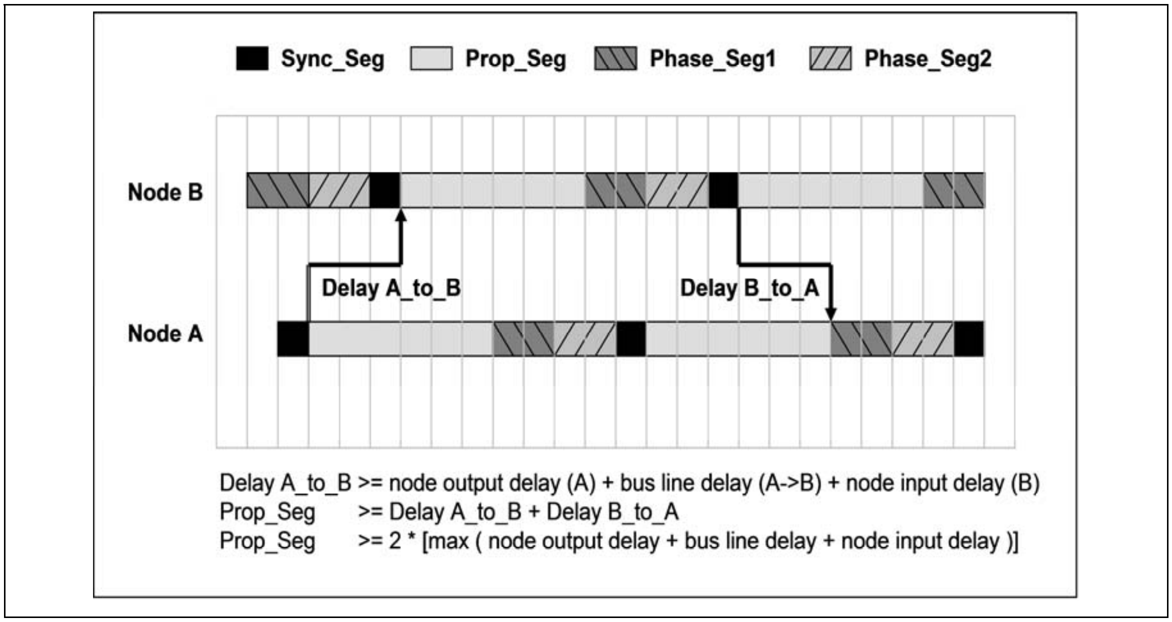


Figure 5-84 Propagation Time Segment

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 to 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 tq, requiring a longer Prop_Seg.

5.13.6.10.3 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 times, 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 Figure 5-85 show 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.

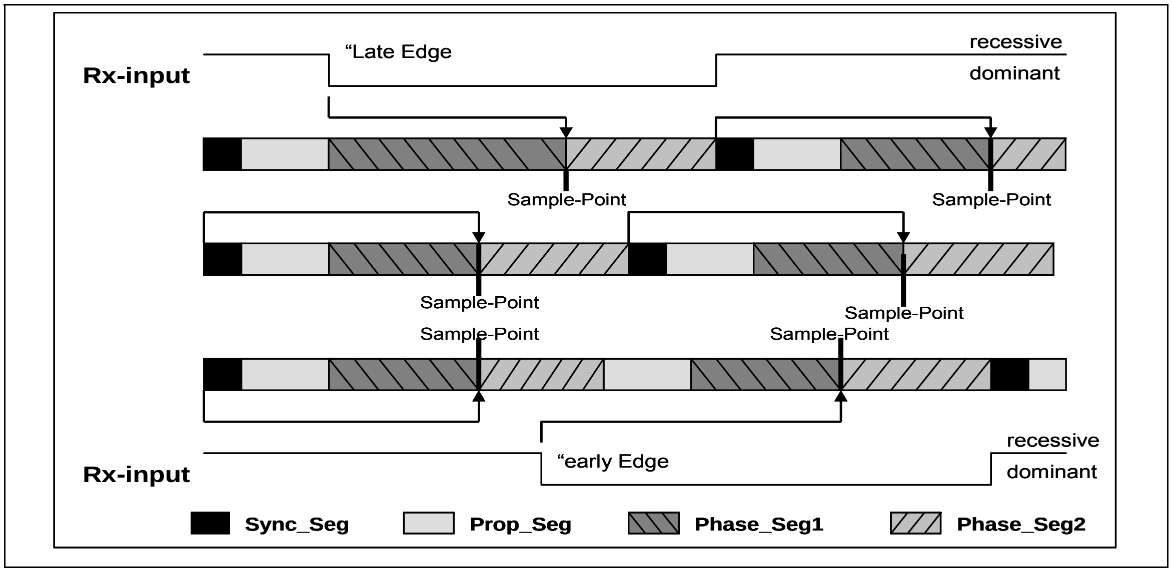


Figure 5-85 Synchronization on "late" and "early" Edges

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 Figure 5-86 show 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 Synchronisation 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.

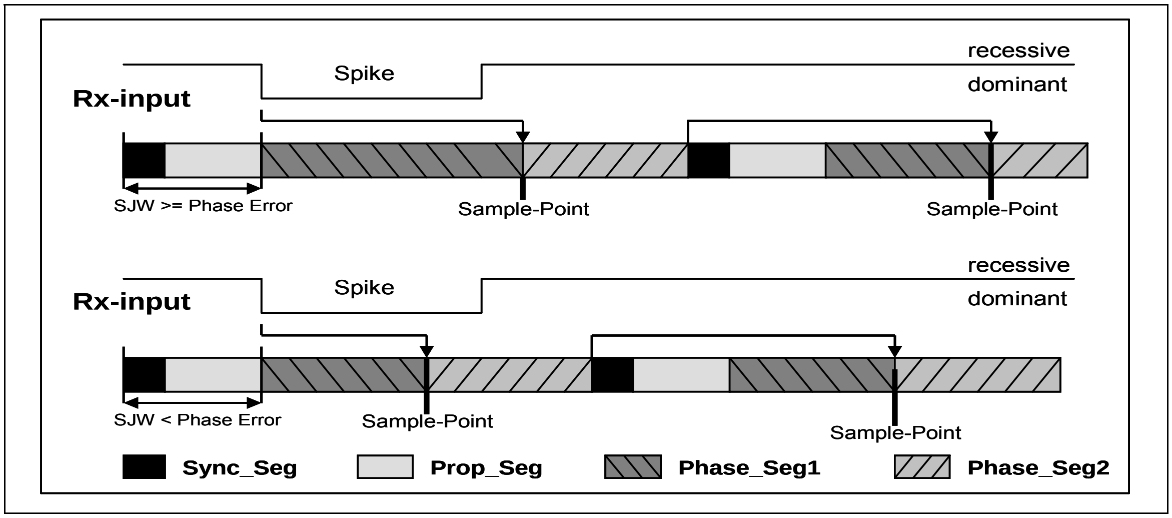


Figure 5-86 Filtering of Short Dominant Spikes

5.13.6.10.4 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:

$$(1 - df) \bullet f_{nom} \le f_{osc} \le (1 + df) \bullet f_{nom}$$

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):

$$I : df <= \frac{Min(Phase\_Seg1, Phase\_Seg2)}{2 \times (13 \times bit\_time - Phase\_Seg2)}$$

$$II : df <= \frac{SJW}{20 \times bit\_time}$$

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 = 8 µs) with a bus length of 40 m.

5.13.6.10.5 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) is combined with Phase_Seg2 (as TSEG2) in one byte, SJW and BRP 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.

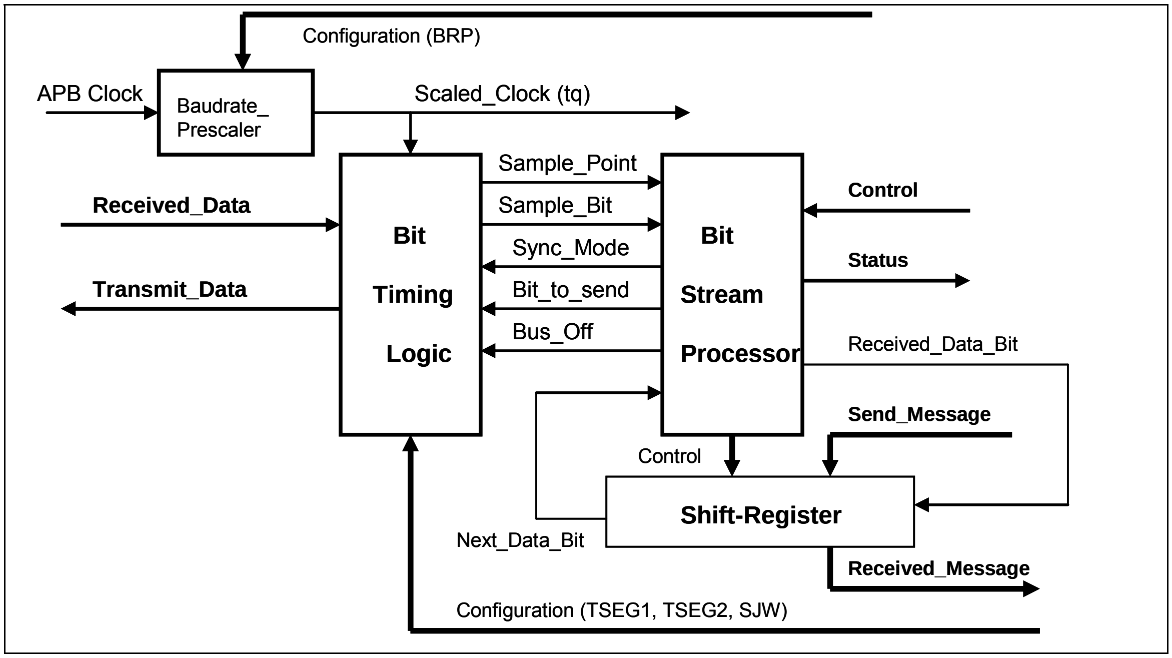


Figure 5-87 Structure of the CAN Core's CAN Protocol Controller

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. Its 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 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.

5.13.6.10.6 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 times 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 5.13.6.10.4: 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 times, 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 show 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 is 0, 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	$= 6 \bullet t_q$
t_{SJW}	100	ns	$= 1 \bullet t_q$
t_{TSeg1}	700	ns	$= t_{Prop} + t_{SJW}$
t_{TSeg2}	200	ns	= Information Processing Time + 1 • t_q
tSync-Seg	100	ns	= 1 • tq
bit time	1000	ns	$= t_{Sync\text{-}Seg} + t_{TSeg1} + t_{TSeg2}$
tolerance for APB_CLK	0.39	%	$= \frac{Min(PB1, PB2)}{2 \times 13 \times (bit\ \ time - PB2))}$ $= \frac{0.1us}{2 \times (13 \times (1us - 0.2us))}$

In this example ,the concatenated bit time parameters are $(2\text{-}1)_3\&(7\text{-}1)_4\&(1\text{-}1)_2\&(1\text{-}1)_6$, the Bit Timing Register is programmed to= 0x1600.

Example for Bit Timing at Low Baudrate

In this example, the frequency of APB_CLK is 2 MHz, BRP is 1, the bit rate is 100 KBit/s.

t_q	1	μs = 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	μs = 1 •t_q
t_{SJW}	4	μs= 4 • t_q
t_{TSeg1}	5	μs = t_{Prop} + t_{SJW}
t_{TSeg2}	4	μs = Information Processing Time + 3 • t_q
$t_{Sync-Seg}$	1	μs = 1 • t_q
bit time	10	μs = $t_{Sync-Seg}$ + 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 $(4\text{-}1)_3$&$(5\text{-}1)_4$&$(4\text{-}1)_2$&$(2\text{-}1)_6$, the Bit Timing Register is programmed to= 0x34C1.

5.13.7 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.

5.13.8 Register Map

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

Register	**Offset**	**R/W**	**Description**	**Reset Value**
CAN0_BA = 0x4018_0000				
CAN_CON	CAN0_BA+0x00	R/W	Control Register	0x0000_0001
CAN_STATUS	CAN0_BA+0x04	R/W	Status Register	0x0000_0000
CAN_ERR	CAN0_BA+0x08	R	Error Counter	0x0000_0000
CAN_BTIME	CAN0_BA+0x0C	R/W	Bit Timing Register	0x0000_2301
CAN_IIDR	CAN0_BA+0x10	R	Interrupt Identifier Register	0x0000_0000
CAN_TEST	CAN0_BA+0x14	R/W	Test Register	*(1)
CAN_BRPE	CAN0_BA+0x18	R/W	BRP Extension Register	0x0000_0000
CAN_IF1_CREQ **CAN_IF2_CREQ**	CAN0_BA+0x20 CAN0_BA+0x80	R/W	IFn (*2) Command Request Registers	0x0000_0001
CAN_IF1_CMASK **CAN_IF2_CMASK**	CAN0_BA+0x24 CAN0_BA+0x84	R/W	IFn Command Mask Registers	0x0000_0000
CAN_IF1_MASK1 **CAN_IF2_MASK1**	CAN0_BA+0x28 CAN0_BA+0x88	R/W	IFn Mask 1 Register	0x0000_FFFF
CAN_IF1_MASK2 **CAN_IF2_MASK2**	CAN0_BA+0x2C CAN0_BA+0x8C	R/W	IFn Mask 2 Register	0x0000_FFFF
CAN_IF1_ARB1 **CAN_IF2_ARB1**	CAN0_BA+0x30 CAN0_BA+0x90	R/W	IFn Arbitration 1 Register	0x0000_0000
CAN_IF1_ARB2 **CAN_IF2_ARB2**	CAN0_BA+0x34 CAN0_BA+0x94	R/W	IFn Arbitration 2 Register	0x0000_0000
CAN_IF1_MCON **CAN_IF2_MCON**	CAN0_BA+0x38 CAN0_BA+0x98	R/W	IFn Message Control Registers	0x0000_0000
CAN_IF1_DAT_An/ **CAN_IF1_DAT_Bn/** **CAN_IF2_DAT_An/** **CAN_IF2_DAT_Bn/**	CAN0_BA+0x3C~40 CAN0_BA+0x44~48 CAN0_BA+0x9C~A0 CAN0_BA+0xA4~A8	R/W	IFn Data An (*3) and Data Bn (*3) Registers eg: CAN_IF1_DAT_A1 = CAN_BA+0x3Ch CAN_IF1_DAT_A2 = CAN_BA+0x40h	0x0000_0000

CAN_TXREQ1 **CAN_TXREQ2**	CAN0_BA+0x100 CAN0_BA+0x104	R	Transmission Request Registers 1 & 2	0x0000_0000
CAN_NDAT1 **CAN_NDAT2**	CAN0_BA+0x120 CAN0_BA+0x124	R	New Data Registers 1 & 2	0x0000_0000
CAN_IPND1 **CAN_IPND2**	CAN0_BA+0x140 CAN0_BA+0x144	R	Interrupt Pending Registers 1 & 2	0x0000_0000
CAN_MVLD1 **CAN_MVLD2**	CAN0_BA+0x160 CAN0_BA+0x164	R	Message Valid Registers 1 & 2	0x0000_0000
CAN_WU_EN	CAN0_BA+0x168	R/W	Wake Up Function Enable	0x0000_0000
CAN_WU_STATUS	CAN0_BA+0x16C	R/W	Wake Up Function Status	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.

5.13.9 CAN Interface Reset State

After the hardware reset, the C_CAN registers hold the reset values given in the register descriptions in CAN register map .

Additionally the *busoff* 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 to ‘0’.

The data stored in the Message RAM is not affected by a hardware reset. After powering 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															WAKU _STS
170h	CAN_RAM_CEN	Reserved															RAM_ CEN
Others	Reserved	Reserved															

Table 5-16 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

CAN Control Register (CAN_CON)

Register	Offset	R/W	Description	Reset Value
CAN_CON	CAN0_BA+0x00	R/W	CAN 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**
Test	**CCE**	**DAR**	**Reserved**	**EIE**	**SIE**	**IE**	**Init**

Bits	Descriptions	
[31:8]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as '0' and must always be written with '0'
[7]	**Test**	**Test Mode Enable** 1 = Test Mode. 0 = Normal Operation.
[6]	**CCE**	**Configuration Change Enable** 1 = Write access to the Bit Timing Register (CAN_BTIME & CAN_BRP) allowed. (while Init bit =1). 0 = No write access to the Bit Timing Register.
[5]	**DAR**	**Disable Automatic Re-transmission** 1 = Automatic Retransmission disabled. 0 = Automatic Retransmission of disturbed messages enabled.
[4]	**Reserved**	**Reserved** This is a reserved bit. This bit is always read as '0' and must always be written with '0'.
[3]	**EIE**	**Error Interrupt Enable** 1 = Enabled - A change in the bits BOff or EWarn in the Status Register will generate an interrupt. 0 = Disabled - No Error Status Interrupt will be generated.
[2]	**SIE**	**Status Change Interrupt Enable** 1 = Enabled - An interrupt will be generated when a message transfer is successfully completed or a CAN bus error is detected.

		0 = Disabled - No Status Change Interrupt will be generated.
[1]	**IE**	**Module Interrupt Enable** 1 = Enabled. 0 = Disabled.
[0]	**Init**	**Init Initialization** 1 = Initialization is started. 0 = Normal Operation.

Note: The busoff recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or resetting the Init bit. If the device goes in the busoff 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 busoff 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 busoff recovery sequence.

CAN Status Register (CAN_STATUS)

Register	Offset	R/W	Description	Reset Value
CAN_STATUS	CAN0_BA+0x04	R/W	CAN 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	Descriptions	
[31:8]	**Reserved**	**Reserved** This is a reserved bit. This bit is always read as ‘0’ and must always be written with ‘0’.
[7]	**BOff**	**Busoff Status** (Read Only) 1 = The CAN module is in busoff state. 0 = The CAN module is not in busoff state.
[6]	**EWarn**	**Error Warning Status** (Read Only) 1 = At least one of the error counters in the EML has reached the error warning limit of 96. 0 = Both error counters are below the error warning limit of 96.
[5]	**EPass**	**Error Passive** (Read Only) 1 = The CAN Core is in the error passive state as defined in the CAN Specification. 0 = The CAN Core is error active.
[4]	**RxOK**	**Received a Message Successfully** 1 = A message has been successfully received since this bit was last reset by the CPU (independent of the result of acceptance filtering). 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.
[3]	**TxOK**	**Transmitted a Message Successfully** 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. 0 = Since this bit was reset by the CPU, no message has been successfully transmitted. This bit is never reset by the CAN Core.
[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. Table 5-17 describes the error codes.

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 busoff 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 busoff 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 5-17 Error Codes

Status Interrupts

A Status Interrupt is generated by bits **BOff** and **EWarn** (Error Interrupt) or by **RxOk**, **TxOk**, and **LEC** (Status Change Interrupt) assumed that the corresponding enable bits in the CAN Control Register are set. A change of bit **EPass** 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	CAN0_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[6:0]**						
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TEC[7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved** This is a reserved bit. This bit is always read as ‘0’ and must always be written with ‘0’.
[15]	**RP**	**Receive Error Passive** 1 = The Receive Error Counter has reached the error passive level as defined in the CAN Specification. 0 = The Receive Error Counter is below the error passive level.
[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	CAN0_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	Descriptions	
[31:15]	**Reserved**	**Reserved** This is a reserved bit. This bit is always read as '0' and must always be written with '0'.
[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 and Init in the CAN Control Register are set.

Interrupt Identify Register (CAN_IIDR)

Register	Offset	R/W	Description	Reset Value
CAN_IIDR	CAN0_BA+0x10	R	Interrupt Identifier 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**
IntId[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IntId[7:0]							

Bits	Descriptions	
[15:0]	**IntId**	**Interrupt Identifier** (Indicates the source of the interrupt. Ref. Table 5-18) 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 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. 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 5-18 Source of Interrupts

Test Register (CAN_TEST)

Register	Offset	R/W	Description	Reset Value
CAN_TEST	CAN0_BA+0x14	R/W	Test Register	0x0000_00x0

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[1:0]**		**LBack**	**Silent**	**Basic**	**Res**	

Bits	Descriptions	
[31:8]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.
[7]	**Rx**	**Monitors the actual value of CAN_RX Pin** (Read Only) 1 = The CAN bus is recessive (CAN_RX = ‘1’). 0 = The CAN bus is dominant (CAN_RX = ‘0’).
[6:5]	**Tx[1:0]**	**Tx[1:0]: Control of CAN_TX pin** 00 = Reset value, CAN_TX 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** 1 = Loop Back Mode is enabled. 0 = Loop Back Mode is disabled.
[3]	**Silent**	**Silent Mode** 1 = The module is in Silent Mode. 0 = Normal operation.
[2]	**Basic**	**Basic Mode** 1= IF1 Registers used as Tx Buffer, IF2 Registers used as Rx Buffer. 0 = Basic Mode disabled.
[1:0]	**Res**	**Reserved** There are reserved bits.

		These bits are always read as ‘0’ and must always be written with ‘0’.

Reset value: 0000 0000 R000 0000 b (R:current value of RX pin)

Write access to the Test Register is enabled by setting the Test bit in the CAN Control Register. The different test functions may be combined, but **Tx[1-0]** ≠ “00” disturbs message transfer.

Baud Rate Prescaler Extension REGISTER (CAN_BRPE)

Register	Offset	R/W	Description	Reset Value
CAN_BRPE	CAN0_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	Descriptions	
[31:4]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.
[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 IF*n* 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. Table 5-19 (IF1 and IF2 Message Interface Register Set) 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
CAN0_BA+0x20	IF1 Command Request	CAN0_BA+0x80	IF2 Command Request
CAN0_BA+0x24	IF1 Command Mask	CAN0_BA+0x84	IF2 Command Mask
CAN0_BA+0x28	IF1 Mask 1	CAN0_BA+0x88	IF2 Mask 1
CAN0_BA+0x2C	IF1 Mask 2	CAN0_BA+0x8C	IF2 Mask 2
CAN0_BA+0x30	IF1 Arbitration 1	CAN0_BA+0x90	IF2 Arbitration 1
CAN0_BA+0x34	IF1 Arbitration 2	CAN0_BA+0x94	IF2 Arbitration 2
CAN0_BA+0x38	IF1 Message Control	CAN0_BA+0x98	IF2 Message Control
CAN0_BA+0x3C	IF1 Data A 1	CAN0_BA+0x9C	IF2 Data A 1
CAN0_BA+0x40	IF1 Data A 2	CAN0_BA+0xA0	IF2 Data A 2
CAN0_BA+0x44	IF1 Data B 1	CAN0_BA+0xA4	IF2 Data B 1
CAN0_BA+0x48	IF1 Data B 2	CAN0_BA+0xA8	IF2 Data B 2

Table 5-19 IF1 and IF2 Message Interface Register

IFn Command Request Register (CAN_IFn_CREQ)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_CREQ	CAN0_BA+0x20/0x80	R/W	IFn Command Request 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**
Busy	**Res**						
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Res		**Message Number**					

Bits	Descriptions	
[15]	**Busy**	**Busy Flag** 1 = Writing to the IFn Command Request Register is in progress. This bit can only be read by the software. 0 = Read/write action has finished.
[14:6]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.
[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 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 IF*n* Command Mask Register specify the transfer direction and select which of the IF*n* Message Buffer Registers are source or target of the data transfer.

Register	Offset	R/W	Description	Reset Value
CAN_IFn_CMASK	CAN0_BA+0x24/0x84	R/W	IFn Command 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**
WR/RD	**Mask**	**Arb**	**Control**	**ClrIntPnd**	**TxRqst/ NewDat**	**Data A**	**Data B**

Bits	Descriptions	
[31:8]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.
[7]	**WR/RD**	**Write / Read** 1 = Write: Transfer data from the selected Message Buffer Registers to the Message Object addressed by the Command Request Register. 0 = Read: Transfer data from the Message Object addressed by the Command Request Register into the selected Message Buffer Registers.
[6]	**Mask**	**Access Mask Bits** Direction = Write 1 = Transfer **Identifier Mask** + **MDir** + **MXtd** to Message Object. 0: = Mask bits unchanged. Direction = Read 1 = Transfer **Identifier Mask** + **MDir** + **MXtd** to IFn Message Buffer Register. 0 = Mask bits unchanged.
[5]	**Arb**	**Access Arbitration Bits** Direction = Write 1 = Transfer **Identifier** + **Dir** + **Xtd** + **MsgVal** to Message Object 0 = Arbitration bits unchanged. Direction = Read 1 = Transfer **Identifier** + **Dir** + **Xtd** + **MsgVal** to IFn Message Buffer Register. 0 = Arbitration bits unchanged.

[4]	**Control**	**Control Access Control Bits** Direction = Write 1 = Transfer Control Bits to Message Object. 0 = Control Bits unchanged Direction = Read 1 = Transfer Control Bits to IFn Message Buffer Register. 0 = Control Bits unchanged.
[3]	**ClrIntPnd**	**Clear Interrupt Pending Bit** Direction = Write When writing to a Message Object, this bit is ignored. Direction = Read 1 = Clear **IntPnd** bit in the Message Object. 0 = **IntPnd** bit remains unchanged.
[2]	**TxRqst/NewDat**	**Access Transmission Request Bit** when Direction = Write 1 = Set TxRqst bit. 0 = TxRqst bit unchanged. 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 Direction = Read 1 = Clear **NewDat** bit in the Message Object 0 = **NewDat** bit remains unchanged. 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]	**Data A**	**Access Data Bytes [3:0]** Direction = Write 1 = Transfer Data Bytes [3:0] to Message Object 0 = Data Bytes [3:0] unchanged. Direction = Read 1 = Transfer Data Bytes [3:0] to IFn Message Buffer Register. 0 = Data Bytes [3:0] unchanged.
[0]	**Data B**	**Access Data Bytes [7:4]** Direction = Write 1 = Transfer Data Bytes [7:4] to Message Object. 0 = Data Bytes [7:4] unchanged. Direction = Read 1 = Transfer Data Bytes [7:4] to IFn Message Buffer Register. 0 = Data Bytes [7:4] unchanged.

IFn Mask 1 Register (CAN_IFn_MASK1)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_MASK1	CAN0_BA+0x28/0x88	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	Descriptions	
[31:16]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as '0' and must always be written with '0'.
[15:0]	**Msk[15:0]**	**Identifier Mask 15-0** 1 = The corresponding identifier bit is used for acceptance filtering. 0 = The corresponding bit in the identifier of the message object cannot inhibit the match in the acceptance filtering.

IFn Mask 2 Register (CAN_IFn_MASK2)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_MASK2	CAN0_BA+0x2C/0x8C	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	Descriptions	
[31:16]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.
[15]	**MXtd**	**Mask Extended Identifier** 1 = The extended identifier bit (IDE) is used for acceptance filtering. 0 = The extended identifier bit (IDE) has no effect on the 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**. For acceptance filtering, only these bits together with mask bits **Msk28** to **Msk18** are considered.
[14]	**MDir**	**Mask Message Direction** 1 = The message direction bit (**Dir**) is used for acceptance filtering. 0 = The message direction bit (**Dir**) has no effect on the acceptance filtering.
[13]	**Reserved**	**Reserved** This is reserved bit. The bit is always read as ‘1’ and must always be written with ‘1’.
[12:0]	**Msk[28:16]**	**Identifier Mask 28-16** 1 = The corresponding identifier bit is used for acceptance filtering. 0 = The corresponding bit in the identifier of the message object cannot inhibit the match in the acceptance filtering.

IFn Arbitration 1 Register (CAN_IFn_ARB1)

Register	Offset	R/W	Description	Reset Value
CAN_IFn_ARB1	CAN0_BA+0x30/0x90	R/W	IFn Arbitration 1 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**
ID[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.
[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	CAN0_BA+0x34/0x94	R/W	IFn Arbitration 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**
MsgVal	**Xtd**	**Dir**	**ID[28:24]**				
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ID[23:16]							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.
[15]	**MsgVal**	**Message Valid** 1 = The Message Object is configured and should be considered by the Message Handler. 0 = The Message Object is ignored 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** in the CAN Control Register. This bit must also be reset before the identifier **Id28-0,** the control bits **Xtd**, **Dir**, or the Data Length Code **DLC3-0** are modified, or if the Messages Object is no longer required.
[14]	**Xtd**	**Extended Identifier** 1 = The 29-bit (“extended”) Identifier will be used for this Message Object. 0 = The 11-bit (“standard”) Identifier will be used for this Message Object.
[13]	**Dir**	**Message Direction** 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 of this Message Object is set (if **RmtEn** = one). 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.
[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	CAN0_BA+0x38/0x98	R/W	IFn Message 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**
NewDat	**MsgLst**	**IntPnd**	**UMask**	**TxIE**	**RxIE**	**RmtEn**	**TxRqst**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
EoB	**Reserved**			**DLC[3:0]**			

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as '0' and must always be written with '0'.
[15]	**NewDat**	**New Data** 1 = The Message Handler or the application software has written new data into the data portion of this Message Object. 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.
[14]	**MsgLst**	**Message Lost (only valid for Message Objects with direction = receive)** 1 = The Message Handler stored a new message into this object when **NewDat** was still set, the CPU has lost a message. 0 = No message lost since last time this bit was reset by the CPU.
[13]	**IntPnd**	**Interrupt Pending** 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. 0 = This message object is not the source of an interrupt.
[12]	**UMask**	**Use Acceptance Mask** 1 = Use Mask (Msk28-0, MXtd, and MDir) for acceptance filtering. 0 = Mask ignored. **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** is set to one.
[11]	**TxIE**	**Transmit Interrupt Enable** 1 = **IntPnd** will be set after a successful transmission of a frame. 0 = **IntPnd** will be left unchanged after the successful transmission of a frame.
[10]	**RxIE**	**Receive Interrupt Enable**

		1 = **IntPnd** will be set after a successful reception of a frame. 0 = **IntPnd** will be left unchanged after a successful reception of a frame.
[9]	**RmtEn**	**Remote Enable** 1 = At the reception of a Remote Frame, **TxRqst** is set. 0 = At the reception of a Remote Frame, **TxRqst** is left unchanged.
[8]	**TxRqst**	**Transmit Request** 1 = The transmission of this Message Object is requested and is not yet done. 0 = This Message Object is not waiting for transmission.
[7]	**EoB**	**End of Buffer** 1 = Single Message Object or last Message Object of a FIFO Buffer. 0 = Message Object belongs to a FIFO Buffer and is not the last Message Object of that 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** There are reserved bits. These bits are always read as '0' and must always be written with '0'.
[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	CAN0_BA+0x3C/0x9C	R/W	IFn Data A1 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**
Data(1)							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data(0)							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.
[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	CAN0_BA+0x40/0xA0	R/W	IFn Data A2 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**
Data(3)							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data(2)							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as '0' and must always be written with '0'.
[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	CAN0_BA+0x44/0xA4	R/W	IFn Data B1 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**
Data(5)							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data(4)							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.
[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	CAN0_BA+0x48/0xA8	R/W	IFn Data B2 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**
Data(7)							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Data(6)							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.
[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 IF*n* Interface Registers. Table 5-20 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]	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7

Table 5-20 Structure of a Message Object in the Message Memory

The Arbitration Registers **ID28-0**, **Xtd**, and **Dir** are used to define the identifier and type of outgoing messages and are used (together with the mask registers **Msk28-0**, **MXtd**, and **MDir**) 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**, **NewDat**, **IntPnd**, and **MsgVal** 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	CAN0_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**
TxRqst 16-9							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TxRqst 8-1							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.
[15:0]	**TxRqst 16-1**	**Transmission Request Bits 16-1 (of all Message Objects)** 1 = The transmission of this Message Object is requested and is not yet done. 0 = This Message Object is not waiting for transmission. These bits are read only.

Transmission Request Register 2 (CAN_TXREQ2)

Register	Offset	R/W	Description	Reset Value
CAN_TXREQ2	CAN0_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	Descriptions	
[31:16]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.
[15:0]	**TxRqst 32-17**	**Transmission Request Bits 32-17 (of all Message Objects)** 1 = The transmission of this Message Object is requested and is not yet done. 0 = This Message Object is not waiting for transmission. 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	CAN0_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**
NewData 8-1							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as '0' and must always be written with '0'.
[15:0]	**NewData16-1**	**New Data Bits 16-1 (of all Message Objects)** 1 = The Message Handler or the application software has written new data into the data portion of this Message Object. 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.

New Data Register 2 (CAN_NDAT2)

Register	Offset	R/W	Description	Reset Value
CAN_NDAT2	CAN0_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**
NewData 32-25							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
NewData 24-17							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.
[15:0]	**NewData 32-17**	**New Data Bits 32-17 (of all Message Objects)** 1 = The Message Handler or the application software has written new data into the data portion of this Message Object. 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.

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	CAN0_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**
IntPnd 8-1							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as '0' and must always be written with '0'.
[15:0]	**IntPnd16-1**	**Interrupt Pending Bits 16-1 (of all Message Objects)** 1 = This message object is the source of an interrupt. 0 = This message object is not the source of an interrupt.

Interrupt Pending Register 2 (CAN_IPND2)

Register	Offset	R/W	Description	Reset Value
CAN_IPND2	CAN0_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**
IntPnd 32-25							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IntPnd 24-17							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as '0' and must always be written with '0'.
[15:0]	**IntPnd 32-17**	**Interrupt Pending Bits 32-17(of all Message Objects)** 1 = This message object is the source of an interrupt. 0 = This message object is not 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	CAN0_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**
MsgVal 16- 9							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MsgVal 8-1							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as '0' and must always be written with '0'.
[15:0]	**MsgVal 16-1**	**Message Valid Bits 16-1 (of all Message Objects)** (Read Only) 1 = This Message Object is configured and should be considered by the Message Handler. 0 = This Message Object is ignored 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	CAN0_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**
MsgVal 32-25							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
MsgVal 24-17							

Bits	Descriptions	
[31:16]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as '0' and must always be written with '0'.
[15:0]	**MsgVal 32-17**	**Message Valid Bits 32-17 (of all Message Objects)** (Read only) 1 = This Message Object is configured and should be considered by the Message Handler. 0 = This Message Object is ignored 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 Register (CAN_WU_EN)

Register	Offset	R/W	Description	Reset Value
CAN_WU_EN	CAN0_BA+0x168	R/W	Wake Up 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							**WAKUP_EN**

Bits	Descriptions	
[31:1]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.
[0]	**WAKUP_EN**	**Wake Up Enable** 1 = The wake-up function is enable. 0 = The wake-up function is disable. **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	CAN0_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	Descriptions	
[31:1]	**Reserved**	**Reserved** There are reserved bits. These bits are always read as ‘0’ and must always be written with ‘0’.
[0]	**WAKUP_STS**	**Wake Up Status** 1 = Wake-up event is occurred. 0 = No wake-up event is occurred. **Note**: The bit can be written ‘0’ to clear.

5.14 PS/2 Device Controller (PS2D)

5.14.1 Overview

PS/2 device controller provides basic timing control for PS/2 communication. All communication between the device and the host is managed through the CLK and DATA pins. Unlike PS/2 keyboard or mouse device controller, the received/transmit code needs to be translated as meaningful code by firmware. The device controller generates the CLK signal after receiving a request to send, but host has ultimate control over communication. DATA sent from the host to the device is read on the rising edge and DATA sent from device to the host is change after rising edge. A 16 bytes FIFO is used to reduce CPU intervention. S/W can select 1 to 16 bytes for a continuous transmission.

5.14.2 Features

- Host communication inhibit and request to send detection
- Reception frame error detection
- Programmable 1 to 16 bytes transmit buffer to reduce CPU intervention
- Double buffer for data reception
- S/W override bus

5.14.3 Block Diagram

The PS/2 device controller consists of APB interface and timing control logic for DATA and CLK lines.

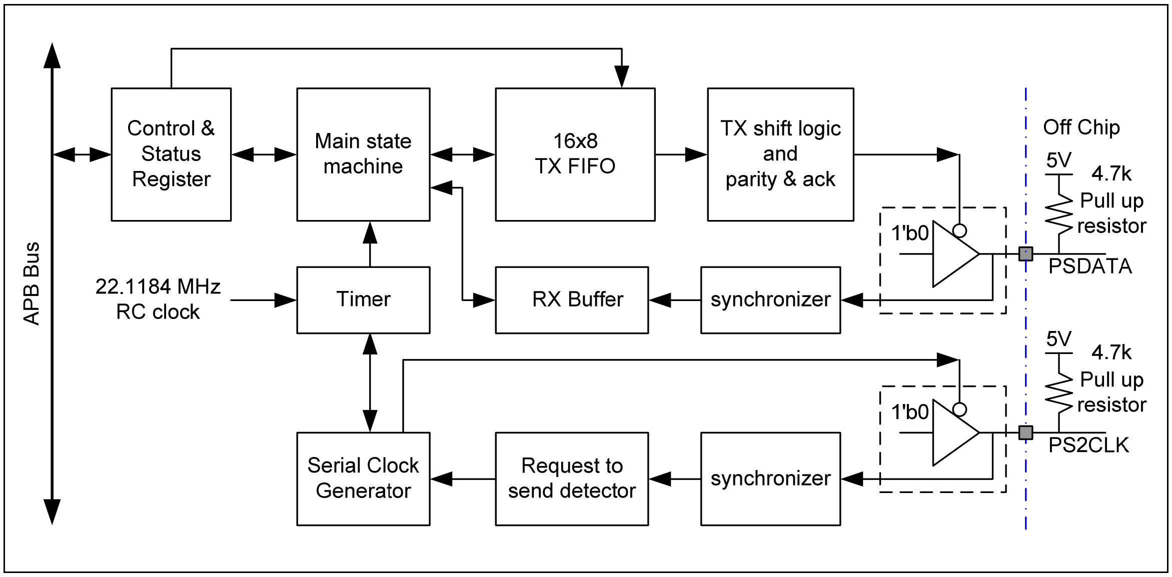


Figure 5-88 PS/2 Device Block Diagram

5.14.4 Functional Description

5.14.4.1 Communication

The PS/2 device implements a bidirectional synchronous serial protocol. The bus is "Idle" when both lines are high (open-collector). This is the only state where the device is allowed start to transmit DATA. The host has ultimate control over the bus and may inhibit communication at any time by pulling the CLK line low.

The CLK signal is generated by PS/2 device. If the host wants to send DATA, it must first inhibit communication from the device by pulling CLK low. The host then pulls DATA low and releases CLK. This is the "Request-to-Send" state and signals the device to start generating CLK pulses.

DATA	CLK	Bus State
High	High	Idle
High	Low	Communication Inhibit
Low	High	Host Request to Send

All data is transmitted one byte at a time and each byte is sent in a frame consisting of 11 or 12 bits. These bits are:

- 1 start bit. This is always 0
- 8 DATA bits, least significant bit first
- 1 parity bit (odd parity)
- 1 stop bit. This is always 1
- 1 acknowledge bit (host-to-device communication only)

The parity bit is set if there is an even number of 1's in the data bits and cleared to 0 if there is an odd number of 1's in the data bits. The number of 1's in the data bits plus the parity bit always add up to an odd number set to 1. This is used for error detection. The device must check this bit and if incorrect it should respond as if it had received an invalid command.

The host may inhibit communication at any time by pulling the CLK line low for at least 100 microseconds. If a transmission is inhibited before the 11th clock pulse, the device must abort the current transmission and prepare to retransmit the current data when host releases Clock. In order to reserve enough time for s/w to decode host command, the transmit logic is blocked by RXINT bit, S/W must clear RXINT bit to start retransmit. S/W can write CLRFIFO to 1 to reset FIFO pointer if need.

Device-to-Host

The device uses a serial protocol with 11-bit frames. These bits are:

- 1 start bit. This is always 0
- 8 DATA bits, least significant bit first
- 1 parity bit (odd parity)
- 1 stop bit. This is always 1

The device writes a bit on the DATA line when CLK is high, and it is read by the host when CLK is low. Figure 5-89 in the following illustrate this.

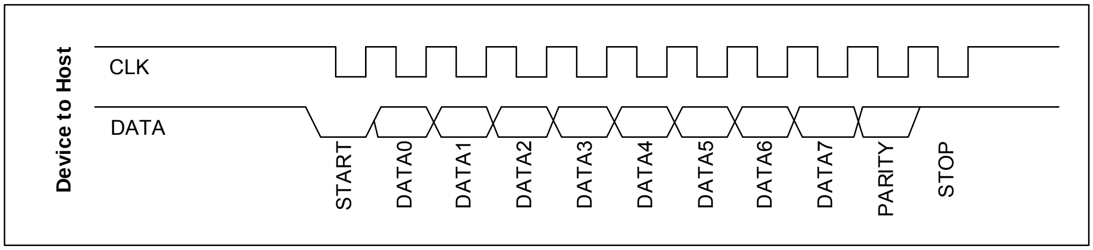


Figure 5-89 Data Format of Device-to-Host

Host-to-Device:

First of all, the PS/2 device always generates the CLK signal. If the host wants to send DATA, it must first put the CLK and DATA lines in a "Request-to-send" state as follows:

- Inhibit communication by pulling CLK low for at least 100 microseconds
- Apply "Request-to-send" by pulling DATA low, then release CLK

The device should check for this state at intervals not to exceed 10 milliseconds. When the device detects this state, it will begin generating CLK signals and CLK in eight DATA bits and one stop bit. The host changes the DATA line only when the CLK line is low, and DATA is read by the device when CLK is high.

After the stop bit is received, the device will acknowledge the received byte by bringing the DATA line low and generating one last CLK pulse. If the host does not release the DATA line after the 11th CLK pulse, the device will continue to generate CLK pulses until the DATA line is released.

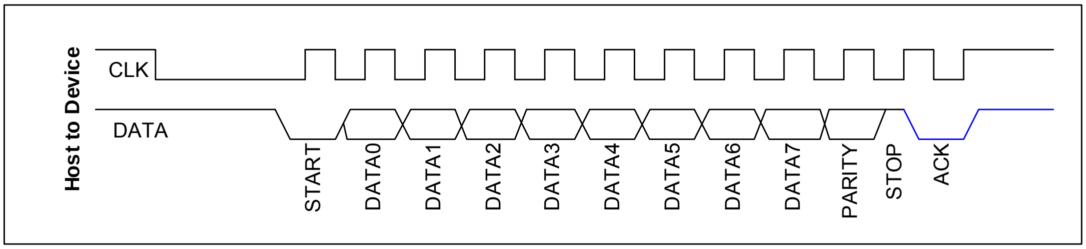


Figure 5-90 Data Format of Host-to-Device

The host and the device DATA and CLK detailed timing for communication is shown as below:

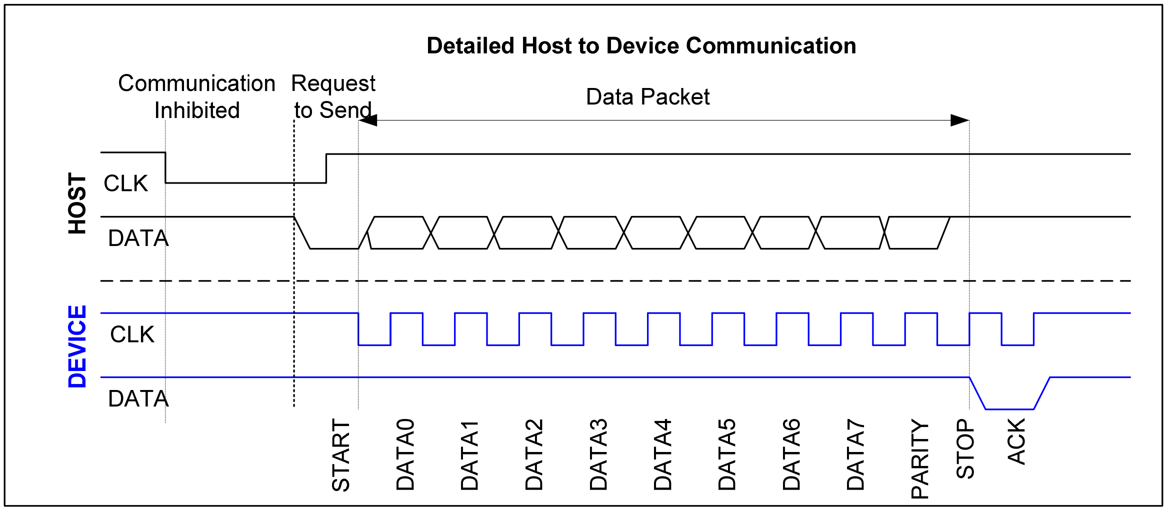


Figure 5-91 PS/2 Bit Data Format

5.14.4.2 PS/2 Bus Timing Specification

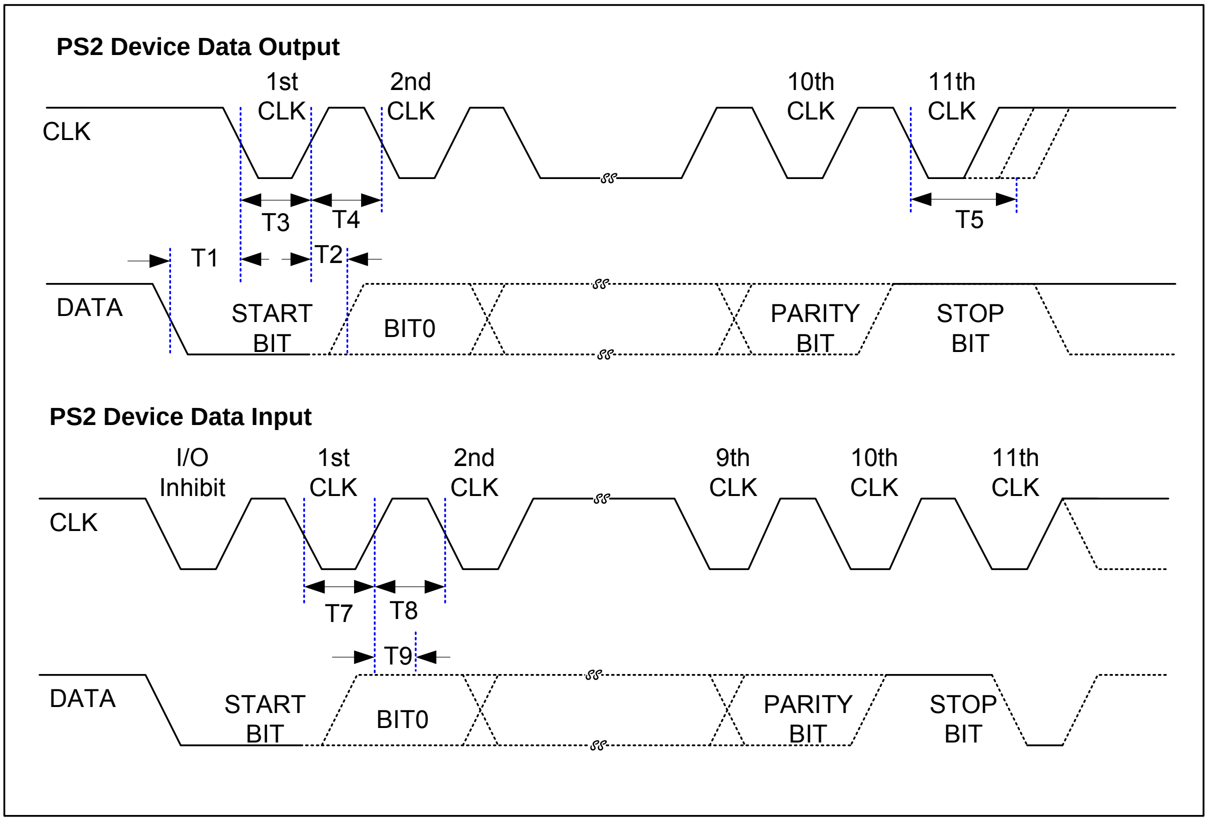


Figure 5-92 PS/2 Bus Timing

Symbol	Timing Parameter	MIN.	Max
T1	DATA transition to the falling edge of CLK	5us	25us
T2	Rising edge of CLK to DATA transition	5us	T4-5us
T3	Duration of CLK inactive	30us	50us
T4	Duration of CLK active	30us	50us
T5	Time to auxiliary device inhibit after 11th clock to ensure auxiliary device does not start another transmission	>0	50us
T7	Duration of CLK inactive	30us	50us
T8	Duration of CLK active	30us	50us
T9	Time from inactive to active CLK transition, use to time auxiliary device sample DATA	5us	25us

5.14.4.3 TX FIFO Operation

Writing PS2TXDATA0 register starts device to host communication. S/W is required to define TXFIFO depth before writing transmission data to TX FIFO. 1st START bit is sent to PS/2 bus 100us after S/W writes TX FIFO, if there is more than 4 bytes data need to be sent, S/W can write residual data to PS2TXDATA1-3 before 4th byte transmit complete. A time delay 100us is added between two consecutive bytes.

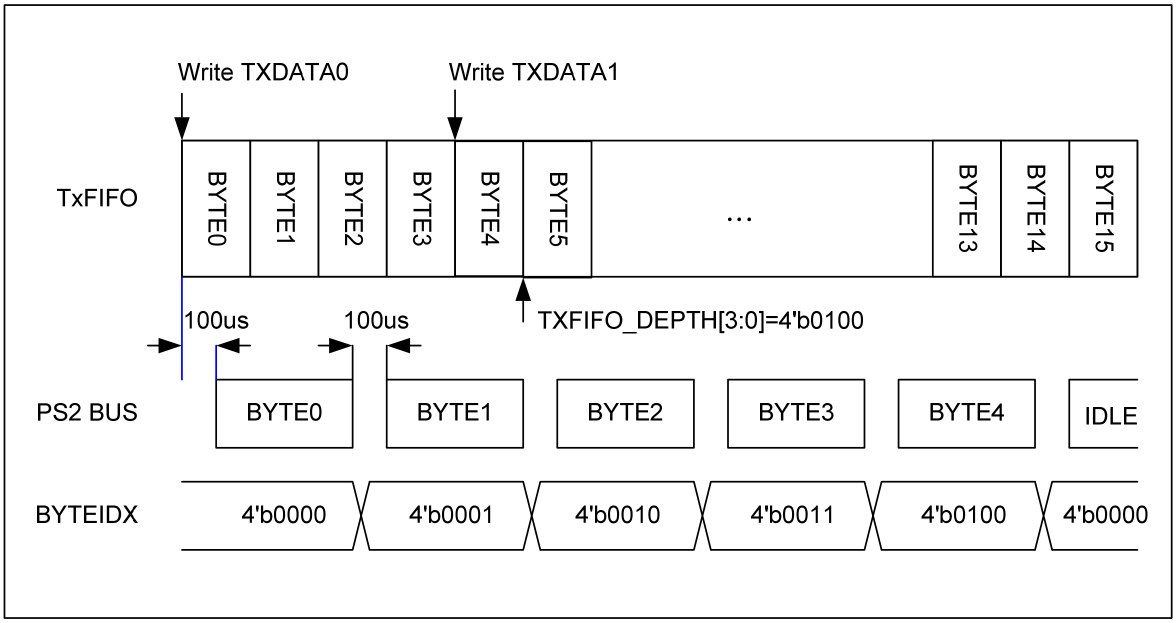


Figure 5-93 PS/2 Data Format

5.14.5 Register Map

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

Register	Offset	R/W	Description	Reset Value
PS2_BA: 0x4010_0000				
PS2CON	PS2_BA+0x00	R/W	PS/2 Control Register	0x0000_0000
PS2TXDATA0	PS2_BA+0x04	R/W	PS/2 Transmit DATA Register 0	0x0000_0000
PS2TXDATA1	PS2_BA+0x08	R/W	PS/2 Transmit DATA Register 1	0x0000_0000
PS2TXDATA2	PS2_BA+0x0C	R/W	PS/2 Transmit DATA Register 2	0x0000_0000
PS2TXDATA3	PS2_BA+0x10	R/W	PS/2 Transmit DATA Register 3	0x0000_0000
PS2RXDATA	PS2_BA+0x14	R	PS/2 Receive DATA Register	0x0000_0000
PS2STATUS	PS2_BA+0x18	R/W	PS/2 Status Register	0x0000_0083
PS2INTID	PS2_BA+0x1C	R/W	PS/2 Interrupt Identification Register	0x0000_0000

5.14.6 Register Description

PS/2 Control Register (PS2CON)

Register	Offset	R/W	Description	Reset Value
PS2CON	PS2_BA+0x00	R/W	PS/2 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				**FPS2DAT**	**FPS2CLK**	**OVERRIDE**	**CLRFIFO**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ACK	**TXFIFO_DEPTH**				**RXINTEN**	**TXINTEN**	**PS2EN**

Bits	Descriptions	
[31:12]	**Reserved**	Reserved
[11]	**FPS2DAT**	**Force PS2DATA Line** It forces PS2DATA high or low regardless of the internal state of the device controller if OVERRIDE is set to high. 1 = Force PS2DATA high 0 = Force PS2DATA low
[10]	**FPS2CLK**	**Force PS2CLK Line** It forces PS2CLK line high or low regardless of the internal state of the device controller if OVERRIDE is set to high. 1 = Force PS2CLK line high 0 = Force PS2CLK line low
[9]	**OVERRIDE**	**Software Override PS/2 CLK/DATA Pin State** 1 = PS2CLK and PS2DATA pins are controlled by S/W 0 = PS2CLK and PS2DATA pins are controlled by internal state machine.
[8]	**CLRFIFO**	**Clear TX FIFO** Write 1 to this bit to terminate device to host transmission. The TXEMPTY bit in PS2STATUS bit will be set to 1 and pointer BYTEIDEX is reset to 0 regardless there is residue data in buffer or not. The buffer content is not been cleared. 1 = Clear FIFO 0 = Not active
[7]	**ACK**	**Acknowledge Enable** 1 = If parity error or stop bit is not received correctly, acknowledge bit will not be sent to

		host at 12th clock 0 = Always send acknowledge to host at 12th clock for host to device communication.
[6:3]	**TXFIFODIPTH**	**Transmit Data FIFO Depth** There is 16 bytes buffer for data transmit. S/W can define the FIFO depth from 1 to 16 bytes depends on application. 0 = 1 byte 1 = 2 bytes … 14 = 15 bytes 15 = 16 bytes
[2]	**RXINTEN**	**Enable Receive Interrupt** 1 = Enable data receive complete interrupt 0 = Disable data receive complete interrupt
[1]	**TXINTEN**	**Enable Transmit Interrupt** 1 = Enable data transmit complete interrupt 0 = Disable data transmit complete interrupt
[0]	**PS2EN**	**Enable PS/2 Device** Enable PS/2 device controller 1 = Enable 0 = Disable

PS/2 TX DATA Register 0-3 (PS2TXDATA0-3)

Register	Offset	R/W	Description	Reset Value
PS2TXDATA0	PS2_BA+0x04	R/W	PS/2 Transmit Data Register0	0x0000_0000
PS2TXDATA1	PS2_BA+0x08	R/W	PS/2 Transmit Data Register1	0x0000_0000
PS2TXDATA2	PS2_BA+0x0C	R/W	PS/2 Transmit Data Register2	0x0000_0000
PS2TXDATA3	PS2_BA+0x10	R/W	PS/2 Transmit Data Register3	0x0000_0000

31	30	29	28	27	26	25	24
PS2TXDATAx[31:24]							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PS2TXDATAx[23:16]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PS2TXDATAx[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PS2TXDATAx[7:0]							

Bits	Descriptions	
[31:0]	**PS2TXDATAx**	**Transmit data** Write data to this register starts device to host communication if bus is in IDLE state. S/W must enable PS2EN before writing data to TX buffer.

PS/2 Receiver DATA Register (PS2RXDATA)

Register	Offset	R/W	Description	Reset Value
PS2RXDATA	PS2_BA+0x14	R	PS/2 Receive 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**
RXDATA[7:0]							

Bits	Descriptions	
[31:8]	**Reserved**	Reserved
[7:0]	**PS2RXDATA**	**Received Data** For host to device communication, after acknowledge bit is sent, the received data is copied from receive shift register to PS2RXDATA register. CPU must read this register before next byte reception complete, otherwise the data will be overwritten and RXOVF bit in PS2STATUS[6] will be set to 1.

PS/2 Status Register (PS2STATUS)

Register	Offset	R/W	Description	Reset Value
PS2STATUS	PS2_BA+0x18	R/W	PS/2 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				**BYTEIDX[3:0]**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TXEMPTY	**RXOVF**	**TXBUSY**	**RXBUSY**	**RXPARITY**	**FRAMERR**	**PS2DATA**	**PS2CLK**

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[31:12]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[11:8]</td><td>BYTEIDX</td><td>Byte Index
It indicates which data byte in transmit data shift register. When all data in FIFO is transmitted and it will be cleared to 0.
It is a read only bit.
<table>
<tr><th>BYTEIDX</th><th>DATA Transmit</th><th>BYTEIDX</th><th>DATA Transmit</th></tr>
<tr><td>0000</td><td>TXDATA0[7:0]</td><td>1000</td><td>TXDATA2[7:0]</td></tr>
<tr><td>0001</td><td>TXDATA0[15:8]</td><td>1001</td><td>TXDATA2[15:8]</td></tr>
<tr><td>0010</td><td>TXDATA0[23:16]</td><td>1010</td><td>TXDATA2[23:16]</td></tr>
<tr><td>0011</td><td>TXDATA0[31:24]</td><td>1011</td><td>TXDATA2[31:24]</td></tr>
<tr><td>0100</td><td>TXDATA1[7:0]</td><td>1100</td><td>TXDATA3[7:0]</td></tr>
<tr><td>0101</td><td>TXDATA1[15:8]</td><td>1101</td><td>TXDATA3[15:8]</td></tr>
<tr><td>0110</td><td>TXDATA1[23:16]</td><td>1110</td><td>TXDATA3[23:16]</td></tr>
<tr><td>0111</td><td>TXDATA1[31:24]</td><td>1111</td><td>TXDATA3[31:24]</td></tr>
</table></td></tr>
<tr><td>[7]</td><td>TXEMPTY</td><td>TX FIFO Empty
When S/W writes any data to PS2TXDATA0-3 the TXEMPTY bit is cleared to 0 immediately if PS2EN is enabled. When transmitted data byte number is equal to FIFODEPTH then TXEMPTY bit is set to 1.
1 = FIFO is empty
0 = There is data to be transmitted
Read only bit.</td></tr>
<tr><td>[6]</td><td>RXOVF</td><td>RX Buffer Overwrite</td></tr>
</table>

		1 = Data in PS2RXDATA register is overwritten by new received data 0 = No overwrite Write 1 to clear this bit.
[5]	**TXBUSY**	**Transmit Busy** This bit indicates that the PS/2 device is currently sending data. 1 = Currently sending data 0 = Idle Read only bit.
[4]	**RXBUSY**	**Receive Busy** This bit indicates that the PS/2 device is currently receiving data. 1 = Currently receiving data 0 = Idle Read only bit.
[3]	**RXPARITY**	**Received Parity** This bit reflects the parity bit for the last received data byte (odd parity). Read only bit.
[2]	**FRAMERR**	**Frame Error** For host to device communication, if STOP bit (logic 1) is not received it is a frame error. If frame error occurs, DATA line may keep at low state after 12th clock. At this moment, S/W overrides PS2CLK to send clock till PS2DATA release to high state. After that, device sends a “Resend” command to host. 1 = Frame error occur 0 = No frame error Write 1 to clear this bit.
[1]	**PS2DATA**	**DATA Pin State** This bit reflects the status of the PS2DATA line after synchronizing and sampling.
[0]	**PS2CLK**	**CLK Pin State** This bit reflects the status of the PS2CLK line after synchronizing.

PS/2 Interrupt Identification Register (PS2INTID)

Register	Offset	R/W	Description	Reset Value
PS2INTID	PS2_BA+0x1C	R/W	PS/2 Interrupt Identification 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						**TXINT**	**RXINT**

Bits	Descriptions	
[31:3]	**Reserved**	Reserved
[1]	**TXINT**	**Transmit Interrupt** This bit is set to 1 after STOP bit is transmitted. Interrupt occur if TXINTEN bit is set to 1. 1 = Transmit interrupt occurs 0 = No interrupt Write 1 to clear this bit to 0.
[0]	**RXINT**	**Receive Interrupt** This bit is set to 1 when acknowledge bit is sent for Host to device communication. Interrupt occurs if RXINTEN bit is set to 1. 1 = Receive interrupt occurs 0 = No interrupt Write 1 to clear this bit to 0.

5.15 I^{2}S Controller (I^{2}S)

5.15.1 Overview

The I^{2}S controller consists of IIS protocol to interface with external audio CODEC. Two 8 word deep FIFO for read path and write path respectively and is capable of handling 8 ~ 32 bit word sizes. DMA controller handles the data movement between FIFO and memory.

5.15.2 Features

- I^{2}S can operate as either master or slave
- Capable of handling 8-, 16-, 24- and 32-bit word sizes
- Mono and stereo audio data supported
- I^{2}S and MSB justified data format supported
- Two 8 word FIFO data buffers are provided, one for transmit and one for receive
- Generates interrupt requests when buffer levels cross a programmable boundary
- Two DMA requests, one for transmit and one for receive

5.15.3 Block Diagram

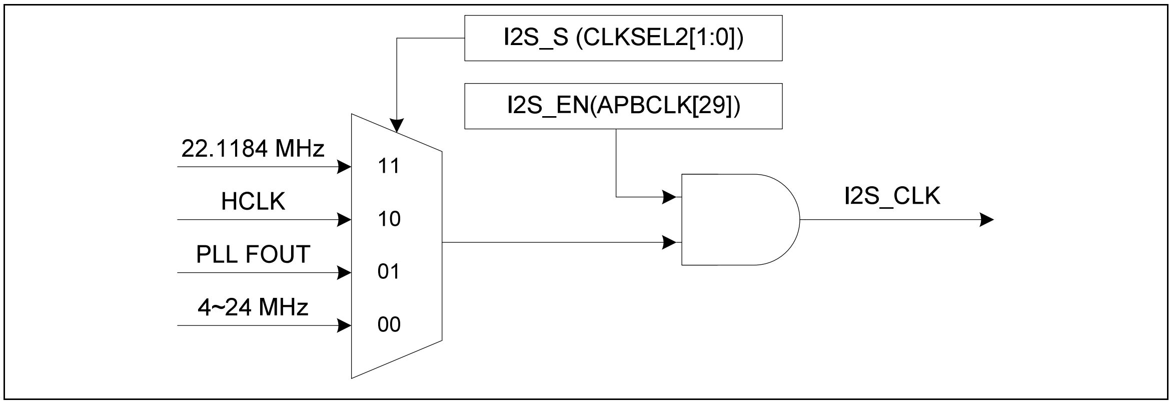


Figure 5-94 I^{2}S Clock Control Diagram

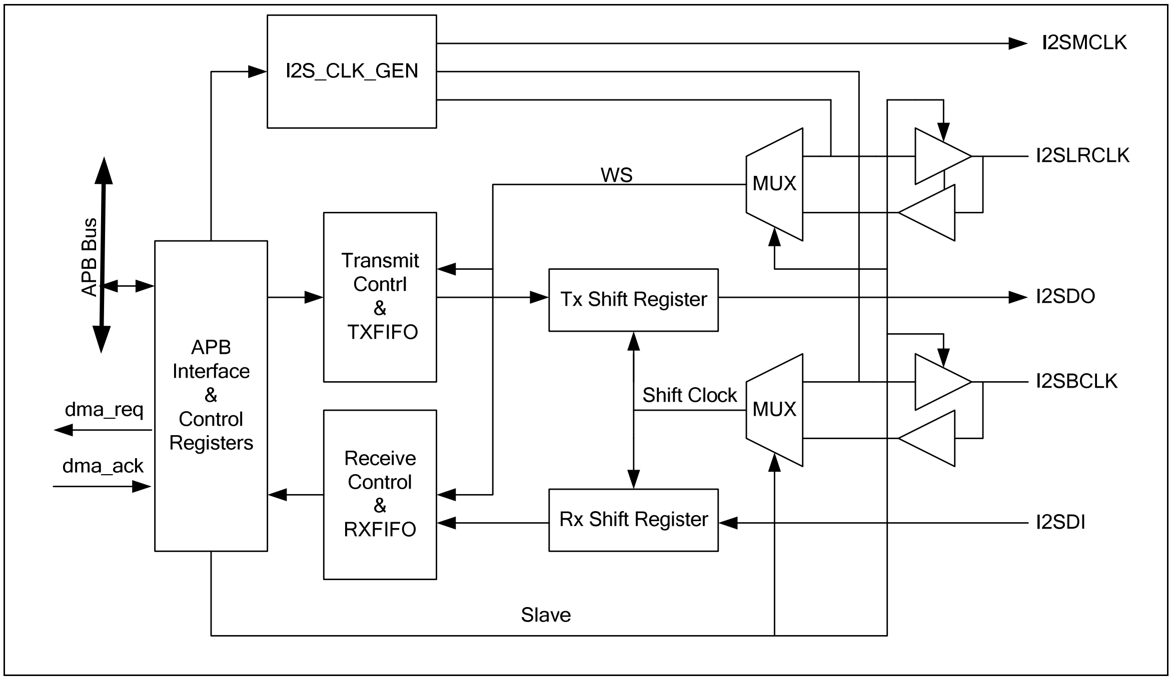


Figure 5-95 I^{2}S Controller Block Diagram

5.15.4 Functional Description

5.15.4.1 I^2S Operation

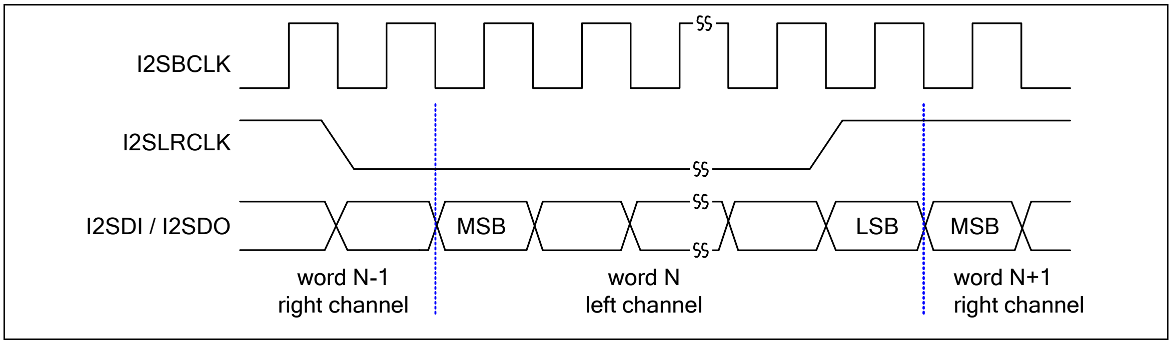


Figure 5-96 I^2S Bus Timing Diagram (Format =0)

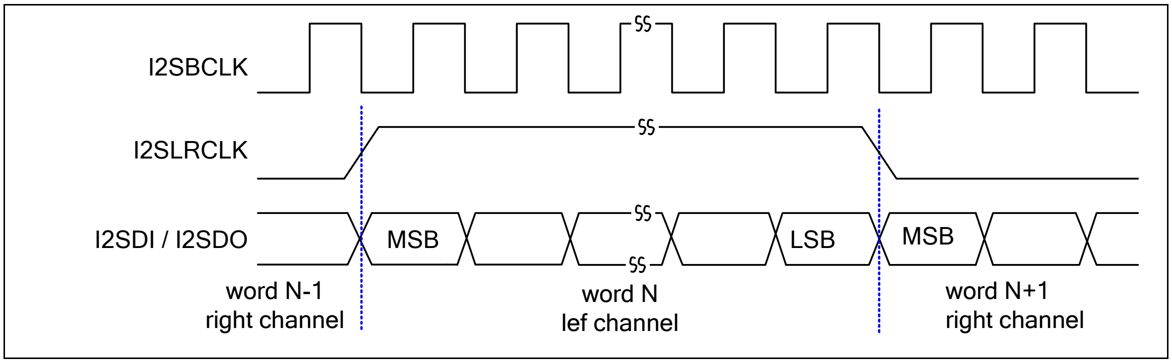


Figure 5-97 MSB Justified Timing Diagram (Format=1)

5.15.4.2 FIFO operation

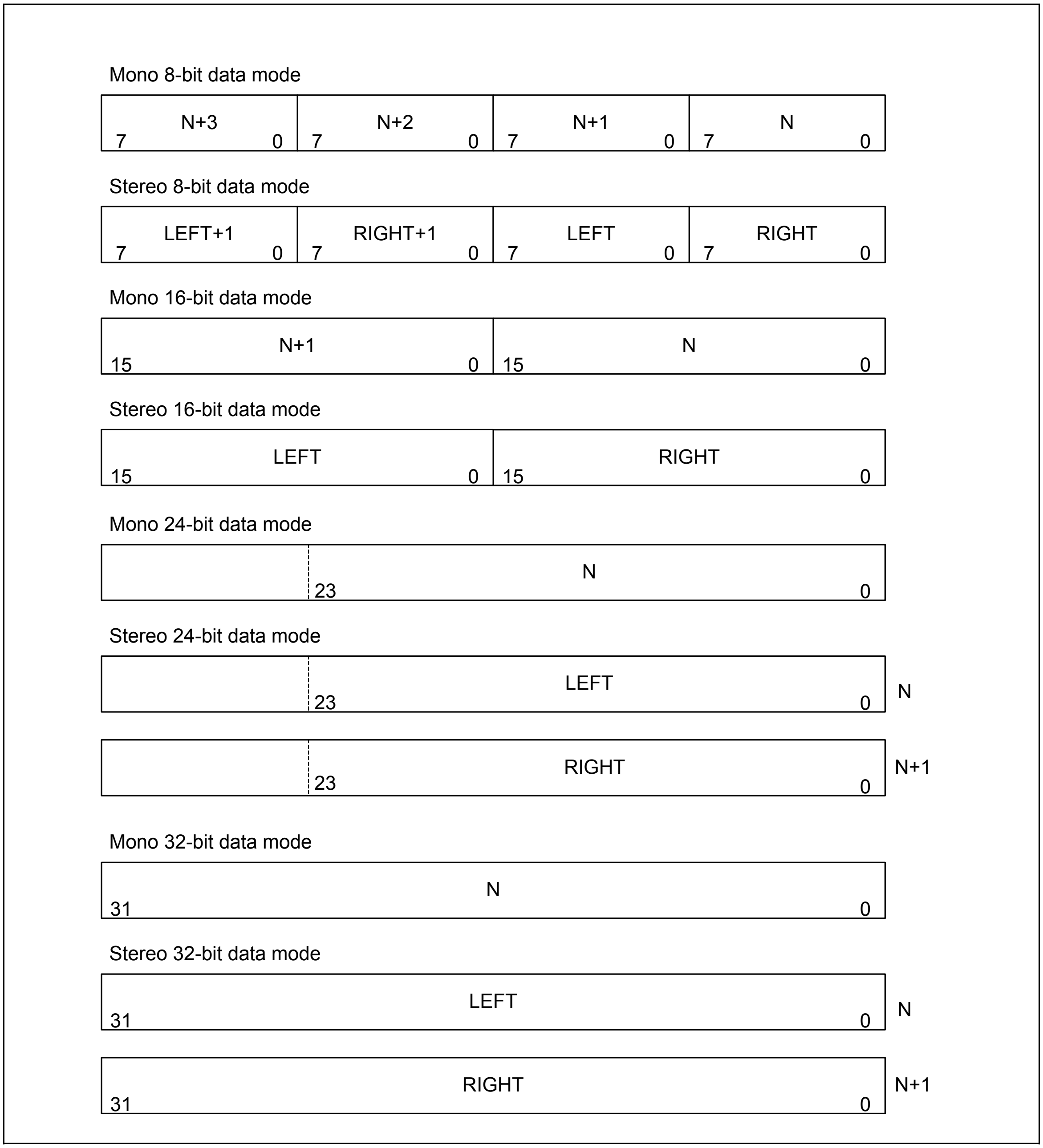


Figure 5-98 FIFO contents for various I^{2}S modes

5.15.5 Register Map

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

Register	Offset	R/W	Description	Reset Value
I2S_BA = 0x401A_0000				
I2S_CON	I2S_BA+0x00	R/W	I^2S Control Register	0x0000_0000
I2S_CLKDIV	I2S_BA+0x04	R/W	I^2S Clock Divider Register	0x0000_0000
I2S_IE	I2S_BA+0x08	R/W	I^2S Interrupt Enable Register	0x0000_0000
I2S_STATUS	I2S_BA+0x0C	R/W	I^2S Status Register	0x0014_1000
I2S_TXFIFO	I2S_BA+0x10	R/W	I^2S Transmit FIFO Register	0x0000_0000
I2S_RXFIFO	I2S_BA+0x14	R/W	I^2S Receive FIFO Register	0x0000_0000

5.15.6 Register Description

I²S Control Register (I2S_CON)

Register	Offset	R/W	Description	Reset Value
I2S_CON	I2S_BA+0x00	R/W	I²S Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved	**Reserved**	**RXDMA**	**TXDMA**	**CLR_RXFIFO**	**CLR_TXFIFO**	**LCHZCEN**	**RCHZCEN**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
MCLKEN	**RXTH[2:0]**			**TXTH[2:0]**			**SLAVE**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
FORMAT	**MONO**	**WORDWIDTH**		**MUTE**	**RXEN**	**TXEN**	**I2SEN**

Bits	Descriptions	
[31:22]	**Reserved**	Reserved
[21]	**RXDMA**	**Enable Receive DMA** When RX DMA is enabled, I²S requests DMA to transfer data from receive FIFO to SRAM if FIFO is not empty. 1 = Enable RX DMA 0 = Disable RX DMA
[20]	**TXDMA**	**Enable Transmit DMA** When TX DMA is enables, I²S request DMA to transfer data from SRAM to transmit FIFO if FIFO is not full. 1 = Enable TX DMA 0 = Disable TX DMA
[19]	**CLR_RXFIFO**	**Clear Receive FIFO** Write 1 to clear receive FIFO, internal pointer is reset to FIFO start point, and RXFIFO_LEVEL[3:0] returns to zero and receive FIFO becomes empty. This bit is cleared by hardware automatically, read it return zero.
[18]	**CLR_TXFIFO**	**Clear Transmit FIFO** Write 1 to clear transmit FIFO, internal pointer is reset to FIFO start point, and TXFIFO_LEVEL[3:0] returns to zero and transmit FIFO becomes empty but data in transmit FIFO is not changed. This bit is clear by hardware automatically, read it return zero.

[17]	**LCHZCEN**	**Left channel zero cross detect enable** If this bit is set to 1, when left channel data sign bit change or next shift data bits are all zero then LZCF flag in I2S_STATUS register is set to 1. 1 = Enable left channel zero cross detect 0 = Disable left channel zero cross detect
[16]	**RCHZCEN**	**Right channel zero cross detect enable** If this bit is set to 1, when right channel data sign bit change or next shift data bits are all zero then RZCF flag in I2S_STATUS register is set to 1. 1 = Enable right channel zero cross detect 0 = Disable right channel zero cross detect
[15]	**MCLKEN**	**Master clock enable** If NuMicro™ NUC100 series, external crystal clock is frequency 2*N*256fs then software can program MCLK_DIV[2:0] in I2S_CLKDIV register to get 256fs clock to audio codec chip. 1 = Enable master clock 0 = Disable master clock
[14:12]	**RXTH[2:0]**	**Receive FIFO threshold level** When received data word(s) in buffer is equal or higher than threshold level then RXTHF flag is set. 000 = 1 word data in receive FIFO 001 = 2 word data in receive FIFO 010 = 3 word data in receive FIFO 011 = 4 word data in receive FIFO 100 = 5 word data in receive FIFO 101 = 6 word data in receive FIFO 110 = 7 word data in receive FIFO 111 = 8 word data in receive FIFO
[11:9]	**TXTH[2:0]**	**Transmit FIFO threshold level** If remain data word (32 bits) in transmit FIFO is the same or less than threshold level then TXTHF flag is set. 000 = 0 word data in transmit FIFO 001 = 1 word data in transmit FIFO 010 = 2 words data in transmit FIFO 011 = 3 words data in transmit FIFO 100 = 4 words data in transmit FIFO 101 = 5 words data in transmit FIFO 110 = 6 words data in transmit FIFO 111 = 7 words data in transmit FIFO

[8]	**SLAVE**	**Slave mode** I^{2}S can operate as master or slave. For master mode, I2S_BCLK and I2S_LRCLK pins are output mode and send bit clock from NuMicro™ NUC100 series to Audio CODEC chip. In slave mode, I2S_BCLK and I2S_LRCLK pins are input mode and I2S_BCLK and I2S_LRCLK signals are received from outer Audio CODEC chip. 1 = Slave mode 0 = Master mode
[7]	**FORMAT**	**Data format** 1 = MSB justified data format 0 = I^{2}S data format
[6]	**MONO**	**Monaural data** 1 = Data is monaural format 0 = Data is stereo format
[5:4]	**WORDWIDTH**	**Word width** 00 = data is 8-bit 01 = data is 16-bit 10 = data is 24-bit 11 = data is 32-bit
[3]	**MUTE**	**Transmit mute enable** 1= Transmit channel zero 0 = Transmit data is shifted from buffer
[2]	**RXEN**	**Receive enable** 1 = Enable data receive 0 = Disable data receive
[1]	**TXEN**	**Transmit enable** 1 = Enable data transmit 0 = Disable data transmit
[0]	**I2SEN**	**Enable I^{2}S controller** 1 = Enable 0 = Disable

I^{2}S Clock Divider (I2S_CLKDIV)

Register	Offset	R/W	Description	Reset Value
I2S_CLKDIV	I2S_BA+0x04	R/W	I^{2}S Clock Divider 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**
BCLK_DIV [7:0]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**MCLK_DIV[2:0]**		

Bits	Descriptions	
[31:16]	**Reserved**	Reserved
[15:8]	**BCLK_DIV [7:0]**	**Bit Clock Divider** If I^{2}S operates in master mode, bit clock is provided by NuMicro™ NUC100 series. Software can program these bits to generate sampling rate clock frequency. F_BCLK = F_I2SCLK /(2x(BCLK_DIV + 1))
[7:3]	**Reserved**	Reserved
[2:0]	**MCLK_DIV[2:0]**	**Master Clock Divider** If chip external crystal frequency is (2xMCLK_DIV)*256fs then software can program these bits to generate 256fs clock frequency to audio codec chip. If MCLK_DIV is set to 0, MCLK is the same as external clock input. For example, sampling rate is 24 kHz and chip external crystal clock is 12.288 MHz, set MCLK_DIV=1. F_MCLK = F_I2SCLK/(2x(MCLK_DIV)) (When MCLK_DIV is >= 1) F_MCLK = F_I2SCLK (When MCLK_DIV is set to 0)

I^{2}S Interrupt Enable Register (I2S_IE)

Register	Offset	R/W	Description	Reset Value
I2S_IE	I2S_BA+0x08	R/W	I^{2}S 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			**LZCIE**	**RZCIE**	**TXTHIE**	**TXOVFIE**	**TXUDFIE**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**RXTHIE**	**RXOVFIE**	**RXUDFIE**

Bits	Descriptions	
[31:13]	**Reserved**	Reserved
[12]	**LZCIE**	**Left channel zero cross interrupt enable** Interrupt occur if this bit is set to 1 and left channel zero cross 1 = Enable interrupt 0 = Disable interrupt
[11]	**RZCIE**	**Right channel zero cross interrupt enable** 1 = Enable interrupt 0 = Disable interrupt
[10]	**TXTHIE**	**Transmit FIFO threshold level interrupt enable** Interrupt occurs if this bit is set to 1 and data words in transmit FIFO is less than TXTH[2:0]. 1 = Enable interrupt 0 = Disable interrupt
[9]	**TXOVFIE**	**Transmit FIFO overflow interrupt enable** Interrupt occurs if this bit is set to 1 and transmit FIFO overflow flag is set to 1 1 = Enable interrupt 0 = Disable interrupt
[8]	**TXUDFIE**	**Transmit FIFO underflow interrupt enable** Interrupt occur if this bit is set to 1 and transmit FIFO underflow flag is set to 1. 1 = Enable interrupt 0 = Disable interrupt
[7:3]	**Reserved**	Reserved

[2]	**RXTHIE**	**Receive FIFO threshold level interrupt enable** When data word in receive FIFO is equal or higher then RXTH[2:0] and the RXTHF bit is set to 1. If RXTHIE bit is enabled, interrupt occur. 1 = Enable interrupt 0 = Disable interrupt
[1]	**RXOVFIE**	**Receive FIFO overflow interrupt enable** 1 = Enable interrupt 0 = Disable interrupt
[0]	**RXUDFIE**	**Receive FIFO underflow interrupt enable** If software read receive FIFO when it is empty then RXUDF flag in I2SSTATUS register is set to 1. 1 = Enable interrupt 0 = Disable interrupt

I^{2}S Status Register (I2S_STATUS)

Register	Offset	R/W	Description	Reset Value
I2S_STATUS	I2S_BA+0x0C	R/W	I^{2}S Status Register	0x0014_1000

31	30	29	28	27	26	25	24
TX_LEVEL[3:0]				RX_LEVEL[3:0]			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
LZCF	RZCF	TXBUSY	TXEMPTY	TXFULL	TXTHF	TXOVF	TXUDF
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved			RXEMPTY	RXFULL	RXTHF	RXOVF	RXUDF
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				RIGHT	I2STXINT	I2SRXINT	I2SINT

Bits	Descriptions	
[31:28]	**TX_LEVEL**	**Transmit FIFO level** These bits indicate word number in transmit FIFO 0000 = No data 0001 = 1 word in transmit FIFO …. 1000 = 8 words in transmit FIFO
[27:24]	**RX_LEVEL**	**Receive FIFO level** These bits indicate word number in receive FIFO 0000 = No data 0001 = 1 word in receive FIFO …. 1000 = 8 words in receive FIFO
[23]	**LZCF**	**Left channel zero cross flag** It indicates left channel next sample data sign bit is changed or all data bits are zero. 1 = Left channel zero cross is detected 0 = No zero cross Software can write 1 to clear this bit to zero
[22]	**RZCF**	**Right channel zero cross flag** It indicates right channel next sample data sign bit is changed or all data bits are zero. 1 = Right channel zero cross is detected 0 = No zero cross Software can write 1 to clear this bit to zero

[21]	**TXBUSY**	**Transmit Busy** This bit is clear to 0 when all data in transmit FIFO and shift buffer is shifted out. And set to 1 when 1st data is load to shift buffer. 1 = Transmit shift buffer is busy 0 = Transmit shift buffer is empty This bit is read only.
[20]	**TXEMPTY**	**Transmit FIFO empty** This bit reflect data word number in transmit FIFO is zero 1 = Empty 0 = Not empty This bit is read only.
[19]	**TXFULL**	**Transmit FIFO full** This bit reflect data word number in transmit FIFO is 8 1 = Full. 0 = Not full. This bit is read only
[18]	**TXTHF**	**Transmit FIFO threshold flag** When data word(s) in transmit FIFO is equal or lower than threshold value set in TXTH[2:0] the TXTHF bit becomes to 1. It keeps at 1 till TXFIFO_LEVEL[3:0] is higher than TXTH[1:0] after software write TXFIFO register. 1 = Data word(s) in FIFO is equal or lower than threshold level 0 = Data word(s) in FIFO is higher than threshold level This bit is read only
[17]	**TXOVF**	**Transmit FIFO overflow flag** Write data to transmit FIFO when it is full and this bit set to 1 1 = Overflow 0 = No overflow Software can write 1 to clear this bit to zero
[16]	**TXUDF**	**Transmit FIFO underflow flag** When transmit FIFO is empty and shift logic hardware read data from data FIFO causes this set to 1. 1 = Underflow 0 = No underflow Software can write 1 to clear this bit to zero
[15:13]	**Reserved**	Reserved
[12]	**RXEMPTY**	**Receive FIFO empty** This bit reflects data words number in receive FIFO is zero 1 = Empty 0 = Not empty This bit is read only.

[11]	**RXFULL**	**Receive FIFO full** This bit reflect data words number in receive FIFO is 8 1 = Full 0 = Not full This bit is read only.
[10]	**RXTHF**	**Receive FIFO threshold flag** When data word(s) in receive FIFO is equal or higher than threshold value set in RXTH[2:0] the RXTHF bit becomes to 1. It keeps at 1 till RXFIFO_LEVEL[3:0] less than RXTH[1:0] after software read RXFIFO register. 1 = Data word(s) in FIFO is equal or higher than threshold level 0 = Data word(s) in FIFO is lower than threshold level This bit is read only
[9]	**RXOVF**	**Receive FIFO overflow flag** When receive FIFO is full and receive hardware attempt write to data into receive FIFO then this bit is set to 1, data in 1st buffer is overwrote. 1 = Overflow occur 0 = No overflow occur Software can write 1 to clear this bit to zero
[8]	**RXUDF**	**Receive FIFO underflow flag** Read receive FIFO when it is empty, this bit set to 1 indicate underflow occur. 1 = Underflow occur 0 = No underflow occur Software can write 1 to clear this bit to zero
[7:4]	**Reserved**	Reserved
[3]	**RIGHT**	**Right channel** This bit indicate current transmit data is belong to right channel 1 = Right channel 0 = Left channel This bit is read only
[2]	**I2STXINT**	**I^{2}S transmit interrupt** 1 = Transmit interrupt 0 = No transmit interrupt This bit is read only
[1]	**I2SRXINT**	**I^{2}S receive interrupt** 1 = Receive interrupt 0 = No receive interrupt This bit is read only

[0]	**I2SINT**	**I^{2}S Interrupt flag** 1 = I^{2}S interrupt 0 = No I^{2}S interrupt It is wire-OR of I2STXINT and I2SRXINT bits. This bit is read only.

I^{2}S Transmit FIFO (I2S_TXFIFO)

Register	Offset	R/W	Description	Reset Value
I2S_TXFIFO	I2S_BA+0x10	R/W	I^{2}S Transmit FIFO	0x0000_0000

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

Bits	Descriptions	
[31:0]	**TXFIFO**	**Transmit FIFO register** I^{2}S contains 8 words (8x32 bit) data buffer for data transmit. Write data to this register to prepare data for transmit. The remain word number is indicated by TX_LEVEL[3:0] in I2S_STATUS

I^{2}S Receive FIFO (I2S_RXFIFO)

Register	Offset	R/W	Description	Reset Value
I2S_RXFIFO	I2S_BA+0x14	R/W	I^{2}S Receive FIFO	0x0000_0000

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

Bits	Descriptions	
[31:0]	**RXFIFO**	**Receive FIFO register** I^{2}S contains 8 words (8x32 bit) data buffer for data receive. Read this register to get data in FIFO. The remaining data word number is indicated by RX_LEVEL[3:0] in I2S_STATUS register.

5.16 Analog-to-Digital Converter (ADC)

5.16.1 Overview

NuMicro™ NUC100 Series contains one 12-bit successive approximation analog-to-digital converters (SAR A/D converter) with 8 input channels. The A/D converter supports three operation modes: single, single-cycle scan and continuous scan mode. The A/D converters can be started by software and external STADC pin.

5.16.2 Features

- Analog input voltage range: 0~Vref (Max to 5.0 V)
- 12-bit resolution and 10-bit accuracy is guaranteed
- Up to 8 single-end analog input channels or 4 differential analog input channels
- Maximum ADC clock frequency is 16 MHz
- Up to 700K SPS conversion rate
- Three operating modes
  - Single mode: A/D conversion is performed one time on a specified channel
  - Single-cycle scan mode: A/D conversion is performed one cycle on all specified channels with the sequence from the lowest numbered channel to the highest numbered channel
  - Continuous scan mode: A/D converter continuously performs Single-cycle scan mode until software stops A/D conversion
- An A/D conversion can be started by
  - Software write 1 to ADST bit
  - External pin STADC
- Conversion results are held in data registers for each channel with valid and overrun indicators
- Conversion result can be compared with specify value and user can select whether to generate an interrupt when conversion result is equal to the compare register setting
- Channel 7 supports 3 input sources: external analog voltage, internal bandgap voltage, and internal temperature sensor output
- Support Self-calibration to minimize conversion error

5.16.3 Block Diagram

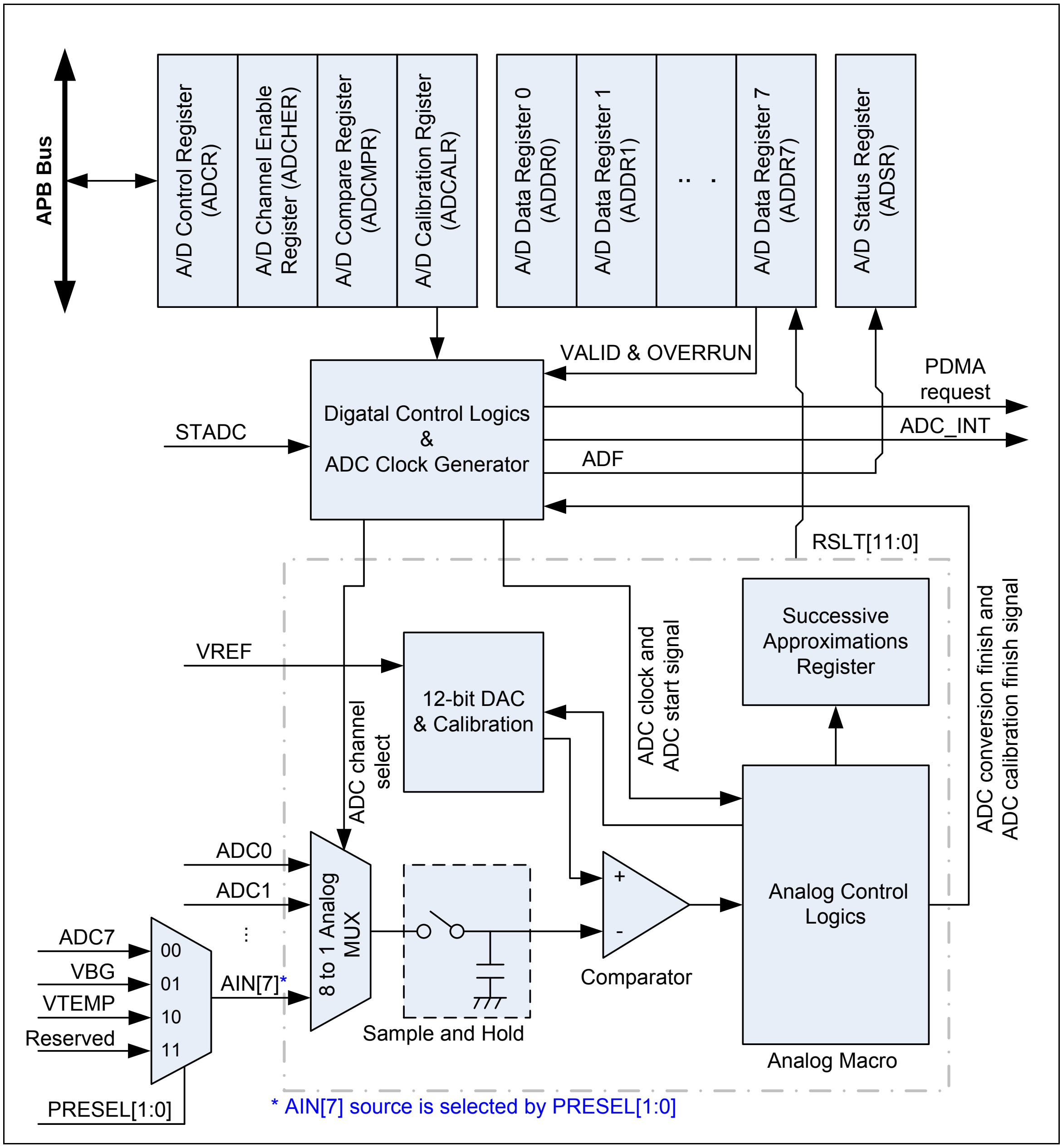


Figure 5-99 ADC Controller Block Diagram

5.16.4 Functional Description

The A/D converter operates by successive approximation with 12-bit resolution. This A/D converter equips with self calibration function to minimize conversion error, user can write 1 to CALEN bit in ADCALR register to enable calibration function, while internal calibration is finished the CAL_DONE bit will be set to 1 by hardware. The ADC has three operation modes: single mode, single-cycle scan mode and continuous scan mode. When changing the operating mode or analog input channel, in order to prevent incorrect operation, software must clear ADST bit to 0 in ADCR register.

5.16.4.1 Self-Calibration

When chip power on or switch ADC input type between single-end input and differential input, it needs to do ADC self calibration to minimize the conversion error. User can write 1 to CALEN bit of ADCALR register to start the self calibration. It needs 127 ADC clocks to complete the calibration and the CAL_DONE bit will be set to 1 by hardware. The detail timing is shown as below:

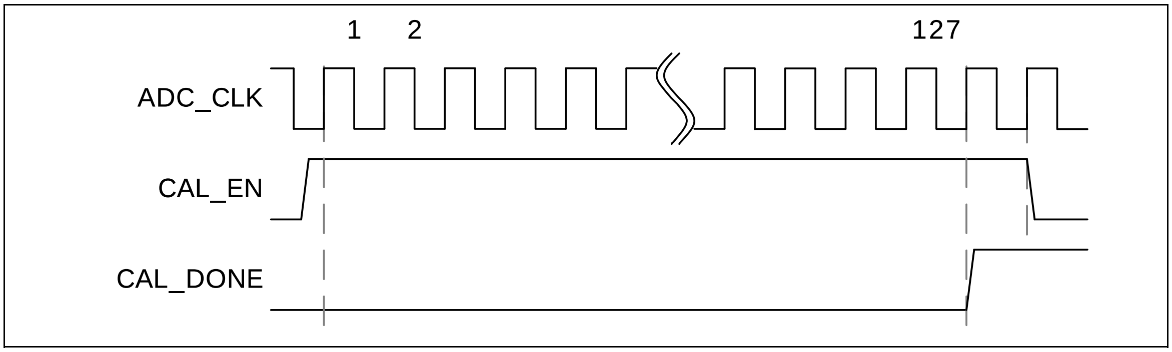


Figure 5-100 ADC Converter Self-Calibration Timing Diagram

5.16.4.2 ADC Clock Generator

The maximum sampling rate is up to 700K SPS. The ADC engine has four clock sources selected by 2-bit ADC_S (CLKSEL[3:2]), the ADC clock frequency is divided by an 8-bit prescaler with the formula:

The ADC clock frequency = (ADC clock source frequency) / (ADC_N+1);

where the 8-bit ADC_N is located in register CLKDIV[23:16].

If the clock source is from HCLK, the ADC_N can't be 0. In generally, software can set ADC_S and ADC_N to get 16 MHz or slightly less.

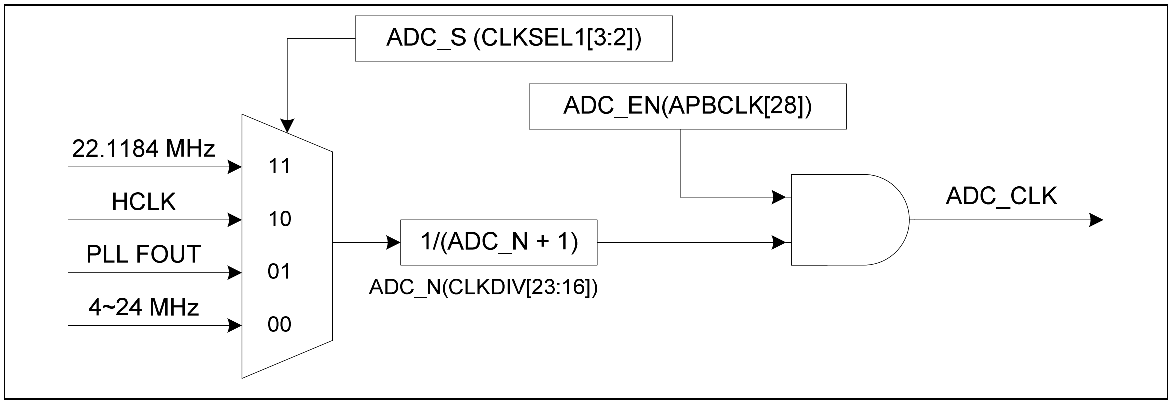


Figure 5-101 ADC Clock Control

5.16.4.3 Single Mode

In single mode, A/D conversion is performed only once on the specified single channel. The operations are as follows:

1. A/D conversion will be started when the ADST bit of ADCR is set to 1 by software or external trigger input.
2. When A/D conversion is finished, the result is stored in the A/D data register corresponding to the channel.
3. The ADF bit of ADSR register will be set to 1. If the ADIE bit of ADCR register is set to 1, the ADC interrupt will be asserted.
4. The ADST bit remains 1 during A/D conversion. When A/D conversion ends, the ADST bit is automatically cleared to 0 and the A/D converter enters idle state. Note that, after clearing the ADST bit, the ADST bit must be kept at 0 at least one ADC clock period before setting it to 1 again. If not, the A/D converter may not work.

Note: If software enables more than one channel in single mode, the channel with the lowest number will be selected and the other enabled channels will be ignored.

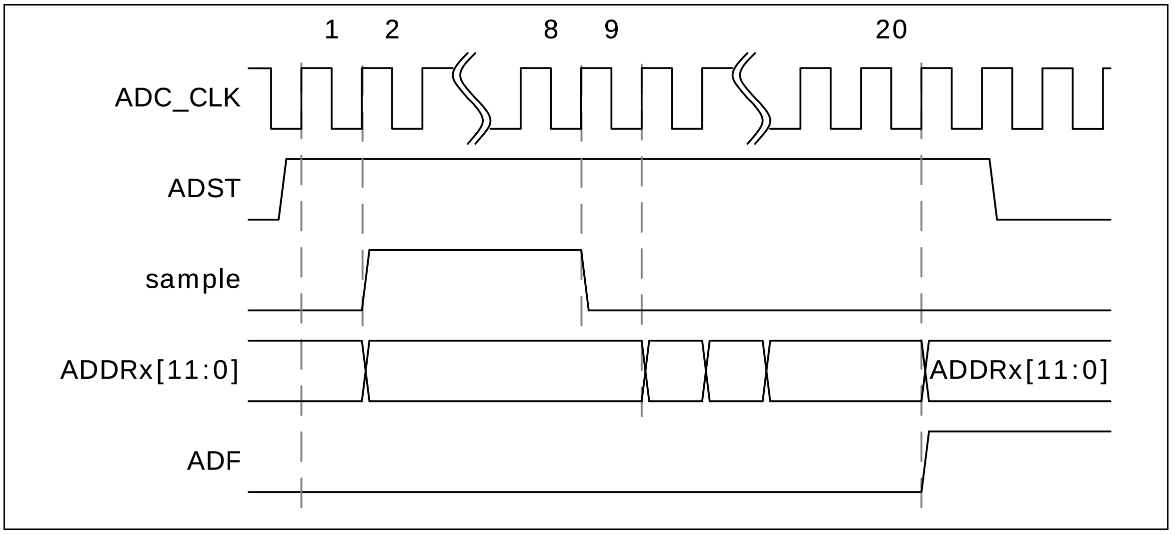


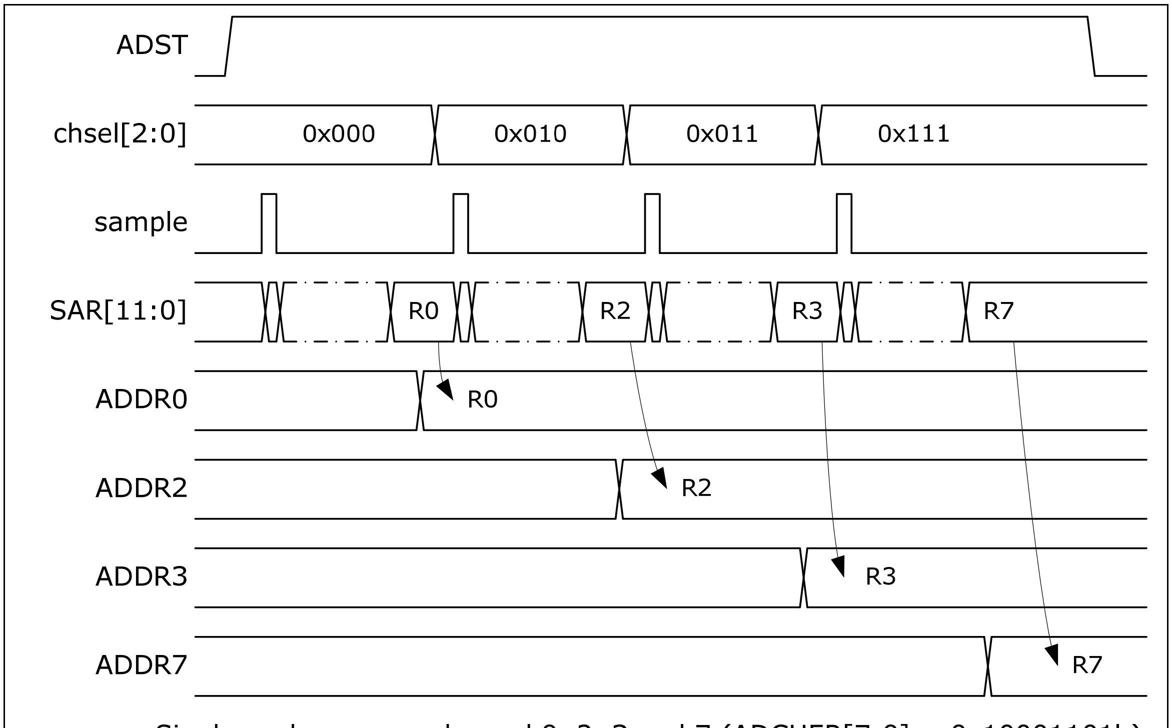
Figure 5-102 Single Mode Conversion Timing Diagram

5.16.4.4 Single-Cycle Scan Mode

In single-cycle scan mode, A/D conversion will sample and convert the specified channels once in the sequence from the lowest number enabled channel to the highest number enabled channel.

1. When the ADST bit of ADCR is set to 1 by software or external trigger input, A/D conversion starts on the channel with the lowest number.
2. When A/D conversion for each enabled channel is completed, the result is sequentially transferred to the A/D data register corresponding to each channel.
3. When the conversions of all the enabled channels are completed, the ADF bit in ADSR is set to 1. If the ADC interrupt function is enabled, the ADC interrupt occurs.
4. After A/D conversion ends, the ADST bit is automatically cleared to 0 and the A/D converter enters idle state. If ADST is cleared to 0 before all enabled ADC channels conversion done, ADC controller will finish current conversion and the result of the lowest enabled ADC channel will become unpredictable. Note that, after clearing the ADST bit to 0, the ADST bit must be kept at 0 at least one ADC clock period before setting it to 1 again. If not, the A/D converter may not work.

An example timing diagram for single-cycle scan on enabled channels (0, 2, 3 and 7) is shown as below:



Single-cycle scan on channel 0, 2, 3 and 7 (ADCHER[7:0] = 0x10001101b)

Figure 5-103 Single-Cycle Scan on Enabled Channels Timing Diagram

5.16.4.5 Continuous Scan Mode

In continuous scan mode, A/D conversion is performed sequentially on the specified channels that enabled by CHEN bits in ADCHER register (maximum 8 channels for ADC). The operations are as follows:

1. When the ADST bit in ADCR is set to 1 by software or external trigger input, A/D conversion starts on the channel with the lowest number.
2. When A/D conversion for each enabled channel is completed, the result of each enabled channel is stored in the A/D data register corresponding to each enabled channel.
3. When A/D converter completes the conversions of all enabled channels sequentially, the ADF bit (ADSR[0]) will be set to 1. If the ADC interrupt function is enabled, the ADC interrupt occurs. The conversion of the enabled channel with the lowest number will start again if software has not cleared the ADST bit.
4. As long as the ADST bit remains at 1, the step 2 ~ 3 will be repeated. When ADST is cleared to 0, ADC controller will finish current conversion and the result of the lowest enabled ADC channel will become unpredictable.

An example timing diagram for continuous scan on enabled channels (0, 2, 3 and 7) is shown as below:

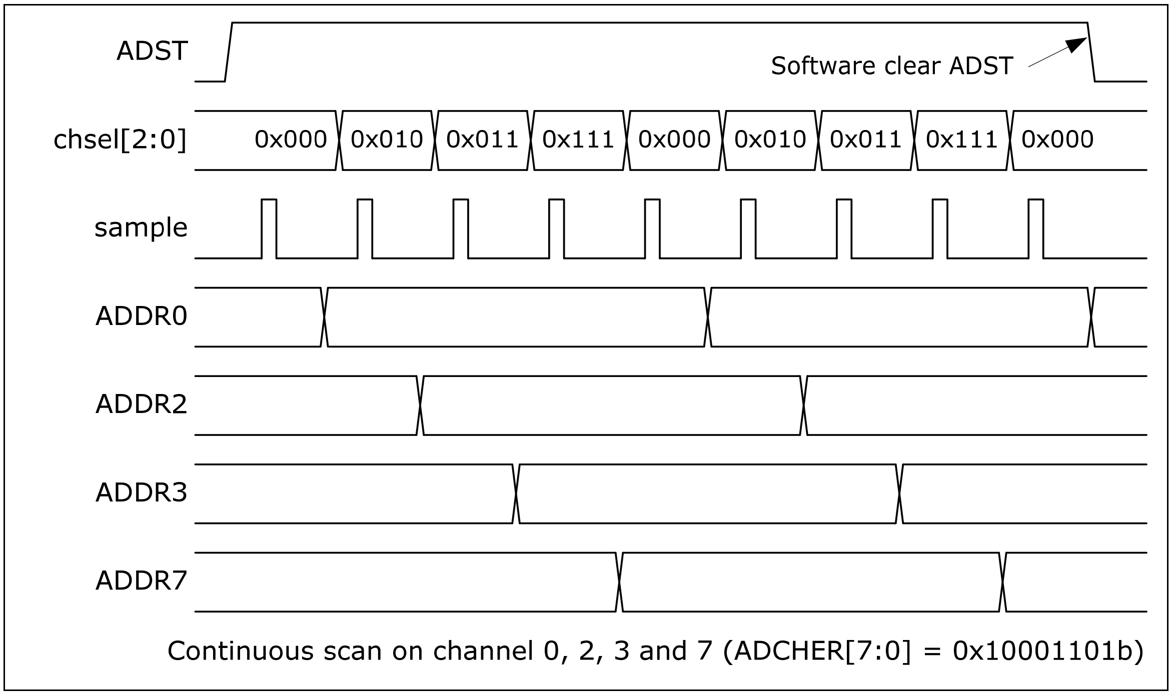


Figure 5-104 Continuous Scan on Enabled Channels Timing Diagram

5.16.4.6 External trigger Input Sampling and A/D Conversion Time

In single-cycle scan mode, A/D conversion can be triggered by external pin request. When the ADCR.TRGEN is set to high to enable ADC external trigger function, setting the TRGS[1:0] bits to 00b is to select external trigger input from the STADC pin. Software can set TRGCOND[1:0] to select trigger condition is falling/rising edge or low/high level. If level trigger condition is selected, the STADC pin must be kept at defined state at least 8 PCLKs. The ADST bit will be set to 1 at the 9th PCLK and start to conversion. Conversion is continuous if external trigger input is kept at active state in level trigger mode. It is stopped only when external condition trigger condition disappears. If edge trigger condition is selected, the high and low state must be kept at least 4 PLCKs. Pulse that is shorter than this specification will be ignored.

5.16.4.7 Conversion Result Monitor by Compare Function

ADC controller provide two sets of compare register ADCMPR0 and ADCMPR1, to monitor maximum two specified channels conversion result from A/D conversion controller, refer to Figure 5-105. Software can select which channel to be monitored by set CMPCH(ADCMPRx[5:0]) and CMPCOND bit is used to check conversion result is less than specify value or greater than (equal to) value specified in CMPD[11:0]. When the conversion of the channel specified by CMPCH is completed, the comparing action will be triggered one time automatically. When the compare result meets the setting, compare match counter will increase 1, otherwise, the compare match counter will be clear to 0. When counter value reach the setting of (CMPMATCNT+1) then CMPF bit will be set to 1, if CMPIE bit is set then an ADC_INT interrupt request is generated. Software can use it to monitor the external analog input pin voltage transition in scan mode without imposing a load on software. Detail logics diagram is shown as below:

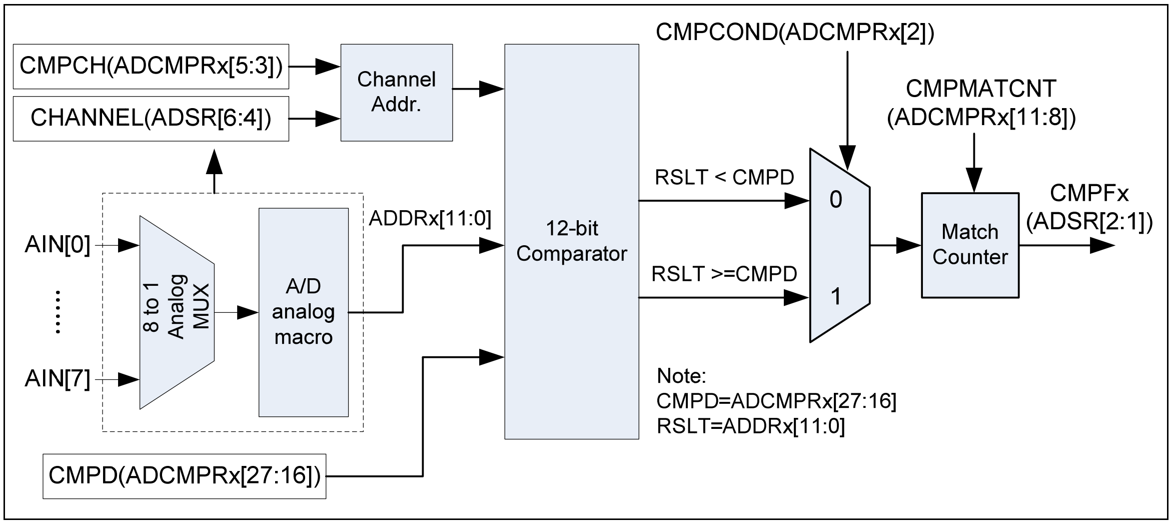


Figure 5-105 A/D Conversion Result Monitor Logics Diagram

5.16.4.8 *Interrupt Sources*

There are three interrupt sources of ADC interrupt. When an ADC operation mode finishes its conversion, the A/D conversion end flag, ADF, will be set to 1. The CMPF0 and CMPF1 are the compare flags of compare function. When the conversion result meets the settings of ADCMPR0/1, the corresponding flag will be set to 1. When one of the flags, ADF, CMPF0 and CMPF1, is set to 1 and the corresponding interrupt enable bit, ADIE of ADCR and CMPIE of ADCMPR0/1, is set to 1, the ADC interrupt will be asserted. Software can clear the flag to revoke the interrupt request.

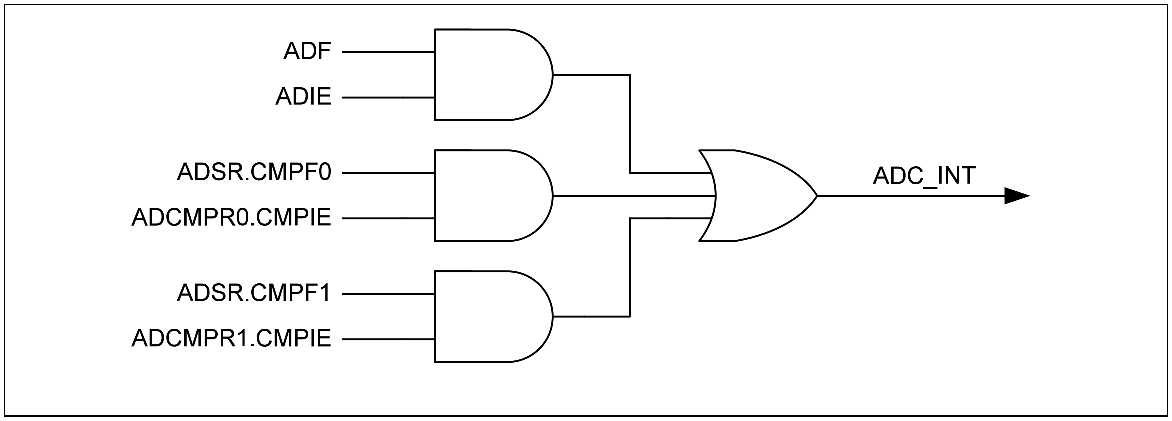


Figure 5-106 A/D Controller Interrupt

5.16.4.9 *Peripheral DMA Request*

When A/D conversion is finished, the conversion result will be loaded into ADDR register and VALID bit will be set to 1. If the PTEN bit of ADCR is set, ADC controller will generate a request to PDMA. User can use PDMA to transfer the conversion results to a user-specified memory space without CPU's intervention. The source address of PDMA operation is fixed at ADPDMA, no matter what channels was selected. When PDMA is transferring the conversion result, ADC will continue converting the next selected channel if the operation mode of ADC is single scan mode or continuous scan mode. User can monitor current PDMA transfer data through reading ADPDMA register. If ADC completes the conversion of a selected channel and the last conversion result of the same channel has not been transferred by PDMA, OVERUN bit of the corresponding channel will be set and the last ADC conversion result will be overwrite by the new ADC conversion result. PDMA will transfer the latest data of selected channels to the user-specified destination address.

5.16.5 Register Map

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

Register	Offset	R/W	Description	Reset Value
ADC_BA = 0x400E_0000				
ADDR0	ADC_BA+0x00	R	A/D Data Register 0	0x0000_0000
ADDR1	ADC_BA+0x04	R	A/D Data Register 1	0x0000_0000
ADDR2	ADC_BA+0x08	R	A/D Data Register 2	0x0000_0000
ADDR3	ADC_BA+0x0C	R	A/D Data Register 3	0x0000_0000
ADDR4	ADC_BA+0x10	R	A/D Data Register 4	0x0000_0000
ADDR5	ADC_BA+0x14	R	A/D Data Register 5	0x0000_0000
ADDR6	ADC_BA+0x18	R	A/D Data Register 6	0x0000_0000
ADDR7	ADC_BA+0x1C	R	A/D Data Register 7	0x0000_0000
ADCR	ADC_BA+0x20	R/W	A/D Control Register	0x0000_0000
ADCHER	ADC_BA+0x24	R/W	A/D Channel Enable Register	0x0000_0000
ADCMPR0	ADC_BA+0x28	R/W	A/D Compare Register 0	0x0000_0000
ADCMPR1	ADC_BA+0x2C	R/W	A/D Compare Register 1	0x0000_0000
ADSR	ADC_BA+0x30	R/W	A/D Status Register	0x0000_0000
ADCALR	ADC_BA+0x34	R/W	A/D Calibration Register	0x0000_0000
ADPDMA	ADC_BA+0x40	R	ADC PDMA current transfer data	0x0000_0000

5.16.6 Register Description

A/D Data Registers (ADDR0 ~ ADDR7)

Register	Offset	R/W	Description	Reset Value
ADDR0	ADC_BA+0x00	R	A/D Data Register 0	0x0000_0000
ADDR1	ADC_BA+0x04	R	A/D Data Register 1	0x0000_0000
ADDR2	ADC_BA+0x08	R	A/D Data Register 2	0x0000_0000
ADDR3	ADC_BA+0x0C	R	A/D Data Register 3	0x0000_0000
ADDR4	ADC_BA+0x10	R	A/D Data Register 4	0x0000_0000
ADDR5	ADC_BA+0x14	R	A/D Data Register 5	0x0000_0000
ADDR6	ADC_BA+0x18	R	A/D Data Register 6	0x0000_0000
ADDR7	ADC_BA+0x1C	R	A/D Data Register 7	0x0000_0000

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

Bits	Descriptions	
[31:18]	**Reserved**	Reserved
[17]	**VALID**	**Valid Flag** 1 = Data in RSLT[15:0] bits is valid 0 = Data in RSLT[15:0] bits is not valid This bit is set to 1 when corresponding channel analog input conversion is completed and cleared by hardware after ADDR register is read. This is a read only bit

[16]	**OVERRUN**	**Over Run Flag** 1 = Data in RSLT[15:0] is overwrite 0 = Data in RSLT[15:0] is recent conversion result If converted data in RSLT[15:0] has not been read before new conversion result is loaded to this register, OVERRUN is set to 1 and previous conversion result is gone. It is cleared by hardware after ADDR register is read. This is a read only bit
[15:0]	**RSLT**	**A/D Conversion Result** This field contains conversion result of ADC. When DMOF bit (ADCR[31]) set to 0, 12-bit ADC conversion result with unsigned format will be filled in RSLT[11:0] and zero will be filled in RSLT[15:12]. When DMOF bit (ADCR[31]) set to 1, 12-bit ADC conversion result with 2'complement format will be filled in RSLT[11:0] and signed bits to will be filled in RSLT[15:12].

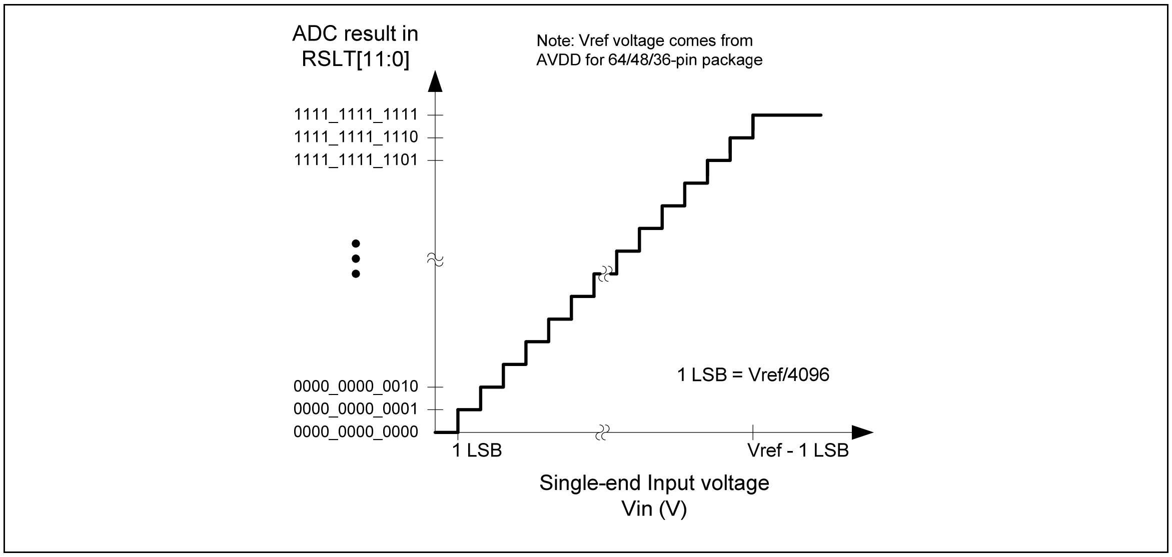


Figure 5-107 ADC single-end input conversion voltage and conversion result mapping diagram

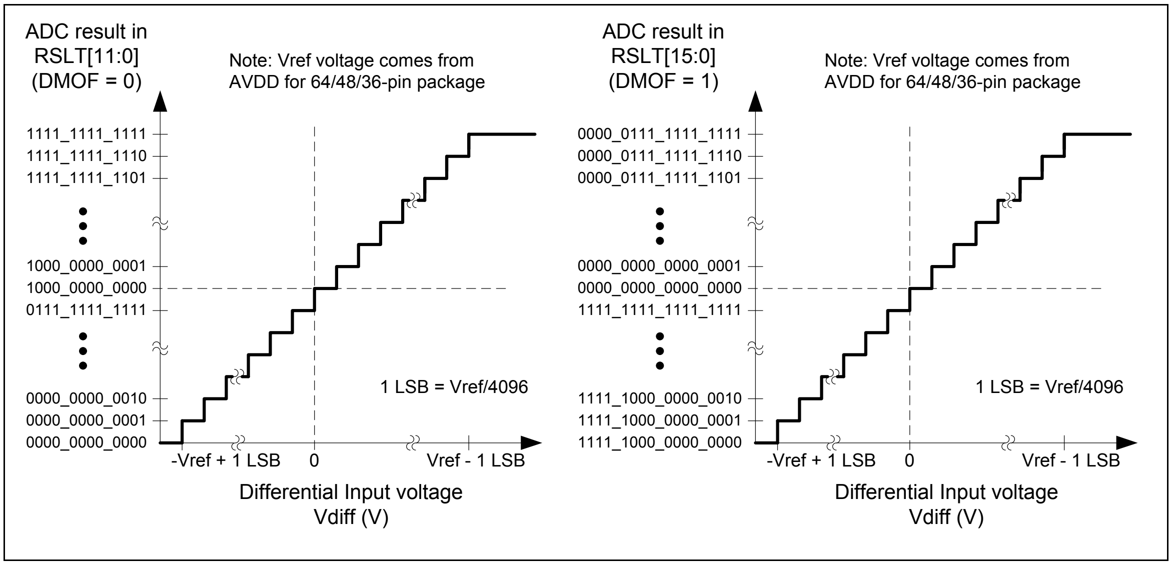


Figure 5-108 ADC differential input conversion voltage and conversion result mapping diagram

A/D Control Register (ADCR)

Register	Offset	R/W	Description	Reset Value
ADCR	ADC_BA+0x20	R/W	ADC Control Register	0x0000_0000

31	30	29	28	27	26	25	24
DMOF	**Reserved**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**ADST**	**DIFFEN**	**PTEN**	**TRGEN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TRGCOND		**TRGS**		**ADMD**		**ADIE**	**ADEN**

Bits	Descriptions	
[31]	**DMOF**	**A/D differential input Mode Output Format** 1 = A/D Conversion result will be filled in RSLT at ADDRx registers with 2'complement format. 0 = A/D Conversion result will be filled in RSLT at ADDRx registers with unsigned format.
[30:12]	**Reserved**	Reserved
[11]	**ADST**	**A/D Conversion Start** 1 = Conversion start 0 = Conversion stopped and A/D converter enter idle state ADST bit can be set to 1 from two sources: software and external pin STADC. ADST will be cleared to 0 by hardware automatically at the ends of single mode and single-cycle scan mode. In continuous scan mode, A/D conversion is continuously performed until software write 0 to this bit or chip reset.

<table>
<tr><td>[10]</td><td>DIFFEN</td><td>Differential Input Mode Enable
1 = Differential analog input mode
0 = Single-end analog input mode
<table>
<tr><th rowspan="2">Differential input paired channel</th><th colspan="2">ADC analog input</th></tr>
<tr><th>V_{plus}</th><th>V_{minus}</th></tr>
<tr><td>0</td><td>ADC0</td><td>ADC1</td></tr>
<tr><td>1</td><td>ADC2</td><td>ADC3</td></tr>
<tr><td>2</td><td>ADC4</td><td>ADC5</td></tr>
<tr><td>3</td><td>ADC6</td><td>ADC7</td></tr>
</table>
Differential input voltage (V_{diff}) = V_{plus} - V_{minus}, where V_{plus} is the analog input; V_{minus} is the inverted analog input.
In differential input mode, only the even number of the two corresponding channels needs to be enabled in ADCHER. The conversion result will be placed to the corresponding data register of the enabled channel.</td></tr>
<tr><td>[9]</td><td>PTEN</td><td>PDMA Transfer Enable
1 = Enable PDMA data transfer in ADDR 0~7
0 = Disable PDMA data transfer
When A/D conversion is completed, the converted data is loaded into ADDR 0~7, software can enable this bit to generate a PDMA data transfer request.
When PTEN=1, software must set ADIE=0 to disable interrupt.</td></tr>
<tr><td>[8]</td><td>TRGEN</td><td>External Trigger Enable
Enable or disable triggering of A/D conversion by external STADC pin.
1= Enable
0= Disable
ADC external trigger function is only supported in single-cycle scan mode.</td></tr>
<tr><td>[7:6]</td><td>TRGCOND</td><td>External Trigger Condition
These two bits decide external pin STADC trigger event is level or edge. The signal must be kept at stable state at least 8 PCLKs for level trigger and 4 PCLKs at high and low state for edge trigger.
00 = Low level
01 = High level
10 = Falling edge
11 = Rising edge</td></tr>
<tr><td>[5:4]</td><td>TRGS</td><td>Hardware Trigger Source
00 = A/D conversion is started by external STADC pin.
Others = Reserved
Software should disable TRGEN and ADST before change TRGS.
In hardware trigger mode, the ADST bit is set by the external trigger from STADC.</td></tr>
</table>

[3:2]	**ADMD**	**A/D Converter Operation Mode** 00 = Single conversion 01 = Reserved 10 = Single-cycle scan 11 = Continuous scan When changing the operation mode, software should disable ADST bit firstly.
[1]	**ADIE**	**A/D Interrupt Enable** 1 = Enable A/D interrupt function 0 = Disable A/D interrupt function A/D conversion end interrupt request is generated if ADIE bit is set to 1.
[0]	**ADEN**	**A/D Converter Enable** 1 = Enable 0 = Disable Before starting A/D conversion function, this bit should be set to 1. Clear it to 0 to disable A/D converter analog circuit for saving power consumption.

A/D Channel Enable Register (ADCHER)

Register	Offset	R/W	Description	Reset Value
ADCHER	ADC_BA+0x24	R/W	A/D Channel Enable	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						**PRESEL[1:0]**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CHEN7	**CHEN6**	**CHEN5**	**CHEN4**	**CHEN3**	**CHEN2**	**CHEN1**	**CHEN0**

Bits	Descriptions	
[31:10]	**Reserved**	Reserved
[9:8]	**PRESEL**	**Analog Input Channel 7 select** 00 = External analog input 01 = Internal bandgap voltage 10 = Internal temperature sensor 11 = Reserved Note: When software select the bandgap voltage as the analog input source of ADC channel 7, ADC clock rate needs to be limited to lower than 300 KHz.
[7]	**CHEN7**	**Analog Input Channel 7 Enable** 1 = Enable 0 = Disable
[6]	**CHEN6**	**Analog Input Channel 6 Enable** 1 = Enable 0 = Disable
[5]	**CHEN5**	**Analog Input Channel 5 Enable** 1 = Enable 0 = Disable
[4]	**CHEN4**	**Analog Input Channel 4 Enable** 1 = Enable 0 = Disable

[3]	**CHEN3**	**Analog Input Channel 3 Enable** 1 = Enable 0 = Disable
[2]	**CHEN2**	**Analog Input Channel 2 Enable** 1 = Enable 0 = Disable
[1]	**CHEN1**	**Analog Input Channel 1 Enable** 1 = Enable 0 = Disable
[0]	**CHEN0**	**Analog Input Channel 0 Enable** 1 = Enable 0 = Disable

A/D Compare Register 0/1 (ADCMPR0/1)

Register	Offset	R/W	Description	Reset Value
ADCMPR0	ADC_BA+0x28	R/W	A/D Compare Register 0	0x0000_0000
ADCMPR1	ADC_BA+0x2C	R/W	A/D Compare Register 1	0x0000_0000

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

Bits	Descriptions	
[31:28]	**Reserved**	Reserved
[27:16]	**CMPD**	**Comparison Data** The 12-bit data is used to compare with conversion result of specified channel. When DMOF bit is set to 0, ADC comparator compares CMPD with conversion result with unsigned format. CMPD should be filled in unsigned format. When DMOF bit is set to 1, ADC comparator compares CMPD with conversion result with 2'complement format. CMPD should be filled in 2'complement format.
[15:12]	**Reserved**	Reserved
[11:8]	**CMPMATCNT**	**Compare Match Count** When the specified A/D channel analog conversion result matches the compare condition defined by CMPCOND[2], the internal match counter will increase 1. When the internal counter reaches the value to (CMPMATCNT +1), the CMPFx bit will be set.
[7:6]	**Reserved**	Reserved
[5:3]	**CMPCH**	**Compare Channel Selection** 000 = Channel 0 conversion result is selected to be compared 001 = Channel 1 conversion result is selected to be compared 010 = Channel 2 conversion result is selected to be compared 011 = Channel 3 conversion result is selected to be compared 100 = Channel 4 conversion result is selected to be compared 101 = Channel 5 conversion result is selected to be compared 110 = Channel 6 conversion result is selected to be compared 111 = Channel 7 conversion result is selected to be compared

[2]	**CMPCOND**	**Compare Condition** 1 = Set the compare condition as that when a 12-bit A/D conversion result is greater or equal to the 12-bit CMPD (ADCMPRx[27:16]), the internal match counter will increase one. 0 = Set the compare condition as that when a 12-bit A/D conversion result is less than the 12-bit CMPD (ADCMPRx[27:16]), the internal match counter will increase one. Note: When the internal counter reaches the value to (CMPMATCNT +1), the CMPFx bit will be set.
[1]	**CMPIE**	**Compare Interrupt Enable** 1 = Enable compare function interrupt 0 = Disable compare function interrupt If the compare function is enabled and the compare condition matches the setting of CMPCOND and CMPMATCNT, CMPF bit will be asserted, in the meanwhile, if CMPIE is set to 1, a compare interrupt request is generated.
[0]	**CMPEN**	**Compare Enable** 1 = Enable compare function 0 = Disable compare function Set this bit to 1 to enable ADC controller to compare CMPD[11:0] with specified channel conversion result when converted data is loaded into ADDR register.

A/D Status Register (ADSR)

Register	Offset	R/W	Description	Reset Value
ADSR	ADC_BA+0x30	R/W	ADC Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
OVERRUN							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
VALID							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**CHANNEL**			**BUSY**	**CMPF1**	**CMPF0**	**ADF**

Bits	Descriptions	
[31:24]	**Reserved**	Reserved
[23:16]	**OVERRUN**	**Over Run flag** It is a mirror to OVERRUN bit in ADDRx It is read only.
[15:8]	**VALID**	**Data Valid flag** It is a mirror of VALID bit in ADDRx It is read only.
[7]	**Reserved**	Reserved
[6:4]	**CHANNEL**	**Current Conversion Channel** This field reflects current conversion channel when BUSY=1. When BUSY=0, it shows number of the next converted channel. It is read only.
[3]	**BUSY**	**BUSY/IDLE** 1 = A/D converter is busy at conversion. 0 = A/D converter is in idle state. This bit is mirror of as ADST bit in ADCR. It is read only.
[2]	**CMPF1**	**Compare Flag** When the selected channel A/D conversion result meets setting condition in ADCMPR1 then this bit is set to 1. And it is cleared by writing 1 to self. 1 = Conversion result in ADDR meets ADCMPR1 setting 0 = Conversion result in ADDR does not meet ADCMPR1 setting

[1]	**CMPF0**	**Compare Flag** When the selected channel A/D conversion result meets setting condition in ADCMPR0 then this bit is set to 1. And it is cleared by writing 1 to self. 1 = Conversion result in ADDR meets ADCMPR0 setting 0 = Conversion result in ADDR does not meet ADCMPR0 setting
[0]	**ADF**	**A/D Conversion End Flag** A status flag that indicates the end of A/D conversion. ADF is set to 1 at these two conditions: 1. When A/D conversion ends in single mode 2. When A/D conversion ends on all specified channels in scan mode This flag can be cleared by writing 1 to self.

A/D Calibration Register (ADCALR)

Register	Offset	R/W	Description	Reset Value
ADCALR	ADC_BA+0x34	R/W	A/D Calibration 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						**CALDONE**	**CALEN**

Bits	Descriptions	
[31:2]	**Reserved**	Reserved
[1]	**CALDONE**	**Calibration is Done** 1 = A/D converter self calibration is done 0 = A/D converter has not been calibrated or calibration is in progress if CALEN bit is set. When 0 is written to CALEN bit, CALDONE bit is cleared by hardware immediately. It is a read only bit.
[0]	**CALEN**	**Self Calibration Enable** 1 = Enable self calibration 0 = Disable self calibration Software can set this bit to 1 enables A/D converter to do self calibration function. It needs 127 ADC clocks to complete calibration. This bit must be kept at 1 after CALDONE asserted. Clearing this bit will disable self calibration function.

A/D PDMA current transfer data Register (ADPDMA)

Register	Offset	R/W	Description	Reset Value
ADPDMA	ADC_BA+0x40	R	A/D 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							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**AD_PDMA[11:8]**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
AD_PDMA[7:0]							

Bits	Descriptions	
[31:12]	**Reserved**	Reserved
[11:0]	**AD_PDMA**	**ADC PDMA current transfer data register** When PDMA transferring, read this register can monitor current PDMA transfer data. This is a read only register.

5.17 Analog Comparator (CMP)

5.17.1 Overview

NuMicro™ NUC100 Series contains two comparators. The comparators can be used in a number of different configurations. The comparator output is a logical one when positive input greater than negative input, otherwise the output is a zero. Each comparator can be configured to cause an interrupt when the comparator output value changes. The block diagram is shown in Figure 5-109.

5.17.2 Features

- Analog input voltage range: 0~5.0 V
- Hysteresis function supported
- Two analog comparators with optional internal reference voltage input at negative end
- One interrupt vector for both comparators

5.17.3 Block Diagram

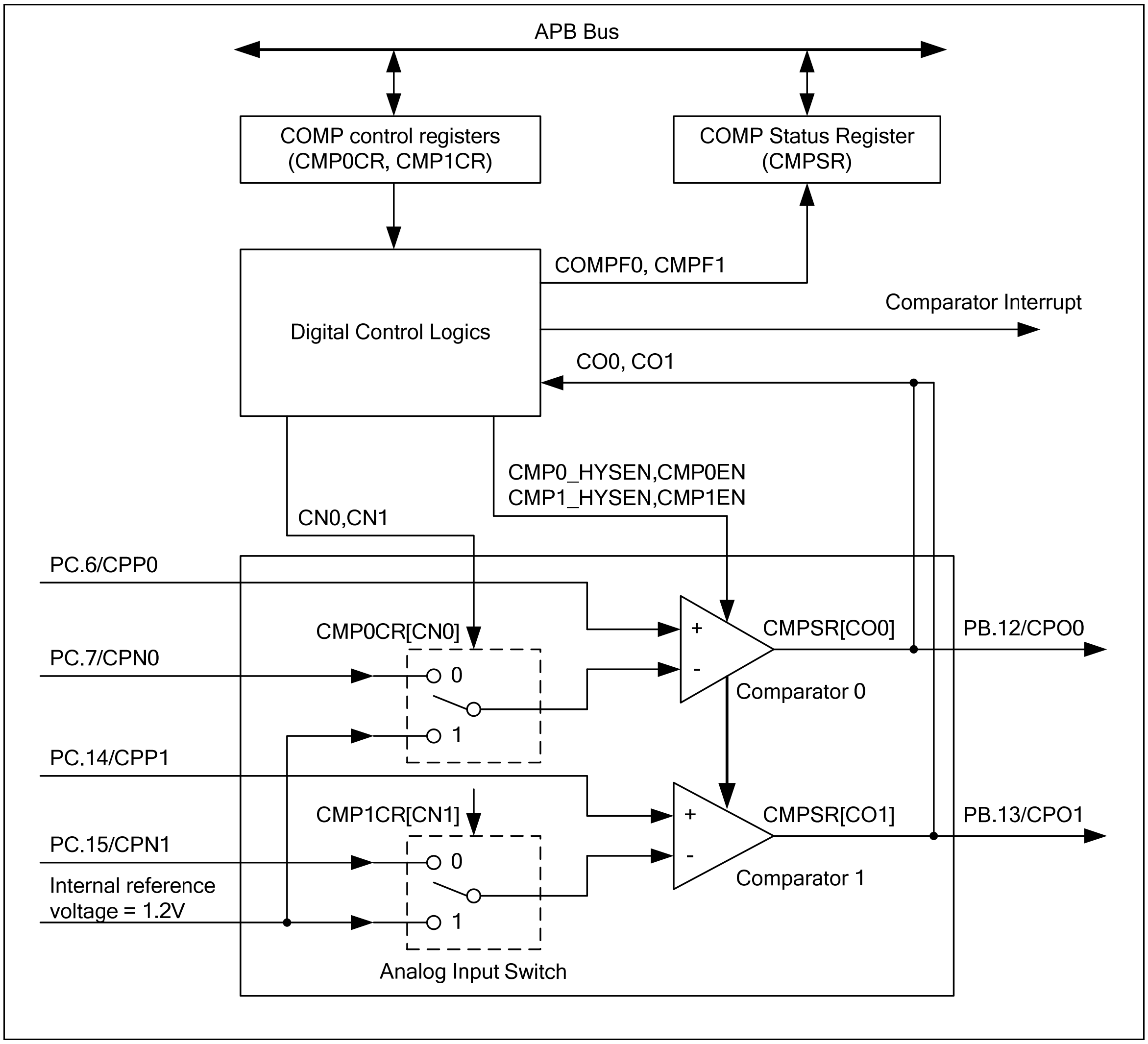


Figure 5-109 Analog Comparator Block Diagram

5.17.4 Functional Description

5.17.4.1 Interrupt Sources

The output of comparators are sampled by PCLK and reflected at CO1 and CO2 of CMPSR register. If CMP0IE/CMP1IE of CMP0CR/CMP1CR is set to 1, the comparator interrupt will be enabled. As the output state of comparator is changed, the comparator interrupt will be asserted and the corresponding flag, CMPF0 or CMPF1, will be set. Software can clear the flag to 0 by writing 1 to it.

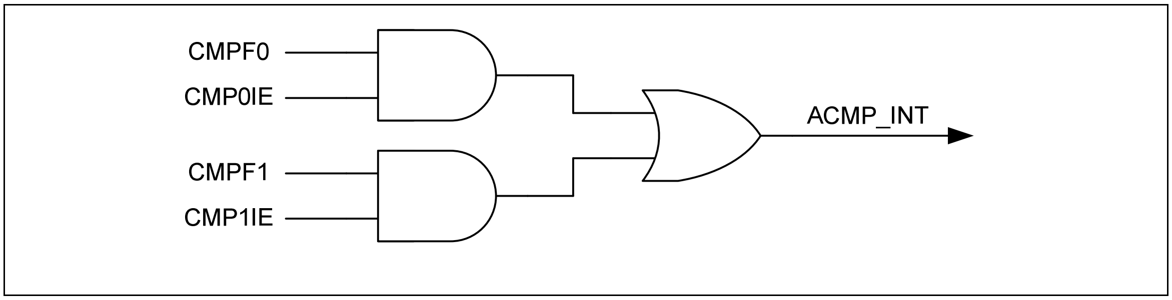


Figure 5-110 Comparator Controller Interrupt Sources

5.17.5 Register Map

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

Register	Offset	R/W	Description	Reset Value
CMP_BA = 0x400D_0000				
CMP0CR	CMP_BA+0x00	R/W	Comparator0 Control Register	0x0000_0000
CMP1CR	CMP_BA+0x04	R/W	Comparator1 Control Register	0x0000_0000
CMPSR	CMP_BA+0x08	R/W	Comparator Status Register	0x0000_0000

5.17.6 Register Description

CMP0 Control Register (CMP0CR)

Register	Offset	R/W	Description	Reset Value
CMP0CR	CMP_BA+0x00	R/W	Comparator0 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			**CN0**	**Reserved**	**CMP0_HYSEN**	**CMP0IE**	**CMP0EN**

Bits	Descriptions	
[31:5]	**Reserved**	Reserved
[4]	**CN0**	**Comparator0 negative input select** 1 = The internal bandgap reference voltage (V_{BG}=1.26 V) is selected as the source of negative comparator input 0 = The source of the negative comparator input is from CPN0 pin
[3]	**Reserved**	Reserved
[2]	**CMP0_HYSEN**	**Comparator0 Hysteresis Enable** 1 = Enable hysteresis function. The typical range is 20mV. 0 = Disable hysteresis function (Default).
[1]	**CMP0IE**	**Comparator0 Interrupt Enable** 1 = Enable interrupt function 0 = Disable interrupt function
[0]	**CMP0EN**	**Comparator0 Enable** 1 = Enable 0 = Disable Comparator output need wait 2 us stable time after CMP0EN is set.

CMP1 Control Register (CMP1CR)

Register	Offset	R/W	Description	Reset Value
CMP1CR	CMP_BA+0x04	R/W	Comparator1 Control Register	0x0000_0000

<table>
<tr><th>31</th><th>30</th><th>29</th><th>28</th><th>27</th><th>26</th><th>25</th><th>24</th></tr>
<tr><td colspan="8">Reserved</td></tr>
<tr><th>23</th><th>22</th><th>21</th><th>20</th><th>19</th><th>18</th><th>17</th><th>16</th></tr>
<tr><td colspan="8">Reserved</td></tr>
<tr><th>15</th><th>14</th><th>13</th><th>12</th><th>11</th><th>10</th><th>9</th><th>8</th></tr>
<tr><td colspan="8">Reserved</td></tr>
<tr><th>7</th><th>6</th><th>5</th><th>4</th><th>3</th><th>2</th><th>1</th><th>0</th></tr>
<tr><td colspan="3">Reserved</td><td>CN1</td><td>Reserved</td><td>CMP1_HYSEN</td><td>CMP1IE</td><td>CMP1EN</td></tr>
</table>

Bits	Descriptions	
[31:5]	**Reserved**	Reserved
[4]	**CN1**	**Comparator1 negative input select** 1 = The internal bandgap reference voltage (V_{BG}=1.26 V) is selected as the source of negative comparator input 0 = The source of the negative comparator input is from CPN1 pin
[3]	**Reserved**	Reserved
[2]	**CMP1_HYSEN**	**Comparator1 Hysteresis Enable** 1 = Enable hysteresis function. The typical range is 20mV. 0 = Disable hysteresis function (Default).
[1]	**CMP1IE**	**Comparator1 Interrupt Enable** 1 = Enable interrupt function 0 = Disable interrupt function
[0]	**CMP1EN**	**Comparator1 Enable** 1 = Enable 0 = Disable Comparator output need wait 2 us stable time after CMP1EN is set.

CMP Status Register (CMPSR)

Register	Offset	R/W	Description	Reset Value
CMPSR	CMP_BA+0x08	R/W	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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**CO1**	**CO0**	**CMPF1**	**CMPF0**

Bits	Descriptions	
[31:4]	**Reserved**	Reserved
[3]	**CO1**	**Comparator1 Output** Synchronized to the APB clock to allow reading by software. Cleared when the comparator is disabled (CMP1EN = 0).
[2]	**CO0**	**Comparator0 Output** Synchronized to the APB clock to allow reading by software. Cleared when the comparator is disabled (CMP0EN = 0).
[1]	**CMPF1**	**Comparator1 Flag** This bit is set by hardware whenever the comparator1 output changes state. This will cause an interrupt if CMP1IE set. Write 1 to clear this bit to zero.
[0]	**CMPF0**	**Comparator0 Flag** This bit is set by hardware whenever the comparator0 output changes state. This will cause an interrupt if CMP0IE set. Write 1 to clear this bit to zero.

5.18 PDMA Controller (PDMA)

5.18.1 Overview

NuMicro™ NUC130/NUC140 contains a peripheral direct memory access (PDMA) controller that transfers data to and from memory or transfer data to and from APB devices. The PDMA has nine channels of DMA (Peripheral-to-Memory or Memory-to-Peripheral or Memory-to-Memory). For each PDMA channel (PDMA CH0~CH8), there is one word buffer as transfer buffer between the Peripherals APB devices and Memory.

Software can stop the PDMA operation by disable PDMA [PDMACEN]. The CPU can recognize the completion of a PDMA operation by software polling or when it receives an internal PDMA interrupt. The PDMA controller can increase source or destination address or fixed them as well.

Notice: The partial of NuMicro™ NUC130/NUC140 only has 1 PDMA channel (channel 0).

5.18.2 Features

- Support nine DMA channels. Each channel can support a unidirectional transfer
- AMBA AHB master/slave interface compatible, for data transfer and register read/write
- Support source and destination address increased mode or fixed mode
- Hardware channel priority. DMA channel 0 has the highest priority and channel 8 has the lowest priority

5.18.3 Block Diagram

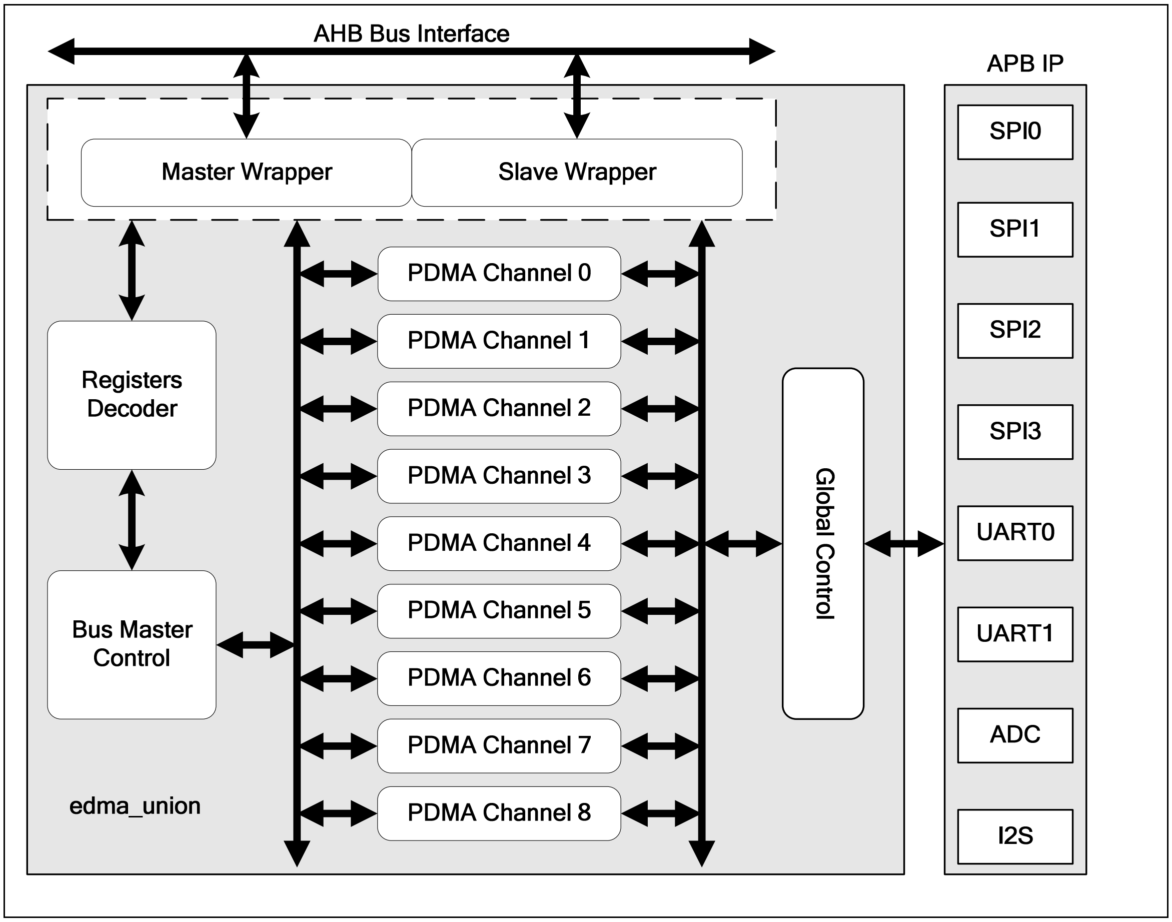


Figure 5-111 PDMA Controller Block Diagram

5.18.4 Function Description

The PDMA controller has nine channels of DMA associated with Peripheral-to-Memory、Memory-to-Peripheral or Memory-to-Memory. For each PDMA channel, there is one word memory as transfer buffer between the Peripherals APB IP and Memory.

The CPU can recognize the completion of a PDMA operation by software polling or when it receives an internal PDMA interrupt. As to the source and destination address, the PDMA controller has two modes: increased and fixed.

Every PDMA default channel behavior is not pre-defined, so users must configure the channel service settings of PDMA_PDSSR0, PDMA_PDSSR1 and PDMA_PDSSR2 before start the related PDMA channel.

Software must enable DMA channel PDMA [PDMACEN] and then write a valid source address to the PDMA_SARx register, a destination address to the PDMA_DARx register, and a transfer count to the PDMA_BCRx register. Next, trigger the DMA_CSRx PDMA [TRIG_EN]. PDMA will continue the transfer until PDMA_CBCRx comes down to zero, If an error occurs during the PDMA operation, the channel stops unless software clears the error condition and sets the PDMA_CSRx [SW_RST] to reset the PDMA channel and set PDMA_CSRx [PDMACEN] and [TRIG_EN] bits field to start again.

In PDMA (Peripheral-to-Memory or Memory-to-Peripheral) mode, DMA can transfer data between the Peripherals APB IP (ex: UART, SPI, ADC….) and Memory.

5.18.5 Register Map

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

Register	**Offset**	**R/W**	**Description**	**Reset Value**
PDMA_BA_ch0 = 0x5000_8000 PDMA_BA_ch1 = 0x5000_8100 PDMA_BA_ch2 = 0x5000_8200 **PDMA_BA_ch3 = 0x5000_8300 PDMA_BA_ch4 = 0x5000_8400 PDMA_BA_ch5 = 0x5000_8500** **PDMA_BA_ch6 = 0x5000_8600 PDMA_BA_ch7 = 0x5000_8700 PDMA_BA_ch8 = 0x5000_8800**				
PDMA_CSRx	PDMA_BA_chx+0x00	R/W	PDMA Control Register	0x0000_0000
PDMA_SARx	PDMA_BA_chx+0x04	R/W	PDMA Source Address Register	0x0000_0000
PDMA_DARx	PDMA_BA_chx+0x08	R/W	PDMA Destination Address Register	0x0000_0000
PDMA_BCRx	PDMA_BA_chx+0x0C	R/W	PDMA Transfer Byte Count Register	0x0000_0000
PDMA_POINTx	PDMA_BA_chx+0x10	R	PDMA Internal buffer pointer	0xXXXX_0000
PDMA_CSARx	PDMA_BA_chx+0x14	R	PDMA Current Source Address Register	0x0000_0000
PDMA_CDARx	PDMA_BA_chx+0x18	R	PDMA Current Destination Address Register	0x0000_0000
PDMA_CBCRx	PDMA_BA_chx+0x1C	R	PDMA Current Transfer Byte Count Register	0x0000_0000
PDMA_IERx	PDMA_BA_chx+0x20	R/W	PDMA Interrupt Enable Register	0x0000_0001
PDMA_ISRx	PDMA_BA_chx+0x24	R/W	PDMA Interrupt Status Register	0x0000_0000
PDMA_SBUF0_cx	PDMA_BA_chx+0x80	R	PDMA Shared Buffer FIFO 0	0x0000_0000
PDMA_BA_GCR = 0x5000_8F00				
PDMA_GCRCSR	PDMA_BA_GCR+0x00	R/W	PDMA Global Control Register	0x0000_0000
PDMA_PDSSR0	PDMA_BA_GCR+0x04	R/W	PDMA Service Selection Control Register 0	0XFFFF_FFFF
PDMA_PDSSR1	PDMA_BA_GCR+0x08	R/W	PDMA Service Selection Control Register 1	0xFFFF_FFFF
PDMA_GCRISR	PDMA_BA_GCR+0x0C	R/W	PDMA Global Interrupt Register	0x0000_0000
PDMA_PDSSR2	PDMA_BA_GCR+0x10	R/W	PDMA Service Selection Control Register 2	0x0000_00FF

5.18.6 Register Description

PDMA Control and Status Register (PDMA_CSRx)

Register	Offset	R/W	Description	Reset Value
PDMA_CSR0	PDMA_BA_ch0+0x00	R/W	PDMA Control and Status Register CH0	0x0000_0000
PDMA_CSR1	PDMA_BA_ch1+0x00	R/W	PDMA Control and Status Register CH1	0x0000_0000
PDMA_CSR2	PDMA_BA_ch2+0x00	R/W	PDMA Control and Status Register CH2	0x0000_0000
PDMA_CSR3	PDMA_BA_ch3+0x00	R/W	PDMA Control and Status Register CH3	0x0000_0000
PDMA_CSR4	PDMA_BA_ch4+0x00	R/W	PDMA Control and Status Register CH4	0x0000_0000
PDMA_CSR5	PDMA_BA_ch5+0x00	R/W	PDMA Control and Status Register CH5	0x0000_0000
PDMA_CSR6	PDMA_BA_ch6+0x00	R/W	PDMA Control and Status Register CH6	0x0000_0000
PDMA_CSR7	PDMA_BA_ch7+0x00	R/W	PDMA Control and Status Register CH7	0x0000_0000
PDMA_CSR8	PDMA_BA_ch8+0x00	R/W	PDMA Control and Status Register CH8	0x0000_0000

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

Bits	Descriptions	
[31:24]	**Reserved**	Reserved
[23]	**TRIG_EN**	**TRIG_EN** 1 = Enable PDMA data read or write transfer. 0 = No effect. Note: When PDMA transfer completed, this bit will be cleared automatically. If the bus error occurs, all PDMA transfer will be stopped. Software must reset all PDMA channel, and then trigger again.
[22:21]	**Reserved**	Reserved

[20:19]	**APB_TWS**	**Peripheral transfer Width Select** 00 = One word (32-bit) is transferred for every PDMA operation. 01 = One byte (8-bit) is transferred for every PDMA operation. 10 = One half-word (16-bit) is transferred for every PDMA operation. 11 = Reserved. Note: This field is meaningful only when MODE_SEL is Peripheral to Memory mode (Peripheral-to-Memory) or Memory to Peripheral mode (Memory-to-Peripheral).
[18:8]	**Reserved**	Reserved
[7:6]	**DAD_SEL**	**Transfer Destination Address Direction Select** 00 = Transfer Destination address is increasing successively. 01 = Reserved. 10 = Transfer Destination address is fixed (This feature can be used when data where transferred from multiple sources to a single destination). 11 = Reserved.
[5:4]	**SAD_SEL**	**Transfer Source Address Direction Select** 00 = Transfer Source address is increasing successively. 01 = Reserved. 10 = Transfer Source address is fixed (This feature can be used when data where transferred from a single source to multiple destinations). 11 = Reserved.
[3:2]	**MODE_SEL**	**PDMA Mode Select** 00 = Memory to Memory mode (Memory-to-Memory). 01 = Peripheral to Memory mode (Peripheral-to-Memory). 10 = Memory to Peripheral mode (Memory-to-Peripheral).
[1]	**SW_RST**	**Software Engine Reset** 0 = Writing 0 to this bit has no effect. 1 = Writing 1 to this bit will reset the internal state machine, pointers and internal buffer. The contents of control register will not be cleared. This bit will auto clear after few clock cycles.
[0]	**PDMACEN**	**PDMA Channel Enable** Setting this bit to 1 enables PDMA's operation. If this bit is cleared, PDMA will ignore all PDMA request and force Bus Master into IDLE state. Note: SW_RST(PDMA_CSRx[1], x= 0~8) will clear this bit

PDMA Transfer Source Address Register (PDMA_SARx)

Register	Offset	R/W	Description	Reset Value
PDMA_SAR0	PDMA_BA_ch0+0x04	R/W	PDMA Transfer Source Address Register CH0	0x0000_0000
PDMA_SAR1	PDMA_BA_ch1+0x04	R/W	PDMA Transfer Source Address Register CH1	0x0000_0000
PDMA_SAR2	PDMA_BA_ch2+0x04	R/W	PDMA Transfer Source Address Register CH2	0x0000_0000
PDMA_SAR3	PDMA_BA_ch3+0x04	R/W	PDMA Transfer Source Address Register CH3	0x0000_0000
PDMA_SAR4	PDMA_BA_ch4+0x04	R/W	PDMA Transfer Source Address Register CH4	0x0000_0000
PDMA_SAR5	PDMA_BA_ch5+0x04	R/W	PDMA Transfer Source Address Register CH5	0x0000_0000
PDMA_SAR6	PDMA_BA_ch6+0x04	R/W	PDMA Transfer Source Address Register CH6	0x0000_0000
PDMA_SAR7	PDMA_BA_ch7+0x04	R/W	PDMA Transfer Source Address Register CH7	0x0000_0000
PDMA_SAR8	PDMA_BA_ch8+0x04	R/W	PDMA Transfer Source Address Register CH8	0x0000_0000

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

Bits	Descriptions	
[31:0]	**PDMA_SAR**	**PDMA Transfer Source Address Register** This field indicates a 32-bit source address of PDMA. Note : The source address must be word alignment

PDMA Transfer Destination Address Register (PDMA_DARx)

Register	Offset	R/W	Description	Reset Value
PDMA_DAR0	PDMA_BA_ch0+0x08	R/W	PDMA Transfer Destination Address Register CH0	0x0000_0000
PDMA_DAR1	PDMA_BA_ch1+0x08	R/W	PDMA Transfer Destination Address Register CH1	0x0000_0000
PDMA_DAR2	PDMA_BA_ch2+0x08	R/W	PDMA Transfer Destination Address Register CH2	0x0000_0000
PDMA_DAR3	PDMA_BA_ch3+0x08	R/W	PDMA Transfer Destination Address Register CH3	0x0000_0000
PDMA_DAR4	PDMA_BA_ch40x08	R/W	PDMA Transfer Destination Address Register CH4	0x0000_0000
PDMA_DAR5	PDMA_BA_ch5+0x08	R/W	PDMA Transfer Destination Address Register CH5	0x0000_0000
PDMA_DAR6	PDMA_BA_ch6+0x08	R/W	PDMA Transfer Destination Address Register CH6	0x0000_0000
PDMA_DAR7	PDMA_BA_ch7+0x08	R/W	PDMA Transfer Destination Address Register CH7	0x0000_0000
PDMA_DAR8	PDMA_BA_ch8+0x08	R/W	PDMA Transfer Destination Address Register CH8	0x0000_0000

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

Bits	Descriptions	
[31:0]	**PDMA_DAR**	**PDMA Transfer Destination Address Register** This field indicates a 32-bit destination address of PDMA. Note : The destination address must be word alignment

PDMA Transfer Byte Count Register (PDMA_BCRx)

Register	Offset	R/W	Description	Reset Value
PDMA_BCR0	PDMA_BA_ch0+0x0C	R/W	PDMA Transfer Byte Count Register CH0	0x0000_0000
PDMA_BCR1	PDMA_BA_ch1+0x0C	R/W	PDMA Transfer Byte Count Register CH1	0x0000_0000
PDMA_BCR2	PDMA_BA_ch2+0x0C	R/W	PDMA Transfer Byte Count Register CH2	0x0000_0000
PDMA_BCR3	PDMA_BA_ch3+0x0C	R/W	PDMA Transfer Byte Count Register CH3	0x0000_0000
PDMA_BCR4	PDMA_BA_ch4+0x0C	R/W	PDMA Transfer Byte Count Register CH4	0x0000_0000
PDMA_BCR5	PDMA_BA_ch5+0x0C	R/W	PDMA Transfer Byte Count Register CH5	0x0000_0000
PDMA_BCR6	PDMA_BA_ch6+0x0C	R/W	PDMA Transfer Byte Count Register CH6	0x0000_0000
PDMA_BCR7	PDMA_BA_ch7+0x0C	R/W	PDMA Transfer Byte Count Register CH7	0x0000_0000
PDMA_BCR8	PDMA_BA_ch8+0x0C	R/W	PDMA Transfer Byte Count Register CH8	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**
PDMA_BCR [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PDMA_BCR [7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	Reserved
[15:0]	**PDMA_BCR**	**PDMA Transfer Byte Count Register** This field indicates a 16-bit transfer byte count number of PDMA, it must be word alignment.

PDMA Internal Buffer Pointer Register (PDMA_POINTx)

Register	Offset	R/W	Description	Reset Value
PDMA_POINT0	PDMA_BA_ch0+0x10	R	PDMA Internal Buffer Pointer Register CH0	0xXXXX_0000
PDMA_POINT1	PDMA_BA_ch1+0x10	R	PDMA Internal Buffer Pointer Register CH1	0xXXXX_0000
PDMA_POINT2	PDMA_BA_ch2+0x10	R	PDMA Internal Buffer Pointer Register CH2	0xXXXX_0000
PDMA_POINT3	PDMA_BA_ch3+0x10	R	PDMA Internal Buffer Pointer Register CH3	0xXXXX_0000
PDMA_POINT4	PDMA_BA_ch4+0x10	R	PDMA Internal Buffer Pointer Register CH4	0xXXXX_0000
PDMA_POINT5	PDMA_BA_ch5+0x10	R	PDMA Internal Buffer Pointer Register CH5	0xXXXX_0000
PDMA_POINT6	PDMA_BA_ch6+0x10	R	PDMA Internal Buffer Pointer Register CH6	0xXXXX_0000
PDMA_POINT7	PDMA_BA_ch7+0x10	R	PDMA Internal Buffer Pointer Register CH7	0xXXXX_0000
PDMA_POINT8	PDMA_BA_ch8+0x10	R	PDMA Internal Buffer Pointer Register CH8	0xXXXX_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				**PDMA_POINT**			

Bits	Descriptions	
[31:2]	**Reserved**	Reserved
[1:0]	**PDMA_POINT**	**PDMA Internal Buffer Pointer Register (Read Only)** This field indicates the internal buffer pointer.

PDMA Current Source Address Register (PDMA_CSARx)

Register	Offset	R/W	Description	Reset Value
PDMA_CSAR0	PDMA_BA_ch0+0x14	R	PDMA Current Source Address Register CH0	0x0000_0000
PDMA_CSAR1	PDMA_BA_ch1+0x14	R	PDMA Current Source Address Register CH1	0x0000_0000
PDMA_CSAR2	PDMA_BA_ch2+0x14	R	PDMA Current Source Address Register CH2	0x0000_0000
PDMA_CSAR3	PDMA_BA_ch3+0x14	R	PDMA Current Source Address Register CH3	0x0000_0000
PDMA_CSAR4	PDMA_BA_ch4+0x14	R	PDMA Current Source Address Register CH4	0x0000_0000
PDMA_CSAR5	PDMA_BA_ch5+0x14	R	PDMA Current Source Address Register CH5	0x0000_0000
PDMA_CSAR6	PDMA_BA_ch6+0x14	R	PDMA Current Source Address Register CH6	0x0000_0000
PDMA_CSAR7	PDMA_BA_ch7+0x14	R	PDMA Current Source Address Register CH7	0x0000_0000
PDMA_CSAR8	PDMA_BA_ch8+0x14	R	PDMA Current Source Address Register CH8	0x0000_0000

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

Bits	Descriptions	
[31:0]	**PDMA_CSAR**	**PDMA Current Source Address Register (Read Only)** This field indicates the source address where the PDMA transfer is just occurring.

PDMA Current Destination Address Register (PDMA_CDARx)

Register	Offset	R/W	Description	Reset Value
PDMA_CDAR0	PDMA_BA_ch0+0x18	R	PDMA Current Destination Address Register CH0	0x0000_0000
PDMA_CDAR1	PDMA_BA_ch1+0x18	R	PDMA Current Destination Address Register CH1	0x0000_0000
PDMA_CDAR2	PDMA_BA_ch2+0x18	R	PDMA Current Destination Address Register CH2	0x0000_0000
PDMA_CDAR3	PDMA_BA_ch3+0x18	R	PDMA Current Destination Address Register CH3	0x0000_0000
PDMA_CDAR4	PDMA_BA_ch4+0x18	R	PDMA Current Destination Address Register CH4	0x0000_0000
PDMA_CDAR5	PDMA_BA_ch5+0x18	R	PDMA Current Destination Address Register CH5	0x0000_0000
PDMA_CDAR6	PDMA_BA_ch6+0x18	R	PDMA Current Destination Address Register CH6	0x0000_0000
PDMA_CDAR7	PDMA_BA_ch7+0x18	R	PDMA Current Destination Address Register CH7	0x0000_0000
PDMA_CDAR8	PDMA_BA_ch8+0x18	R	PDMA Current Destination Address Register CH8	0x0000_0000

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

Bits	Descriptions	
[31:0]	**PDMA_CDAR**	**PDMA Current Destination Address Register (Read Only)** This field indicates the destination address where the PDMA transfer is just occurring.

PDMA Current Byte Count Register (PDMA_CBCRx)

Register	Offset	R/W	Description	Reset Value
PDMA_CBCR0	PDMA_BA_ch0+0x1C	R	PDMA Current Byte Count Register CH0	0x0000_0000
PDMA_CBCR1	PDMA_BA_ch1+0x1C	R	PDMA Current Byte Count Register CH1	0x0000_0000
PDMA_CBCR2	PDMA_BA_ch2+0x1C	R	PDMA Current Byte Count Register CH2	0x0000_0000
PDMA_CBCR3	PDMA_BA_ch3+0x1C	R	PDMA Current Byte Count Register CH3	0x0000_0000
PDMA_CBCR4	PDMA_BA_ch4+0x1C	R	PDMA Current Byte Count Register CH4	0x0000_0000
PDMA_CBCR5	PDMA_BA_ch5+0x1C	R	PDMA Current Byte Count Register CH5	0x0000_0000
PDMA_CBCR6	PDMA_BA_ch6+0x1C	R	PDMA Current Byte Count Register CH6	0x0000_0000
PDMA_CBCR7	PDMA_BA_ch7+0x1C	R	PDMA Current Byte Count Register CH7	0x0000_0000
PDMA_CBCR8	PDMA_BA_ch8+0x1C	R	PDMA Current Byte Count Register CH8	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**
PDMA_CBCR [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PDMA_CBCR [7:0]							

Bits	Descriptions	
[31:16]	**Reserved**	Reserved
[15:0]	**PDMA_CBCR**	**PDMA Current Byte Count Register (Read Only)** This field indicates the current remained byte count of PDMA. Note: SW_RST will clear this register value.

PDMA Interrupt Enable Control Register (PDMA_IERx)

Register	Offset	R/W	Description	Reset Value
PDMA_IER0	PDMA_BA_ch0+0x20	R/W	PDMA Interrupt Enable Control Register CH0	0x0000_0001
PDMA_IER1	PDMA_BA_ch1+0x20	R/W	PDMA Interrupt Enable Control Register CH1	0x0000_0001
PDMA_IER2	PDMA_BA_ch2+0x20	R/W	PDMA Interrupt Enable Control Register CH2	0x0000_0001
PDMA_IER3	PDMA_BA_ch3+0x20	R/W	PDMA Interrupt Enable Control Register CH3	0x0000_0001
PDMA_IER4	PDMA_BA_ch4+0x20	R/W	PDMA Interrupt Enable Control Register CH4	0x0000_0001
PDMA_IER5	PDMA_BA_ch5+0x20	R/W	PDMA Interrupt Enable Control Register CH5	0x0000_0001
PDMA_IER6	PDMA_BA_ch6+0x20	R/W	PDMA Interrupt Enable Control Register CH6	0x0000_0001
PDMA_IER7	PDMA_BA_ch7+0x20	R/W	PDMA Interrupt Enable Control Register CH7	0x0000_0001
PDMA_IER8	PDMA_BA_ch8+0x20	R/W	PDMA Interrupt Enable Control Register CH8	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						**BLKD_IE**	**TABORT_IE**

Bits	Descriptions	
[31:2]	**Reserved**	Reserved
[1]	**BLKD_IE**	**PDMA Transfer Done Interrupt Enable** 1 = Enable interrupt generator during PDMA transfer done. 0 = Disable interrupt generator during PDMA transfer done.
[0]	**TABORT_IE**	**PDMA Read/Write Target Abort Interrupt Enable** 1 = Enable target abort interrupt generation during PDMA transfer. 0 = Disable target abort interrupt generation during PDMA transfer.

PDMA Interrupt Status Register (PDMA_ISRx)

Register	Offset	R/W	Description	Reset Value
PDMA_ISR0	PDMA_BA_ch0+0x24	R/W	PDMA Interrupt Status Register CH0	0x0X0X_0000
PDMA_ISR1	PDMA_BA_ch1+0x24	R/W	PDMA Interrupt Status Register CH1	0x0X0X_0000
PDMA_ISR2	PDMA_BA_ch2+0x24	R/W	PDMA Interrupt Status Register CH2	0x0X0X_0000
PDMA_ISR3	PDMA_BA_ch3+0x24	R/W	PDMA Interrupt Status Register CH3	0x0X0X_0000
PDMA_ISR4	PDMA_BA_ch4+0x24	R/W	PDMA Interrupt Status Register CH4	0x0X0X_0000
PDMA_ISR5	PDMA_BA_ch5+0x24	R/W	PDMA Interrupt Status Register CH5	0x0X0X_0000
PDMA_ISR6	PDMA_BA_ch6+0x24	R/W	PDMA Interrupt Status Register CH6	0x0X0X_0000
PDMA_ISR7	PDMA_BA_ch7+0x24	R/W	PDMA Interrupt Status Register CH7	0x0X0X_0000
PDMA_ISR8	PDMA_BA_ch8+0x24	R/W	PDMA Interrupt Status Register CH8	0x0X0X_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						BLKD_IF	TABORT_IF

Bits	Descriptions	
[31:2]	**Reserved**	Reserved
[1]	**BLKD_IF**	**Block Transfer Done Interrupt Flag** This bit indicates that PDMA has finished all transfer. 1 = Done 0 = Not finished yet Software can write 1 to clear this bit to zero
[0]	**TABORT_IF**	**PDMA Read/Write Target Abort Interrupt Flag** 1 = Bus ERROR response received 0 = No bus ERROR response received Software can write 1 to clear this bit to zero

Note: The PDMA_ISR [TABORT_IF] indicate bus master received ERROR response or not. If bus master received ERROR response, it means that target abort is happened. PDMAC will stop transfer and respond this event to software then go to IDLE state. When target abort occurred, software must reset PDMA, and then transfer those data again.

PDMA Shared Buffer FIFO 0 (PDMA_SBUF0_cx)

Register	Offset	R/W	Description	Reset Value
PDMA_SBUF0_c0	PDMA_BA_ch0+0x080	R	PDMA Shared Buffer FIFO 0 Register CH0	0x0000_0000
PDMA_SBUF0_c1	PDMA_BA_ch1+0x180	R	PDMA Shared Buffer FIFO 0 Register CH1	0x0000_0000
PDMA_SBUF0_c2	PDMA_BA_ch2+0x280	R	PDMA Shared Buffer FIFO 0 Register CH2	0x0000_0000
PDMA_SBUF0_c3	PDMA_BA_ch3+0x380	R	PDMA Shared Buffer FIFO 0 Register CH3	0x0000_0000
PDMA_SBUF0_c4	PDMA_BA_ch4+0x480	R	PDMA Shared Buffer FIFO 0 Register CH4	0x0000_0000
PDMA_SBUF0_c5	PDMA_BA_ch5+0x580	R	PDMA Shared Buffer FIFO 0 Register CH5	0x0000_0000
PDMA_SBUF0_c6	PDMA_BA_ch6+0x680	R	PDMA Shared Buffer FIFO 0 Register CH6	0x0000_0000
PDMA_SBUF0_c7	PDMA_BA_ch7+0x780	R	PDMA Shared Buffer FIFO 0 Register CH7	0x0000_0000
PDMA_SBUF0_c8	PDMA_BA_ch8+0x880	R	PDMA Shared Buffer FIFO 0 Register CH8	0x0000_0000

31	30	29	28	27	26	25	24
PDMA_SBUF0 [31:24]							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PDMA_SBUF0 [23:16]							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PDMA_SBUF0 [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PDMA_SBUF0 [7:0]							

Bits	Descriptions	
[31:0]	**PDMA_SBUF0**	**PDMA Shared Buffer FIFO 0 (Read Only)** Each channel has its own 1 words internal buffer.

PDMA Global Control Register (PDMA_GCRCSR)

Register	Offset	R/W	Description	Reset Value
PDMA_GCRCSR	PDMA_BA_GCR+0x00	R/W	PDMA Global Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							**CLK8_EN**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
CLK7_EN	**CLK6_EN**	**CLK5_EN**	**CLK4_EN**	**CLK3_EN**	**CLK2_EN**	**CLK1_EN**	**CLK0_EN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Descriptions	
[31:17]	**Reserved**	Reserved
[16]	**CLK8_EN**	**PDMA Controller Channel 8 Clock Enable Control** 0 = Disable 1 = Enable
[15]	**CLK7_EN**	**PDMA Controller Channel 7 Clock Enable Control** 0 = Disable 1 = Enable
[14]	**CLK6_EN**	**PDMA Controller Channel 6 Clock Enable Control** 0 = Disable 1 = Enable
[13]	**CLK5_EN**	**PDMA Controller Channel 5 Clock Enable Control** 0 = Disable 1 = Enable
[12]	**CLK4_EN**	**PDMA Controller Channel 4 Clock Enable Control** 0 = Disable 1 = Enable
[11	**CLK3_EN**	**PDMA Controller Channel 3 Clock Enable Control** 0 = Disable 1 = Enable

[10	**CLK2_EN**	**PDMA Controller Channel 2 Clock Enable Control** 0 = Disable 1 = Enable
[9]	**CLK1_EN**	**PDMA Controller Channel 1 Clock Enable Control** 0 = Disable 1 = Enable
[8]	**CLK0_EN**	**PDMA Controller Channel 0 Clock Enable Control** 0 = Disable 1 = Enable
[7:0]	**Reserved**	Reserved

PDMA Service Selection Control Register 0 (PDMA_PDSSR0)

Register	Address	R/W	Description	Reset Value
PDMA_PDSSR0	PDMA_BA_GCR+0x04	R/W	PDMA Service Selection Control Register 0	0xFFFF_FFFF

31	30	29	28	27	26	25	24
SPI3_TXSEL				SPI3_RXSEL			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
SPI2_TXSEL				SPI2_RXSEL			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SPI1_TXSEL				SPI1_RXSEL			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
SPI0_TXSEL				SPI0_RXSEL			

Bits	Descriptions	
[31:28]	**SPI3_TXSEL**	**PDMA SPI3 TX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral SPI3 TX. Software can configure the TX channel setting by SPI3_TXSEL. The channel configuration is the same as SPI0_RXSEL field. Please refer to the explanation of SPI0_RXSEL.
[27:24]	**SPI3_RXSEL**	**PDMA SPI3 RX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral SPI3 RX. Software can configure the RX channel setting by SPI3_RXSEL. The channel configuration is the same as SPI0_RXSEL field. Please refer to the explanation of SPI0_RXSEL.
[23:20]	**SPI2_TXSEL**	**PDMA SPI2 TX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral SPI2 TX. Software can configure the TX channel setting by SPI2_TXSEL. The channel configuration is the same as SPI0_RXSEL field. Please refer to the explanation of SPI0_RXSEL.
[19:16]	**SPI2_RXSEL**	**PDMA SPI2 RX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral SPI2 RX. Software can configure the RX channel setting by SPI2_RXSEL. The channel configuration is the same as SPI0_RXSEL field. Please refer to the explanation of SPI0_RXSEL.
[15:12]	**SPI1_TXSEL**	**PDMA SPI1 TX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral SPI1 TX. Software can configure the TX channel setting by SPI1_TXSEL. The channel configuration is the same as SPI0_RXSEL field. Please refer to the explanation of SPI0_RXSEL.

[11:8]	**SPI1_RXSEL**	**PDMA SPI1 RX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral SPI1 RX. Software can configure the RX channel setting by SPI1_RXSEL. The channel configuration is the same as SPI0_RXSEL field. Please refer to the explanation of SPI0_RXSEL.
[7:4]	**SPI0_TXSEL**	**PDMA SPI0 TX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral SPI0 TX. Software can configure the TX channel setting by SPI0_TXSEL. The channel configuration is the same as SPI0_RXSEL field. Please refer to the explanation of SPI0_RXSEL.
[3:0]	**SPI0_RXSEL**	**PDMA SPI0 RX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral SPI0 RX. Software can change the channel RX setting by SPI0_RXSEL 4’b0000: CH0 4’b0001: CH1 4’b0010: CH2 4’b0011: CH3 4’b0100: CH4 4’b0101: CH5 4’b0110: CH6 4’b0111: CH7 4’b1000: CH8 Others : Reserved Note: Ex : SPI0_RXSEL = 4’b0110, that means SPI0_RX is connected to PDMA_CH6

PDMA Service Selection Control Register 1 (PDMA_PDSSR1)

Register	Address	R/W	Description	Reset Value
PDMA_PDSSR1	PDMA_BA_GCR+0x08	R/W	PDMA Service Selection Control Register 1	0xFFFF_FFFF

31	30	29	28	27	26	25	24
Reserved				**ADC_RXSEL**			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
UART1_TXSEL				**UART1_RXSEL**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
UART0_TXSEL				**UART0_RXSEL**			

Bits	Descriptions	
[31:28]	**Reserved**	Reserved
[27:24]	**ADC_RXSEL**	**PDMA ADC RX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral ADC RX. Software can configure the RX channel setting by ADC_RXSEL. The channel configuration is the same as UART0_RXSEL field. Please refer to the explanation of UART0_RXSEL
[23:16]	**Reserved**	Reserved
[15:12]	**UART1_TXSEL**	**PDMA UART1 TX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral UART1 TX. Software can configure the TX channel setting by UART1_TXSEL. The channel configuration is the same as UART0_RXSEL field. Please refer to the explanation of UART0_RXSEL
[11:8]	**UART1_RXSEL**	**PDMA UART1 RX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral UART1 RX. Software can configure the RX channel setting by UART1_RXSEL. The channel configuration is the same as UART0_RXSEL field. Please refer to the explanation of UART0_RXSEL
[7:4]	**UART0_TXSEL**	**PDMA UART0 TX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral UART0 TX. Software can configure the TX channel setting by UART0_TXSEL. The channel configuration is the same as UART0_RXSEL field. Please refer to the explanation of UART0_RXSEL

[3:0]	**UART0_RXSEL**	This filed defines which PDMA channel is connected to the on-chip peripheral UART0 RX. Software can change the channel RX setting by UART0_RXSEL 4’b0000: CH0 4’b0001: CH1 4’b0010: CH2 4’b0011: CH3 4’b0100: CH4 4’b0101: CH5 4’b0110: CH6 4’b0111: CH7 4’b1000: CH8 Others : Reserved Note: Ex : UART0_RXSEL = 4’b0110, that means UART0_RX is connected to PDMA_CH6

PDMA Global Interrupt Status Register (PDMA_GCRISR)

Register	Offset	R/W	Description	Reset Value
PDMA_GCRISR	PDMA_BA_GCR+0x0C	R	PDMA Global Interrupt Status Register	0x0000_0000

31	30	29	28	27	26	25	24
INTR	**Reserved**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							**INTR8**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INTR7	**INTR6**	**INTR5**	**INTR4**	**INTR3**	**INTR2**	**INTR1**	**INTR0**

Bits	Descriptions	
[31]	**INTR**	**Interrupt Pin Status** This bit is the Interrupt status of PDMA controller. Note: This bit is read only
[30:9]	**Reserved**	Reserved
[8]	**INTR8**	**Interrupt Pin Status of Channel 8** This bit is the Interrupt status of PDMA channel8. Note: This bit is read only
[7]	**INTR7**	**Interrupt Pin Status of Channel 7** This bit is the Interrupt status of PDMA channel7. Note: This bit is read only
[6]	**INTR6**	**Interrupt Pin Status of Channel 6** This bit is the Interrupt status of PDMA channel6. Note: This bit is read only
[5]	**INTR5**	**Interrupt Pin Status of Channel 5** This bit is the Interrupt status of PDMA channel5. Note: This bit is read only
[4]	**INTR4**	**Interrupt Pin Status of Channel 4** This bit is the Interrupt status of PDMA channel4. Note: This bit is read only

[3]	**INTR3**	**Interrupt Pin Status of Channel 3** This bit is the Interrupt status of PDMA channel3. Note: This bit is read only
[2]	**INTR2**	**Interrupt Pin Status of Channel 2** This bit is the Interrupt status of PDMA channel2. Note: This bit is read only
[1]	**INTR1**	**Interrupt Pin Status of Channel 1** This bit is the Interrupt status of PDMA channel1. Note: This bit is read only
[0]	**INTR0**	**Interrupt Pin Status of Channel 0** This bit is the Interrupt status of PDMA channel0. Note: This bit is read only

PDMA Service Selection Control Register 2 (PDMA_PDSSR2)

Register	Offset	R/W	Description	Reset Value
PDMA_PDSSR2	PDMA_BA_GCR+0x10	R/W	PDMA Service Selection Control Register 2	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**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
I2S_TXSEL				**I2S_RXSEL**			

Bits	Descriptions	
[31:8]	**Reserved**	Reserved
[7:4]	**I2S_TXSEL**	**PDMA I^{2}S TX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral I^{2}S TX. Software can configure the TX channel setting by I2S_TXSEL. The channel configuration is the same as I2S_RXSEL field. Please refer to the explanation of I2S_RXSEL.
[3:0]	**I2S_RXSEL**	**PDMA I^{2}S RX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral I^{2}S RX. Software can change the channel RX setting by I2S_RXSEL 4’b0000: CH0 4’b0001: CH1 4’b0010: CH2 4’b0011: CH3 4’b0100: CH4 4’b0101: CH5 4’b0110: CH6 4’b0111: CH7 4’b1000: CH8 Others : Reserved Note: Ex : I2S_RXSEL = 4’b0110, that means I2S_RX is connected to PDMA_CH6

5.19 External Bus Interface (EBI)

5.19.1 Overview

The NuMicro™ NUC130/NUC140 LQFP-64 and LQFP-100 package equips an external bus interface (EBI) for external device used.

To save the connections between external device and this chip, EBI support address bus and data bus multiplex mode. And, address latch enable (ALE) signal supported differentiate the address and data cycle.

5.19.2 Features

External Bus Interface has the following functions:

- External devices with max. 64K-byte size (8-bit data width)/128K-byte (16-bit data width) supported
- Variable external bus base clock (MCLK) supported
- 8-bit or 16-bit data width supported
- Variable data access time (tACC), address latch enable time (tALE) and address hold time (tAHD) supported
- Address bus and data bus multiplex mode supported to save the address pins
- Configurable idle cycle supported for different access condition: Write command finish (W2X), Read-to-Read (R2R)

5.19.3 Block Diagram

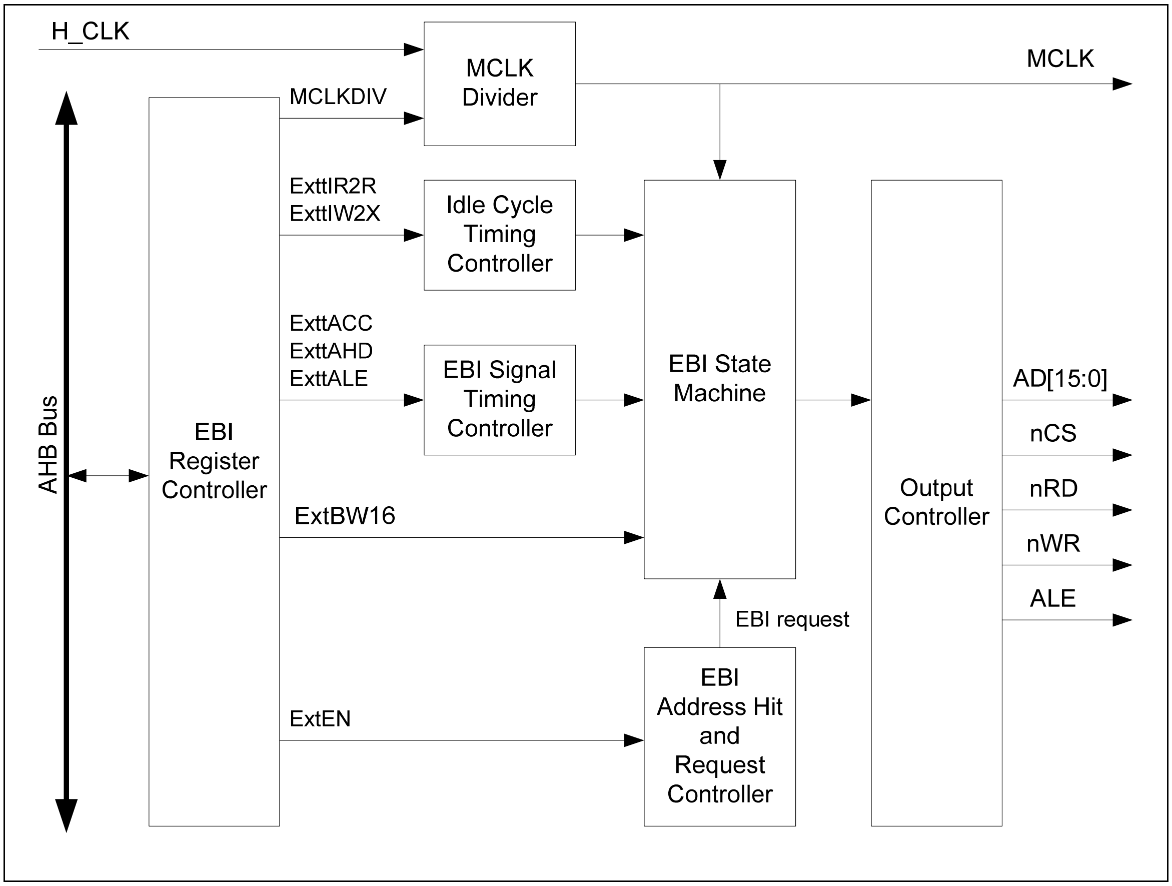


Figure 5-112 EBI Block Diagram

5.19.4 Function Description

5.19.4.1 EBI Area and Address Hit

EBI mapping address is located at 0x6000_0000 ~ 0x6001_FFFF and the total memory space is 128Kbyte. When system request address hit EBI's memory space, the corresponding EBI chip select signal is assert and EBI state machine operates.

For an 8-bit device (64Kbyte), EBI mapped this 64Kbyte device to 0x6000_0000 ~ 0x6000_FFFF and 0x6001_0000 ~ 0x6001_FFFF simultaneously.

5.19.4.2 EBI Data Width Connection

EBI support 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 logic to latch the address. In this case, pin ALE is connected to the latch device to latch the address value. Pin 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 AD [15:0] shared by address and 16-bit data. For 8-bit device, only AD [7:0] shared by address and 8-bit data, AD [15:8] is dedicated for address and could be connected to 8-bit device directly.

For 8-bit data width, chip system address bit [15:0] is used as the device's address [15:0]. For 16-bit data width, chip system address bit [16:1] is used as the device's address [15:0] and chip system address bit [0] is useless.

EBI bit width	System address (AHBADR)	EBI address (AD)
8-bit	AHBADR[15:0]	AD[15:0]
16-bit	AHBADR[16:1]	AD[15:0]

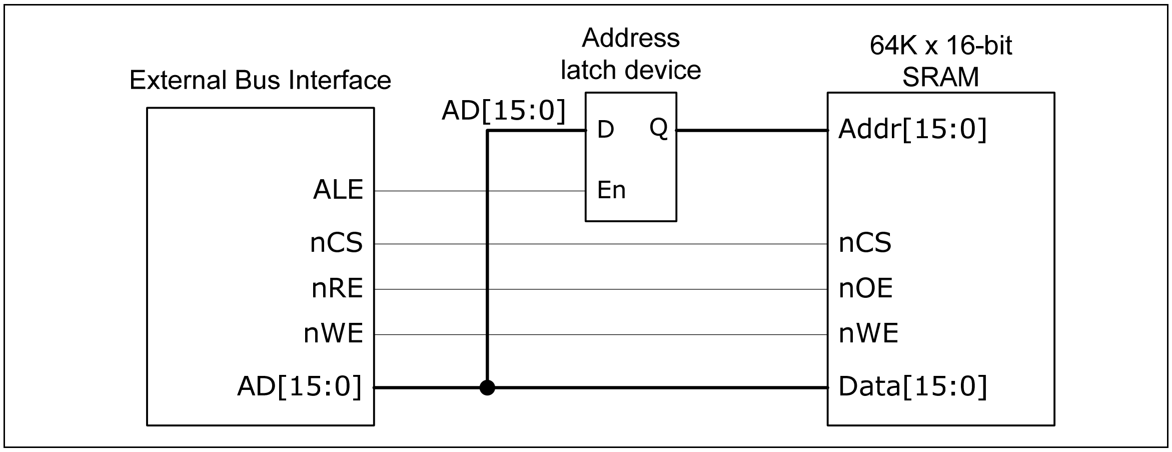


Figure 5-113 Connection of 16-bit EBI Data Width with 16-bit Device

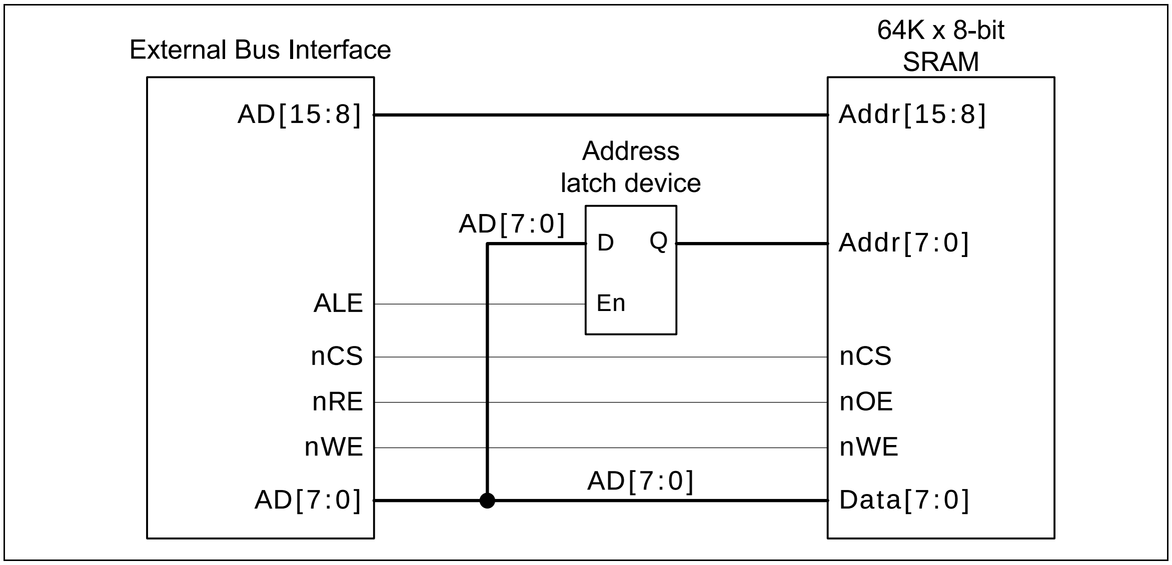


Figure 5-114 Connection of 8-bit EBI Data Width with 8-bit Device

When system access data width is lager 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.

5.19.4.3 EBI Operating Control

MCLK Control

In the chip, all EBI signals will be synchronized by MCLK when EBI is operating. When chip connects to the external device with slower operating frequency, the MCLK can divide most to HCLK/32 by setting MCLKDIV of register EBICON. Therefore, chip can suitable for a wide frequency range of EBI device. If MCLK is set to HCLK/1, EBI signals are synchronized by positive edge of MCLK, else by negative edge of MCLK.

Operation and Access Timing Control

In the start of access, chip select (nCS) asserts to low and wait one MCLK for address setup time (tASU) for address stable. Then ALE asserts to high after address is stable and keeps for a period of time (tALE) for address latch. After latch address, ALE asserts to low and wait one MCLK for latch hold time (tLHD) and another one MCLK cycle (tA2D) that is inserted behind address hold time to be the bus turn-around time for address change to data. Then nRD asserts to low when read access or nWR asserts to low when write access. Then nRD or 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.

EBI controller provides a flexible timing control for different external device. In EBI timing control, tASU, tLHD and tA2D are fixed to 1 MCLK cycle, tAHD can modulate to 1~8 MCLK cycles by setting ExttAHD of register EXTIME, tACC can modulate to 1~32 MCLK cycles by setting ExttACC of register EXTIME, and tALE can modulate to 1~8 MCLK cycles by setting tALE of register EBICON.

Parameter	Value	Unit	Description
tASU	1	MCLK	Address Latch Setup Time.
tALE	1 ~ 8	MCLK	ALE High Period. Controlled by ExttALE of EBICON.
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 ExttACC of EXTIME.
tAHD	1 ~ 8	MCLK	Data Access Hold Time. Controlled by ExttAHD of EXTIME.
IDLE	0 ~ 15	MCLK	Idle Cycle. Controlled by ExtIR2R and ExtIW2X of EXTIME.

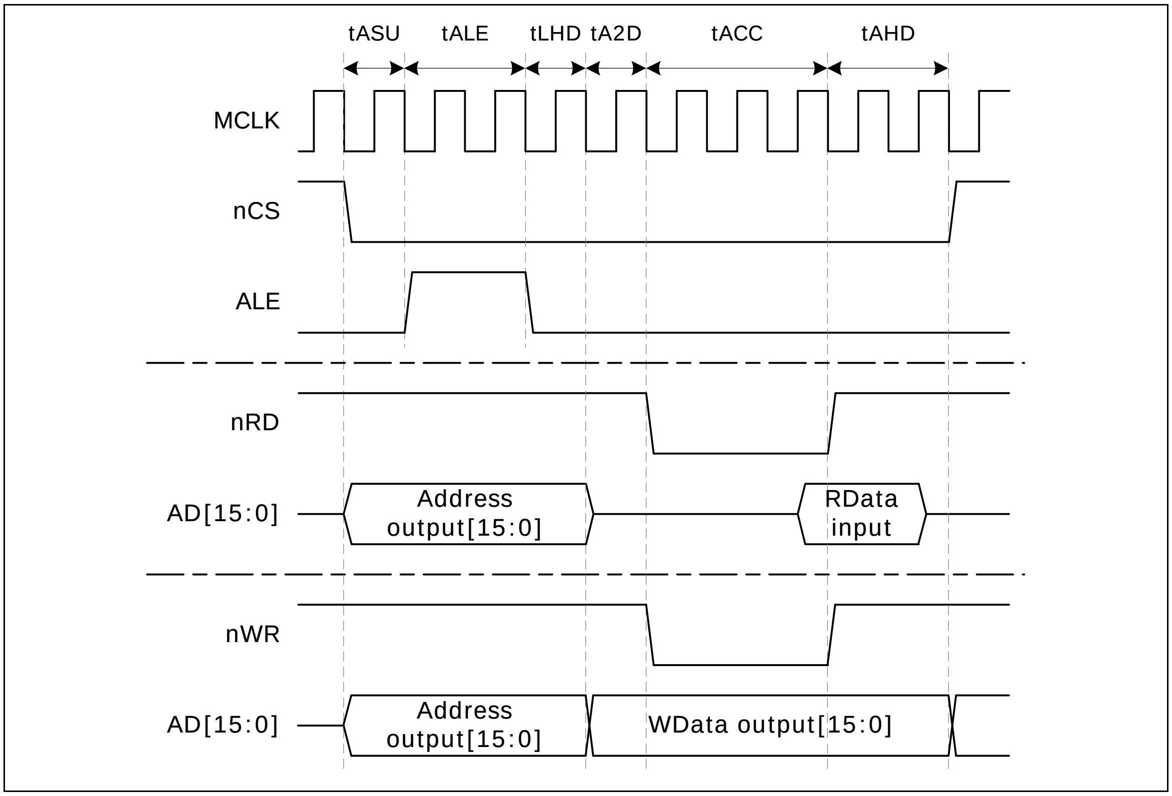


Figure 5-115 Timing Control Waveform for 16-bit Data Width

Figure 5-115 is an example of setting 16-bit data width. In this example, AD bus is used for being address[15:0] and data[15:0]. When ALE assert to high, AD is address output. After address is latched, ALE asserts to low and the AD bus change to high impedance to wait device output data in read access operation, or it is used for being write data output.

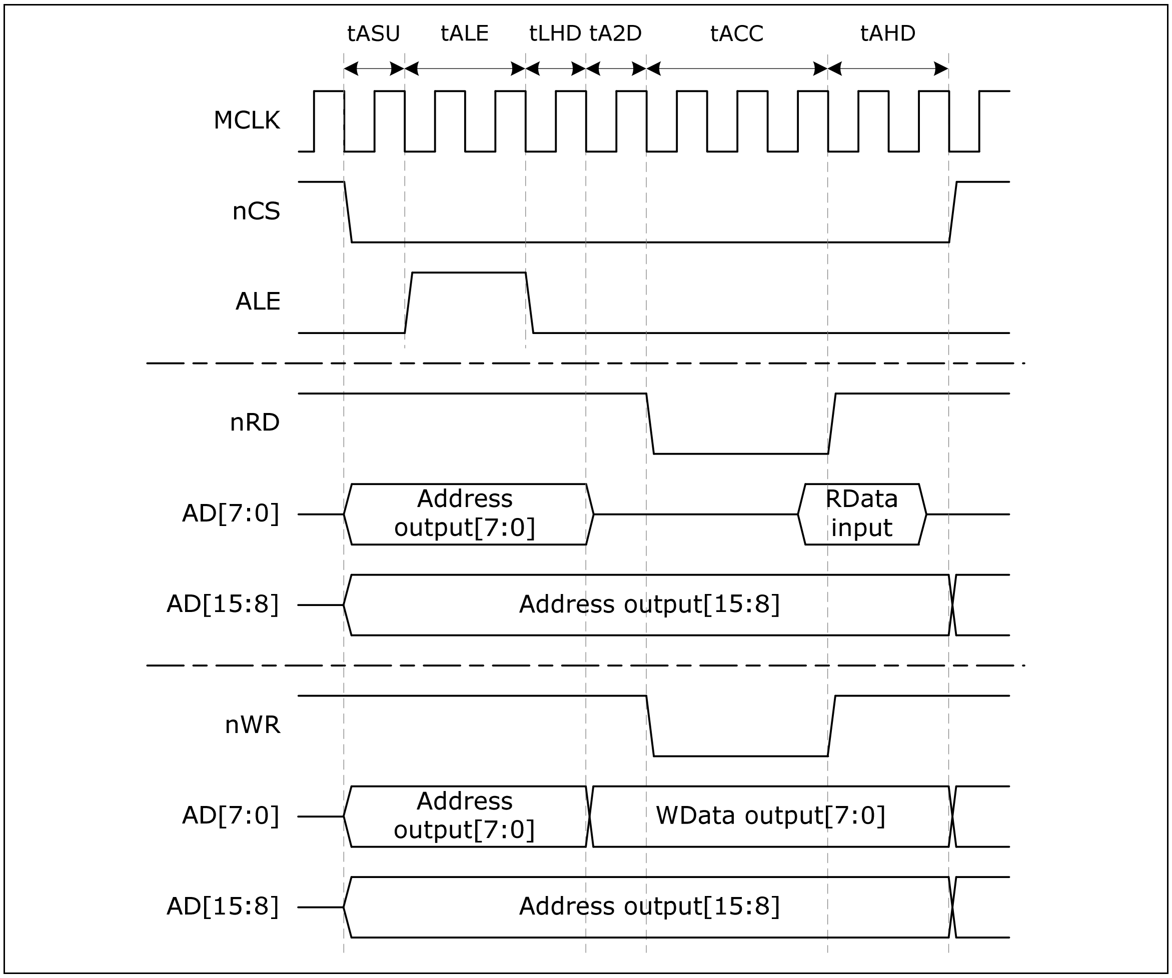


Figure 5-116 Timing Control Waveform for 8-bit Data Width

Figure 5-116 is an example of setting 8-bit data width. The difference between 8-bit and 16-bit data width is AD[15:8]. In 8-bit data width setting, 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 5-117 show idle cycle as below:

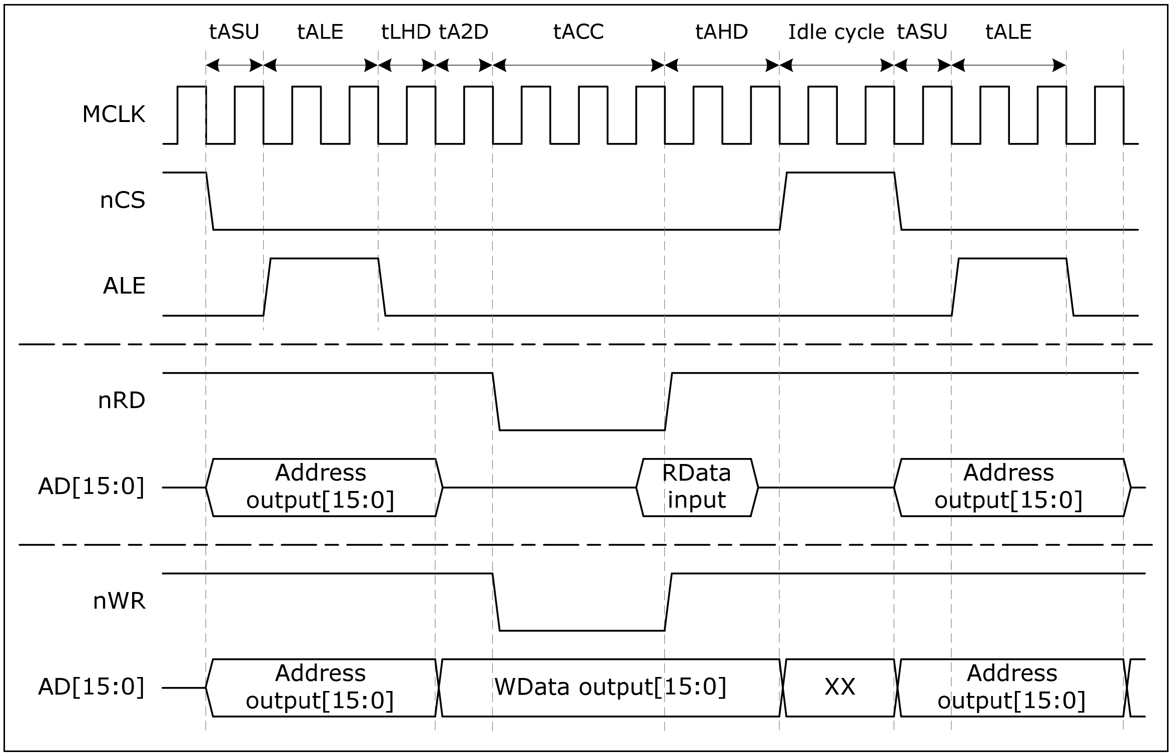


Figure 5-117 Timing Control Waveform for Insert Idle Cycle

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

By setting ExtIW2X, and ExtIR2R of register EXTIME, the time of idle cycle can be specified from 0~15 MCLK.

5.19.5 Register Map

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

Register	Offset	R/W	Description	Reset Value
EBI_BA = 0x5001_0000				
EBICON	EBI_BA+0x00	R/W	External Bus Interface General Control Register	0x0000_0000
EXTIME	EBI_BA+0x04	R/W	External Bus Interface Timing Control Register	0x0000_0000

5.19.6 Register Description

External Bus Interface CONTROL REGISTER (EBICON)

Register	Offset	R/W	Description	Reset Value
EBICON	EBI_BA+0x00	R/W	External Bus Interface General Control Register	0x0000_0000

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reversed					**ExttALE**		
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reversed					**MCLKDIV**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reversed						**ExtBW16**	**ExtEN**

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[31:19]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[18:16]</td><td>ExttALE</td><td>Expand Time of ALE
The ALE width (tALE) to latch the address can be controlled by ExttALE.
tALE = (ExttALE+1)*MCLK</td></tr>
<tr><td>[15:11]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[10:8]</td><td>MCLKDIV</td><td>External Output Clock Divider
The frequency of EBI output clock is controlled by MCLKDIV as follows table:
<table>
<tr><th>MCLKDIV</th><th>Output clock (MCLK)</th></tr>
<tr><td>000</td><td>HCLK/1</td></tr>
<tr><td>001</td><td>HCLK/2</td></tr>
</table>
</td></tr>
</table>

		010: HCLK/4 011: HCLK/8 100: HCLK/16 101: HCLK/32 11X: default Notice: Default value of output clock is HCLK/1
[7:2]	**Reserved**	Reserved
[1]	**ExtBW16**	**EBI data width 16-bit** This bit defines if the data bus is 8-bit or 16-bit. 1 = EBI data width is 16-bit 0 = EBI data width is 8-bit
[0]	**ExtEN**	**EBI Enable** This bit is the functional enable bit for EBI. 1 = EBI function is enabled 0 = EBI function is disabled

External Bus Interface Timing CONTROL REGISTER (EXTIME)

Register	Offset	R/W	Description	Reset Value
EXTIME	EBI_BA+0x04	R/W	External Bus Interface Timing Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				**ExtIR2R**			
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reversed							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ExtIW2X				**Reversed**	**ExttAHD**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ExttACC					**Reversed**		

Bits	Descriptions	
[31:28]	**Reserved**	Reserved
[27:24]	**ExtIR2R**	**Idle State Cycle Between Read-Read** When read action is finish and next action is going to read, idle state is inserted and nCS return to high if ExtIR2R is not zero. Idle state cycle = (ExtIR2R*MCLK)
[23:16]	**Reserved**	Reserved
[15:12]	**ExtIW2X**	**Idle State Cycle After Write** When write action is finish, idle state is inserted and nCS return to high if ExtIW2X is not zero. Idle state cycle = (ExtIW2X*MCLK)
[11]	**Reserved**	Reserved
[10:8]	**ExttAHD**	**EBI Data Access Hold Time** ExttAHD define data access hold time (tAHD). tAHD = (ExttAHD +1) * MCLK
[7:3]	**ExttACC**	**EBI Data Access Time** ExttACC define data access time (tACC). tACC = (ExttACC +1) * MCLK
[2:0]	**Reserved**	Reserved

6 FLASH MEMORY CONTROLLER (FMC)

6.1 Overview

NuMicro™ NUC100 Series equips with 128/64/32K bytes on chip embedded Flash for application program memory (APROM) that can be updated through ISP procedure. In System Programming (ISP) function enables user to update program memory when chip is soldered on PCB. After chip power on, Cortex-M0 CPU fetches code from APROM or LDROM decided by boot select (CBS) in Config0. By the way, NuMicro™ NUC100 Series also provides additional DATA Flash for user, to store some application dependent data before chip power off. For 128K bytes APROM device, the data flash is shared with original 128K program memory and its start address is configurable and defined by user application request in Config1. For 64K/32K bytes APROM device, the data flash is fixed at 4K.

6.2 Features

- Run up to 50 MHz with zero wait state for continuous address read access
- 128/64/32KB application program memory (APROM)
- 4KB in system programming (ISP) loader program memory (LDROM)
- Configurable or fixed 4KB data flash with 512 bytes page erase unit
- Programmable data flash start address for 128K APROM device
- In System Program (ISP) to update on chip Flash

6.3 Block Diagram

The flash memory controller consist of AHB slave interface, ISP control logic, writer interface and flash macro interface timing control logic. The block diagram of flash memory controller is shown as following:

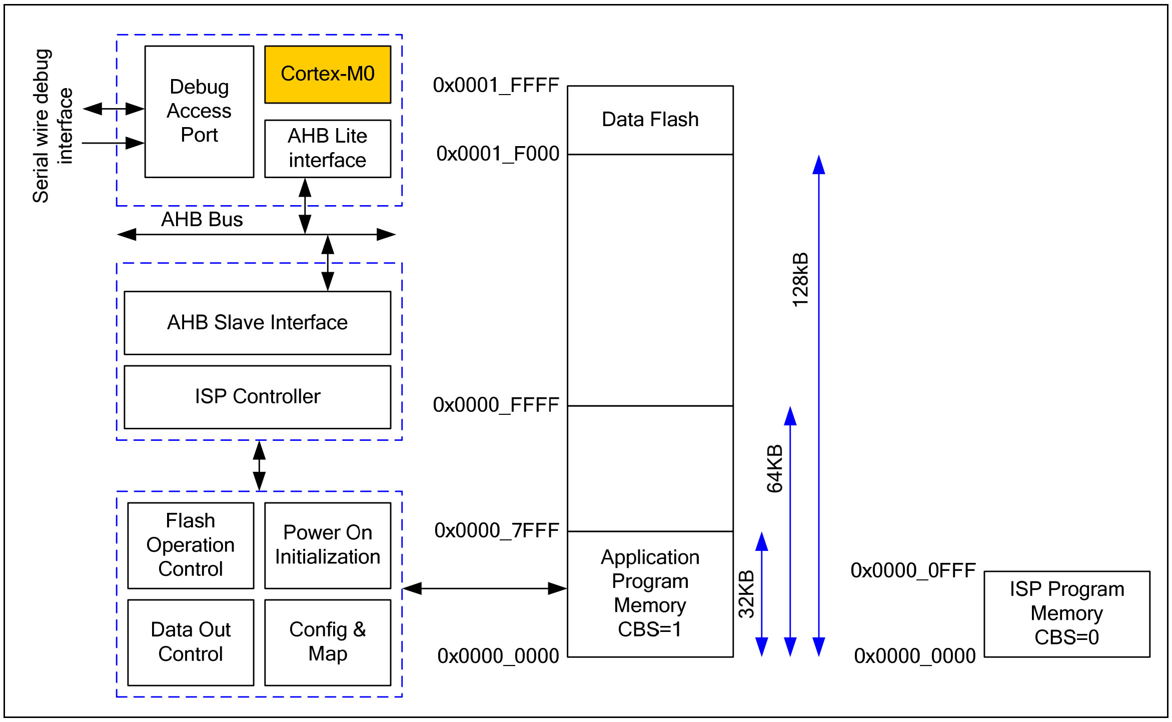


Figure 6-1 Flash Memory Control Block Diagram

6.4 Flash Memory Organization

NuMicro™ NUC100 Series flash memory consists of Program memory (128/64/32KB), data flash, ISP loader program memory, user configuration. User configuration block provides several bytes to control system logic, like flash security lock, boot select, Brown-Out voltage level, data flash base address, ..., and so on. 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 application request by writer before chip is mounted on PCB. The data flash start address and its size can defined by user depends on application in 128KB APROM device. For 64/32KB APROM devices, its size is 4KB and start address is fixed at 0x0001_F000.

Block Name	Size	Start Address	End Address
AP-ROM	32/64/(128-0.5*N) KB	0x0000_0000	0x0000_7FFF (32KB) 0x0000_FFFF (64KB) DFBADR-1 (128KB if DFEN=0)
Reserved for future use	896KB	0x0002_0000	0x000F_FFFF
Data Flash	4/4/0.5*N KB	0x0001_F000 DFBADR	0x0001_FFFF
LD-ROM	4 KB	0x0010_0000	0x0010_0FFF
User Configuration	2 words	0x0030_0000	0x0030_0004

Table 6-1 Memory Address Map

The Flash memory organization is shown as below:

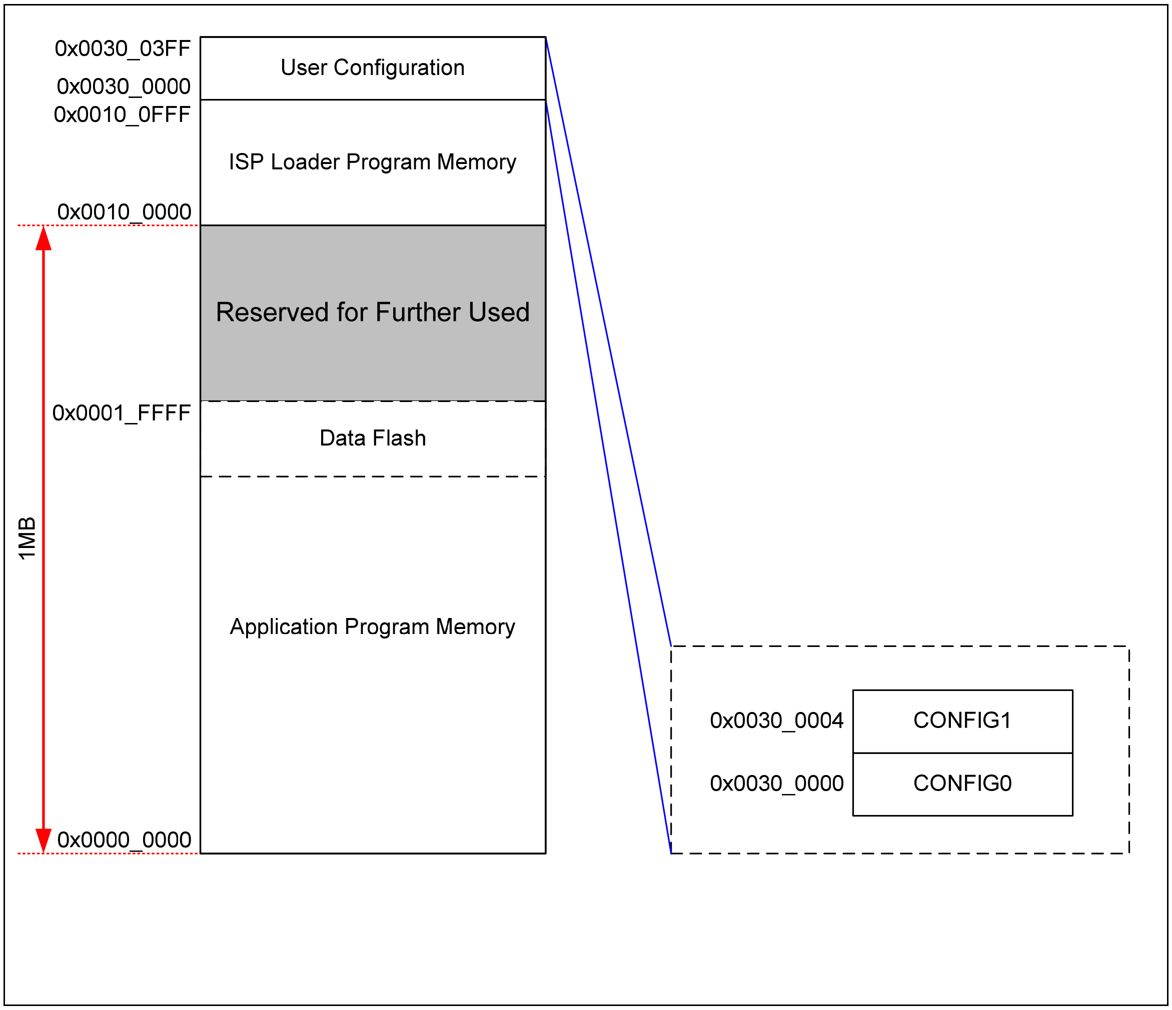


Figure 6-2 Flash Memory Organization

6.5 Boot Selection

NuMicro™ NUC100 Series provides in system programming (ISP) feature to enable user to update program memory when chip is mounted on PCB. A dedicated 4KB program memory is used to store ISP firmware. Users can select to start program fetch from APROM or LDROM by (CBS) in Config0.

6.6 Data Flash

NuMicro™ NUC100 Series provides data flash for user to store data. It is read/write through ISP procedure. The size of each erase unit is 512 bytes. When a word will be changed, all 128 words need to be copied to another page or SRAM in advance. For 128KB APROM device, the data flash and application program share the same 128KB memory, if DFEN bit in Config0 is enabled, the data flash base address is defined by DFBADR and application program memory size is (128-0.5*N)KB and data flash size is 0.5*N KB. For 64/32KB APROM devices, data flash size is 4KB and start address is fixed at 0x0001_F000.

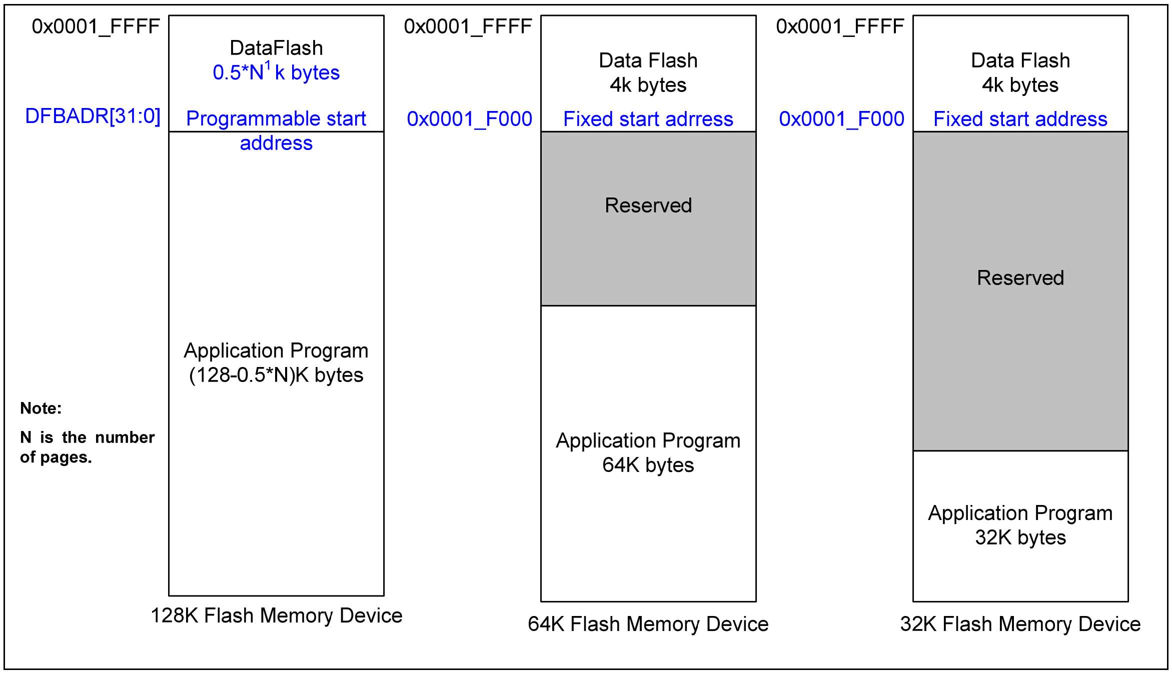


Figure 6-3 Flash Memory Structure

6.7 User Configuration

Config0 (Address = 0x0030_0000)

31	30	29	28	27	26	25	24
Reserved			CKF	Reserved	CFOSC		
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
CBODEN	CBOV1	CBOV0	CBORST	Reserved			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CBS	Reserved					LOCK	DFEN

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[31:29]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[28]</td><td>CKF</td><td>XT1 Clock Filter Enable
0 = Disable XT1 clock filter
1 = Enable XT1 clock filter</td></tr>
<tr><td>[27]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[26:24]</td><td>CFOSC</td><td>CPU Clock Source Selection After Reset
<table>
<tr><th>FOSC[2:0]</th><th>Clock Source</th></tr>
<tr><td>000</td><td>External 4~24 MHz high speed crystal clock</td></tr>
<tr><td>111</td><td>Internal RC 22.1184 MHz high speed oscillator clock</td></tr>
<tr><td>Others</td><td>Reserved</td></tr>
</table>
The value of CFOSC will be load to CLKSEL0.HCLK_S[2:0] in system register after any reset occurs.</td></tr>
<tr><td>[23]</td><td>CBODEN</td><td>Brown-Out Detector Enable
0= Enable Brown-Out detect after power on
1= Disable Brown-Out detect after power on</td></tr>
<tr><td>[22:21]</td><td>CBOV1-0</td><td>Brown-Out Voltage Selection
<table>
<tr><th>CBOV1</th><th>CBOV0</th><th>Brown-Out voltage</th></tr>
<tr><td>1</td><td>1</td><td>4.5 V</td></tr>
<tr><td>1</td><td>0</td><td>3.8 V</td></tr>
<tr><td>0</td><td>1</td><td>2.7 V</td></tr>
<tr><td>0</td><td>0</td><td>2.2 V</td></tr>
</table></td></tr>
</table>

[20]	**CBORST**	**Brown-Out Reset Enable** 0 = Enable Brown-Out reset after power on 1 = Disable Brown-Out reset after power on
[19:8]	**Reserved**	Reserved
[7]	**CBS**	**Chip Boot Selection** 0 = Chip boot from LDROM 1 = Chip boot from APROM
[6:2]	**Reserved**	Reserved
[1]	**LOCK**	**Security Lock** 0 = Flash data is locked 1 = Flash data is not locked When flash data is locked, only device ID, Config0 and Config1 can be read by writer and ICP through serial debug interface. Others data is locked as 0xFFFFFFFF. ISP can read data anywhere regardless of LOCK bit value.
[0]	**DFEN**	**Data Flash Enable** (This bit is work only for 128KB APROM device) 0 = Enable data flash 1 = Disable data flash

Config1 (Address = 0x0030_0004)

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

Bits	Descriptions	
[31:20]	**Reserved**	Reserved (It is mandatory to program 0x00 to these Reserved bits)
[19:0]	**DFBADR**	**Data Flash Base Address** (This register is work only for 128KB APROM device) For 128KB APROM device, its data flash base address is defined by user. Since on chip flash erase unit is 512 bytes, it is mandatory to keep bit 8-0 as 0. This configuration is only valid for 128KB flash device.

6.8 In System Program (ISP)

The program memory and data flash supports both in hardware programming and in system programming (ISP). Hardware programming mode uses gang-writers to reduce programming costs and time to market while the products enter into the mass production state. However, if the product is just under development or the end product needs firmware updating in the hand of an end user, the hardware programming mode will make repeated programming difficult and inconvenient. ISP method makes it easy and possible. NuMicro™ NUC100 Series supports ISP mode allowing a device to be reprogrammed under software control. Furthermore, the capability to update the application firmware makes wide range of applications possible.

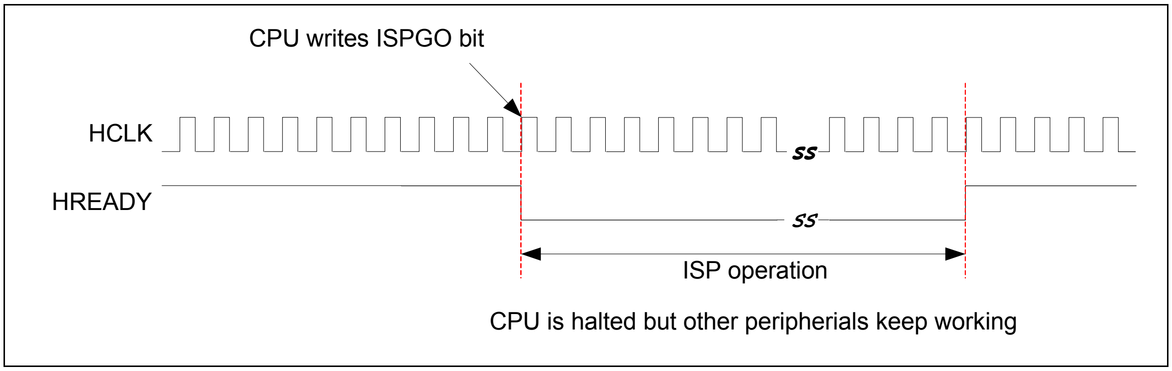
ISP is performed without removing the microcontroller from the system. Various interfaces enable LDROM firmware to get new program code easily. The most common method to perform ISP is via UART along with the firmware in LDROM. General speaking, PC transfers the new APROM code through serial port. Then LDROM firmware receives it and re-programs into APROM through ISP commands. Nuvoton provides ISP firmware and PC application program for NuMicro™ NUC100 Series. It makes users quite easy perform ISP through Nuvoton ISP tool.

6.8.1 ISP Procedure

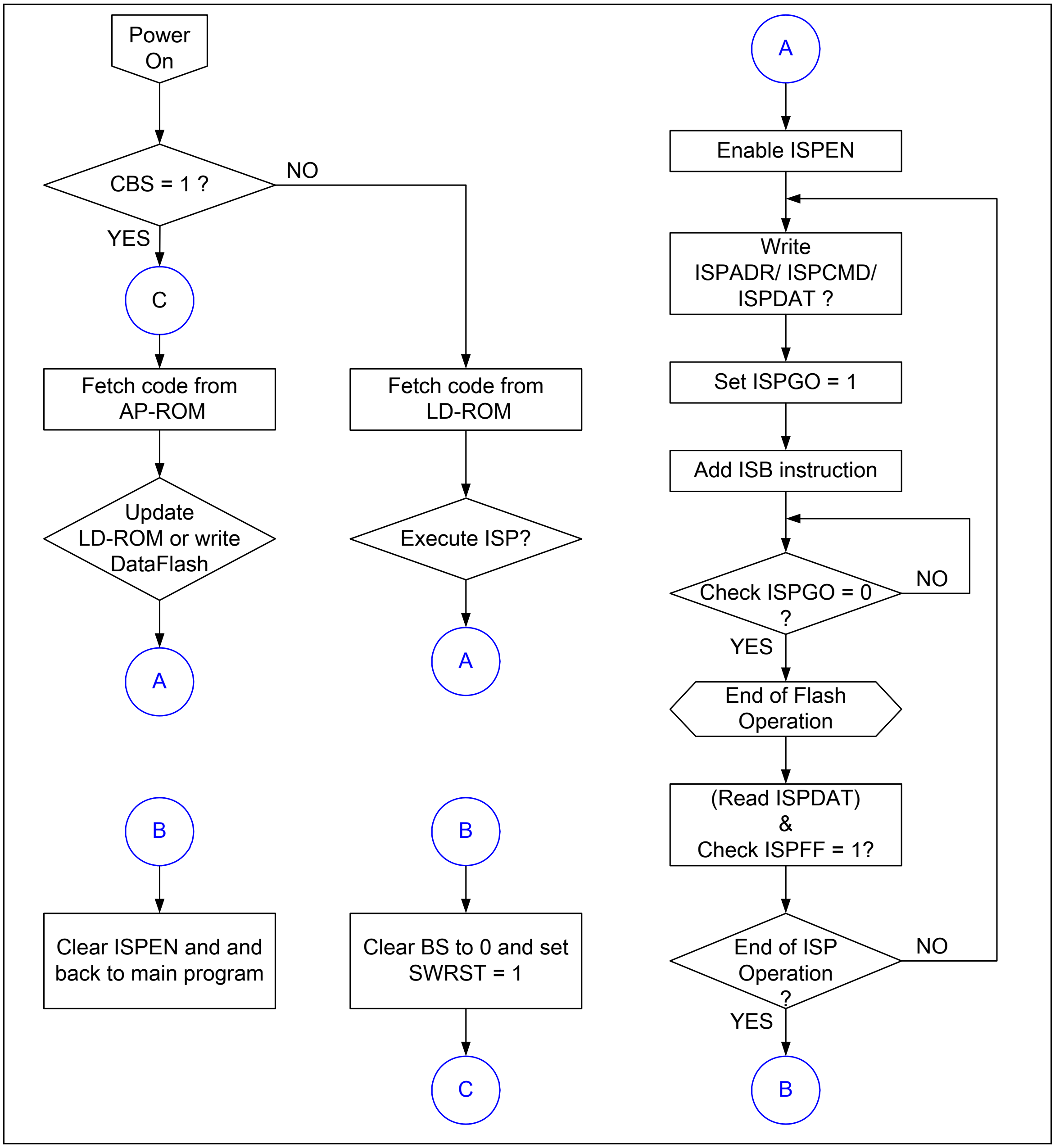
NuMicro™ NUC100 Series supports booting from APROM or LDROM initially defined by user configuration bit (CBS). If user wants to update application program in APROM, he can write BS=1 and starts software reset to make chip boot from LDROM. The first step to start ISP function is write ISPEN bit to 1. S/W is required to write REGWRPROT register in Global Control Register (GCR, 0x5000_0100) with 0x59, 0x16 and 0x88 before writing ISPCON register. This procedure is used to protect flash memory from destroying owning to unintended write during power on/off duration.

Several error conditions are checked after software writes ISPGO bit. If error condition occurs, ISP operation is not been started and ISP fail flag will be set instead of. ISPFF flag is cleared by s/w, it will not be over written in next ISP operation. The next ISP procedure can be started even ISPFF bit keeps at 1. It is recommended that s/w to check ISPFF bit and clear it after each ISP operation if it is set to 1.

When ISPGO bit is set, CPU will wait for ISP operation finish, during this period; peripheral still keeps working as usual. If any interrupt request occur, CPU will not service it till ISP operation finish. When ISP operation is finished, the ISPGO bit will be cleared by hardware automatically. User can know if ISP operation is finished by checking this bit. User should add ISB instruction next to the instruction which set 1 to ISPGO bit to ensure correct execution of the instructions following ISP operation.



Note that NuMicro™ NUC100 Series allows user to update CONFIG value by ISP.


Power On
CBS = 1 ?
NO
YES
C
Fetch code from AP-ROM
Fetch code from LD-ROM
Update LD-ROM or write DataFlash
Execute ISP?
A
A
B
B
Clear ISPEN and and back to main program
Clear BS to 0 and set SWRST = 1
C
A
Enable ISPEN
Write ISPADR/ ISPCMD/ ISPDAT ?
Set ISPGO = 1
Add ISB instruction
Check ISPGO = 0 ?
NO
YES
End of Flash Operation
(Read ISPDAT) & Check ISPFF = 1?
End of ISP Operation ?
NO
YES
B

ISP Mode	**ISPCMD**			**ISPADR**			**ISPDAT**
	FOEN	FCEN	FCTRL[3:0]	A21	A20	A[19:0]	D[31:0]
FLASH Page Erase	1	0	0010	0	A20	Address in A[19:0]	x
FLASH Program	1	0	0001	0	A20	Address in A[19:0]	Data in D[31:0]
FLASH Read	0	0	0000	0	A20	Address in A[19:0]	Data out D[31:0]
CONFIG Page Erase	1	0	0010	1	1	Address in A[19:0]	x
CONFIG Program	1	0	0001	1	1	Address in A[19:0]	Data in D[31:0]
CONFIG Read	0	0	0000	1	1	Address in A[19:0]	Data out D[31:0]

Table 6-2 ISP Mode

6.9 Flash Control Register Map

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

Register	Offset	R/W	Description	Reset Value
Base Address (FMC_BA) : 0x5000_C000				
ISPCON	FMC_BA+0x00	R/W	ISP Control Register	0x0000_0000
ISPADR	FMC_BA+0x04	R/W	ISP Address Register	0x0000_0000
ISPDAT	FMC_BA+0x08	R/W	ISP Data Register	0x0000_0000
ISPCMD	FMC_BA+0x0C	R/W	ISP Command Register	0x0000_0000
ISPTRG	FMC_BA+0x10	R/W	ISP Trigger Register	0x0000_0000
DFBADR	FMC_BA+0x14	R	Data Flash Start Address (AP ROM size is less than 128KB)	0x0001_F000
DFBADR	FMC_BA+0x14	R	Data Flash Start Address (AP ROM size is equal to 128KB)	0x0000_0000
FATCON	FMC_BA+0x18	R/W	Flash Access Window Control Register	0x0000_0000

6.10 Flash Control Register Description

ISP Control Register (ISPCON)

Register	Offset	R/W	Description	Reset Value
ISPCON	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							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved	**ET**			**Reserved**	**PT**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**ISPFF**	**LDUEN**	**CFGUEN**	**APUEN**	**Reserved**	**BS**	**ISPEN**

Bits	Descriptions	
[31:15]	**Reserved**	Reserved
[14:12]	**ET[2:0]**	**Flash Erase Time** (write-protection bits) ET[2] | ET[1] | ET[0] | Erase Time (ms) 0 | 0 | 0 | 20 (default) 0 | 0 | 1 | 25 0 | 1 | 0 | 30 0 | 1 | 1 | 35 1 | 0 | 0 | 3 1 | 0 | 1 | 5 1 | 1 | 0 | 10 1 | 1 | 1 | 15
[11]	**Reserved**	Reserved

<table>
<tr><td>[8:10]</td><td>PT[2:0]</td><td>Flash Program Time (write-protection bits)
<table>
<tr><th>PT[2]</th><th>PT[1]</th><th>PT[0]</th><th>Program Time (us)</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>40</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>45</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>50</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>55</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>20</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>25</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>30</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>35</td></tr>
</table></td></tr>
<tr><td>[7]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[6]</td><td>ISPFF</td><td>ISP Fail Flag (write-protection bit)
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
(3) CONFIG is erased/programmed if CFGUEN is set to 0
(4) Destination address is illegal, such as over an available range
Write 1 to clear.</td></tr>
<tr><td>[5]</td><td>LDUEN</td><td>LDROM Update Enable (write-protection bit)
LDROM update enable bit.
1 = LDROM can be updated when the chip runs in APROM
0 = LDROM can not be updated</td></tr>
<tr><td>[4]</td><td>CFGUEN</td><td>Enable Config-bits Update by ISP (write-protection bit)
1 = Enable ISP can update config-bits
0 = Disable ISP can update config-bits</td></tr>
<tr><td>[3]</td><td>APUEN</td><td>APROM Update Enable (write-protection bit)
1 = APROM can be updated when the chip runs in APROM
0 = APROM can not be updated when the chip runs in APROM</td></tr>
<tr><td>[2]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[1]</td><td>BS</td><td>Boot Select (write-protection bit)
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 in Config0 after any reset is happened except CPU reset (RSTS_CPU is 1) or system reset (RSTS_SYS) is happened
1 = boot from LDROM
0 = boot from APROM</td></tr>
</table>

[0]	**ISPEN**	**ISP Enable** (write-protection bit) ISP function enable bit. Set this bit to enable ISP function. 1 = Enable ISP function 0 = Disable ISP function

ISP Address (ISPADR)

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

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

Bits	Descriptions	
[31:0]	**ISPADR**	**ISP Address** NuMicro™ NUC100 Series equips with a maximum 32Kx32 embedded flash, it supports word program only. ISPADR[1:0] must be kept 00b for ISP operation.

ISP Data Register (ISPDAT)

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

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

Bits	Descriptions	
[31:0]	**ISPDAT**	**ISP Data** Write data to this register before ISP program operation Read data from this register after ISP read operation

ISP Command (ISPCMD)

Register	Offset	R/W	Description	Reset Value
ISPCMD	FMC_BA+0x0C	R/W	ISP Command 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		FOEN	FCEN	FCTRL			

<table>
<tr><th>Bits</th><th colspan="2">Descriptions</th></tr>
<tr><td>[31:6]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[5]</td><td>FOEN</td><td rowspan="3">ISP Command
ISP command table is showed below:
<table>
<tr><th>Operation Mode</th><th>FOEN</th><th>FCEN</th><th colspan="4">FCTRL[3:0]</th></tr>
<tr><td>Read</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>Program</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>Page Erase</td><td>1</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td></tr>
</table></td></tr>
<tr><td>[4]</td><td>FCEN</td></tr>
<tr><td>[3:0]</td><td>FCTRL</td></tr>
</table>

ISP Trigger Control Register (ISPTRG)

Register	Offset	R/W	Description	Reset Value
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	Descriptions	
[31:1]	**Reserved**	Reserved
[0]	**ISPGO**	**ISP start trigger** Write 1 to start ISP operation and this bit will be cleared to 0 by hardware automatically when ISP operation is finished. 1 = ISP is on going 0 = ISP operation is finished

Data Flash Base Address Register (DFBADR)

Register	Offset	R/W	Description	Reset Value
DFBADR	FMC_BA+0x14	R	Data flash Base Address	0x0001_F000

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

Bits	Descriptions	
[31:0]	**DFBADR**	**Data Flash Base Address** This register indicates data flash start address. It is a read only register. For 128KB flash memory device, the data flash size is defined by user configuration, register content is loaded from Config1 when chip power on but for 64/32KB device, it is fixed at 0x0001_F000.

Flash Access Time Control Register (FATCON)

Register	Offset	R/W	Description	Reset Value
FATCON	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			**LFOM**	**FATS[2:0]**			**FPSEN**

Bits	Descriptions	
[31:5]	**Reserved**	Reserved
[4]	**LFOM**	**Low Frequency Optimization Mode** (write-protection bit) When chip operation frequency is lower than 25 MHz, chip can work more efficiently by setting this bit to 1 1 = Enable low frequency optimization mode 0 = Disable low frequency optimization mode
[3:1]	**FATS**	**Flash Access Time Window Select** (write-protection bits) These bits are used to decide flash sense amplifier active duration. FATS / Access Time window (ns): 000 / 40 001 / 50 010 / 60 011 / 70 100 / 80 101 / 90 110 / 100 111 / Reserved
[0]	**FPSEN**	**Flash Power Save Enable** (write-protection bit) If CPU clock is slower than 24 MHz, then s/w can enable flash power saving function. 1 = Enable flash power saving 0 = Disable flash power saving

FATS	Access Time window (ns)
000	40
001	50
010	60
011	70
100	80
101	90
110	100
111	Reserved

7 ELECTRICAL CHARACTERISTICS

7.1 Absolute Maximum Ratings

PARAMETER	SYMBOL	MIN.	MAX	UNIT
DC Power Supply	VDD–VSS	-0.3	+7.0	V
Input Voltage	VIN	VSS-0.3	VDD+0.3	V
Oscillator Frequency	$1/t_{CLCL}$	4	24	MHz
Operating Temperature	TA	-40	+85	°C
Storage Temperature	TST	-55	+150	°C
Maximum Current into VDD		-	120	mA
Maximum Current out of VSS			120	mA
Maximum Current sunk by a I/O pin			35	mA
Maximum Current sourced by a I/O pin			35	mA
Maximum Current sunk by total I/O pins			100	mA
Maximum Current sourced by total I/O pins			100	mA

Note: Exposure to conditions beyond those listed under absolute maximum ratings may adversely affects the lift and reliability of the device.

7.2 DC Electrical Characteristics

7.2.1 NuMicro™ NUC130/NUC140 DC Electrical Characteristics

(VDD-VSS=3.3 V, TA = 25°C, FOSC = 50 MHz unless otherwise specified.)

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Operation voltage	V_{DD}	2.5		5.5	V	V_{DD} =2.5 V ~ 5.5 V up to 50 MHz
Power Ground	V_{SS} AV_{SS}	-0.3			V	
LDO Output Voltage	V_{LDO}	-10%	2.5	+10%	V	V_{DD} > 2.7 V
Analog Operating Voltage	AV_{DD}	0		V_{DD}	V	
Analog Reference Voltage	Vref	0		AV_{DD}	V	
Operating Current Normal Run Mode @ 50 MHz	I_{DD1}		51		mA	V_{DD} = 5.5 V@50 MHz, enable all IP and PLL, XTAL=12 MHz
	I_{DD2}		25		mA	V_{DD} = 5.5 V@50 MHz, disable all IP and enable PLL, XTAL=12 MHz
	I_{DD3}		48		mA	V_{DD} = 3 V@50 MHz, enable all IP and PLL, XTAL=12 MHz
	I_{DD4}		23		mA	V_{DD} = 3 V@50 MHz, disable all IP and enable PLL, XTAL=12 MHz
Operating Current Normal Run Mode @ 12 MHz	I_{DD5}		19		mA	V_{DD} = 5.5 V@12 MHz, enable all IP and disable PLL, XTAL=12 MHz
	I_{DD6}		7		mA	V_{DD} = 5.5 V@12 MHz, disable all IP and disable PLL, XTAL=12 MHz
	I_{DD7}		17		mA	V_{DD} = 3 V@12 MHz, enable all IP and disable PLL, XTAL=12 MHz

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
	I_{DD8}		6		mA	V_{DD} = 3 V@12 MHz, disable all IP and disable PLL, XTAL=12 MHz
Operating Current Normal Run Mode @ 4 MHz	I_{DD9}		11		mA	V_{DD} = 5 V@4 MHz, enable all IP and disable PLL, XTAL=4 MHz
	I_{DD10}		3		mA	V_{DD} = 5 V@4 MHz, disable all IP and disable PLL, XTAL=4 MHz
	I_{DD11}		10		mA	V_{DD} = 3 V@4 MHz, enable all IP and disable PLL, XTAL=4 MHz
	I_{DD12}		2.5		mA	V_{DD} = 3 V@4 MHz, disable all IP and disable PLL, XTAL=4 MHz
Operating Current Idle Mode @ 50 MHz	I_{IDLE1}		35		mA	V_{DD}= 5.5 V@50 MHz, enable all IP and PLL, XTAL=12 MHz
	I_{IDLE2}		15		mA	V_{DD}=5.5 V@50 MHz, disable all IP and enable PLL, XTAL=12 MHz
	I_{IDLE3}		33		mA	V_{DD} = 3 V@50 MHz, enable all IP and PLL, XTAL=12 MHz
	I_{IDLE4}		13		mA	V_{DD} = 3 V@50 MHz, disable all IP and enable PLL, XTAL=12 MHz
Operating Current Idle Mode @ 12 MHz	I_{IDLE5}		10		mA	V_{DD} = 5.5 V@12 MHz, enable all IP and disable PLL, XTAL=12 MHz
	I_{IDLE6}		4.5		mA	V_{DD} = 5.5 V@12 MHz, disable all IP and disable PLL, XTAL=12 MHz
	I_{IDLE7}		9		mA	V_{DD} = 3 V@12 MHz, enable all IP and disable PLL, XTAL=12 MHz

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
	I_{IDLE8}		3.5		mA	V_{DD} = 3 V@12 MHz, disable all IP and disable PLL, XTAL=12 MHz
Operating Current Idle Mode @ 4 MHz	I_{IDLE9}		4		mA	V_{DD} = 5 V@4 MHz, enable all IP and disable PLL, XTAL=4 MHz
	I_{IDLE10}		2.5		mA	V_{DD} = 5 V@4 MHz, disable all IP and disable PLL, XTAL=4 MHz
	I_{IDLE11}		3.5		mA	V_{DD} = 3 V@4 MHz, enable all IP and disable PLL, XTAL=4 MHz
	I_{IDLE12}		1.5		mA	V_{DD} = 3 V@4 MHz, disable all IP and disable PLL, XTAL=4 MHz
Standby Current Power down Mode	I_{PWD1}		12		μA	V_{DD} = 5.5 V, RTC OFF, No load @ Disable BOV function
	I_{PWD2}		9		μA	V_{DD} = 3.3 V, RTC OFF, No load @ Disable BOV function
	I_{PWD3}				μA	V_{DD} = 5.5 V, RTC run , No load @ Disable BOV function
	I_{PWD4}				μA	V_{DD} = 3.3 V, RTC run , No load @ Disable BOV function
Input Current PA, PB, PC, PD, PE (Quasi-bidirectional mode)	I_{IN1}		-50	-60	μA	V_{DD} = 5.5 V, V_{IN} = 0 V or V_{IN}=V_{DD}
Input Current at /RESET[1]	I_{IN2}	-55	-45	-30	μA	V_{DD} = 3.3 V, V_{IN} = 0.45 V
Input Leakage Current PA, PB, PC, PD, PE	I_{LK}	-2	-	+2	μA	V_{DD} = 5.5 V, 0<V_{IN}<V_{DD}
Logic 1 to 0 Transition Current PA~PE (Quasi-bidirectional mode)	I_{TL} [3]	-650	-	-200	μA	V_{DD} = 5.5 V, V_{IN}<2.0 V
Input Low Voltage PA, PB, PC, PD, PE (TTL input)	V_{IL1}	-0.3	-	0.8	V	V_{DD} = 4.5 V
		-0.3	-	0.6		V_{DD} = 2.5 V
Input High Voltage PA, PB, PC, PD, PE (TTL input)	V_{IH1}	2.0	-	V_{DD} +0.2	V	V_{DD} = 5.5 V
		1.5	-	V_{DD} +0.2		V_{DD} =3.0 V
Input Low Voltage PA, PB, PC, PD, PE (Schmitt input)	V_{IL2}	-0.5	-	0.4 V_{DD}	V	

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Input High Voltage PA, PB, PC, PD, PE (Schmitt input)	V_{IH2}	0.6 V_{DD}	-	V_{DD}+0.5	V	
Hysteresis voltage of PA~PE (Schmitt input)	V_{HY}		0.2 V_{DD}		V	
Input Low Voltage XT1[*2]	V_{IL3}	0	-	0.8	V	V_{DD} = 4.5 V
		0	-	0.4		V_{DD} = 3.0 V
Input High Voltage XT1[*2]	V_{IH3}	3.5	-	V_{DD} +0.2	V	V_{DD} = 5.5 V
		2.4	-	V_{DD} +0.2		V_{DD} = 3.0 V
Input Low Voltage X32I[*2]	V_{IL4}	0	-	0.4	v	
Input High Voltage X32I[*2]	V_{IH4}	1.7		2.5	V	
Negative going threshold (Schmitt input), /RESET	V_{ILS}	-0.5	-	0.3 V_{DD}	V	
Positive going threshold (Schmitt input), /RESET	V_{IHS}	0.7 V_{DD}	-	V_{DD}+0.5	V	
Source Current PA, PB, PC, PD, PE (Quasi-bidirectional Mode)	I_{SR11}	-300	-370	-450	μA	V_{DD} = 4.5 V, V_S = 2.4 V
	I_{SR12}	-50	-70	-90	μA	V_{DD} = 2.7 V, V_S = 2.2 V
	I_{SR12}	-40	-60	-80	μA	V_{DD} = 2.5 V, V_S = 2.0 V
Source Current PA, PB, PC, PD, PE (Push-pull Mode)	I_{SR21}	-20	-24	-28	mA	V_{DD} = 4.5 V, V_S = 2.4 V
	I_{SR22}	-4	-6	-8	mA	V_{DD} = 2.7 V, V_S = 2.2 V
	I_{SR22}	-3	-5	-7	mA	V_{DD} = 2.5 V, V_S = 2.0 V
Sink Current PA, PB, PC, PD, PE (Quasi-bidirectional and Push-pull Mode)	I_{SK1}	10	16	20	mA	V_{DD} = 4.5 V, V_S = 0.45 V
	I_{SK1}	7	10	13	mA	V_{DD} = 2.7 V, V_S = 0.45 V
	I_{SK1}	6	9	12	mA	V_{DD} = 2.5 V, V_S = 0.45 V
Brown-Out voltage with BOV_VL [1:0] =00b	$V_{BO2.2}$	2.1	2.2	2.3	V	
Brown-Out voltage with BOV_VL [1:0] =01b	$V_{BO2.7}$	2.6	2.7	2.8	V	
Brown-Out voltage with BOV_VL [1:0] =10b	$V_{BO3.8}$	3.6	3.8	4.0	V	
Brown-Out voltage with BOV_VL [1:0] =11b	$V_{BO4.5}$	4.3	4.5	4.7	V	
Hysteresis range of BOD voltage	V_{BH}	30	-	150	mV	V_{DD} = 2.5 V~5.5 V

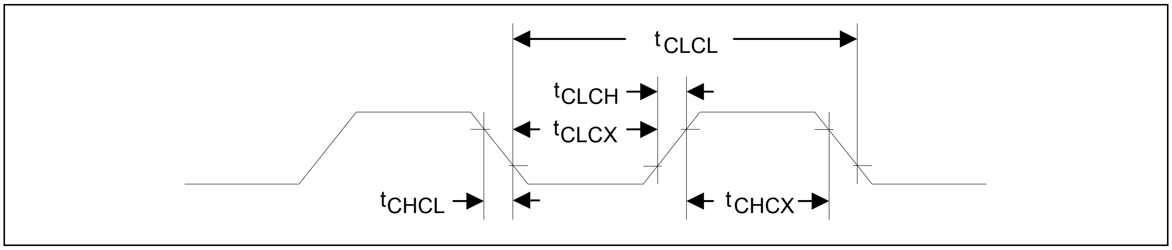
PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Bandgap voltage	V_{BG}	1.20	1.26	1.32	V	V_{DD} = 2.5 V~5.5 V

Note:

1. /RESET pin is a Schmitt trigger input.
2. Crystal Input is a CMOS input.
3. Pins of PA, PB, PC, PD and PE can source a transition current when they are being externally driven from 1 to 0. In the condition of V_{DD}=5.5 V, 5he transition current reaches its maximum value when V_{IN} approximates to 2 V.

7.3 AC Electrical Characteristics

7.3.1 External 4~24 MHz High Speed Oscillator



Note: Duty cycle is 50%.

SYMBOL	PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
t_{CHCX}	Clock High Time		20	-	-	nS
t_{CLCX}	Clock Low Time		20	-	-	nS
t_{CLCH}	Clock Rise Time		-	-	10	nS
t_{CHCL}	Clock Fall Time		-	-	10	nS

7.3.2 External 4~24 MHz High Speed Crystal

PARAMETER	CONDITION	MIN.	TYP..	MAX.	UNIT
Input clock frequency	External crystal	4	12	24	MHz
Temperature	-	-40	-	85	℃
VDD	-	2.5	5	5.5	V

7.3.2.1 Typical Crystal Application Circuits

CRYSTAL	C1	C2	R
4 MHz ~ 24 MHz	without	without	without

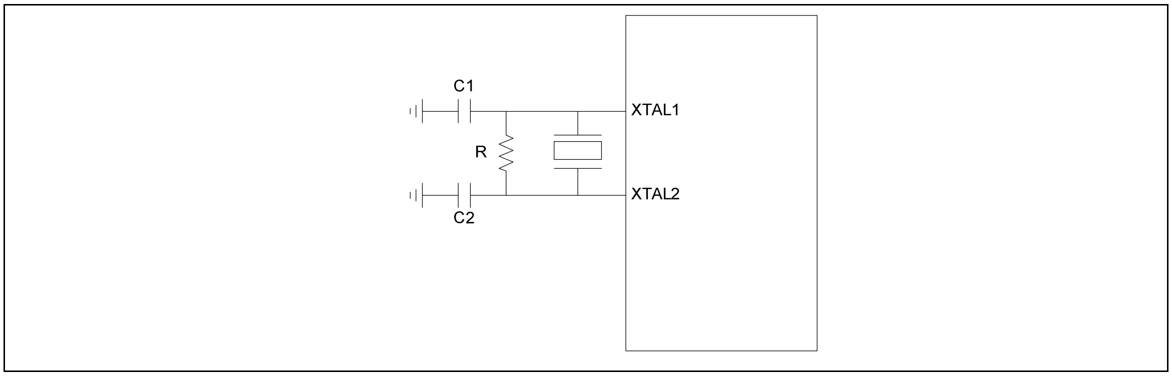


Figure 7-1 Typical Crystal Application Circuit

7.3.3 External 32.768 kHz Low Speed Crystal

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Input clock frequency	External crystal	-	32.768	-	kHz
Temperature	-	-40	-	85	℃
VDD	-	2.5	-	5.5	V

7.3.4 Internal 22.1184 MHz High Speed Oscillator

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Supply voltage[1]	-	2.5	-	5.5	V
Center Frequency	-	-	22.1184	-	MHz
Calibrated Internal Oscillator Frequency	+25 C; V_{DD} =5 V	-1	-	+1	%
	-40 C~+85 C; VDD=2.5 V~5.5 V	-3	-	+3	%
Operation Current	V_{DD} =5 V	-	500	-	uA

7.3.5 Internal 10 kHz Low Speed Oscillator

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Supply voltage[1]	-	2.5	-	5.5	V
Center Frequency	-	-	10	-	kHz
Calibrated Internal Oscillator Frequency	+25 C; V_{DD} =5 V	-30	-	+30	%
	-40 C~+85 C; V_{DD}=2.5 V~5.5 V	-50	-	+50	%

Note: Internal operation voltage comes from LDO.

7.4 Analog Characteristics

7.4.1 Specification of 12-bit SARADC

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT
-	Resolution	-	-	12	Bit
DNL	Differential nonlinearity error	-	±3	-	LSB
INL	Integral nonlinearity error	-	±4	-	LSB
EO	Offset error	-	±1	10	LSB
EG	Gain error (Transfer gain)	-	1	1.005	-
-	Monotonic	Guaranteed			
FADC	ADC clock frequency	-	-	16	MHz
TCAL	Calibration time	-	127	-	Clock
TS	Sample time	-	7	-	Clock
TADC	Conversion time	-	13	-	Clock
FS	Sample rate	-	-	600	K SPS
VLDO	Supply voltage	-	2.5	-	V
VADD		3	-	5.5	V
IDD	Supply current (Avg.)	-	0.5	-	mA
IDDA		-	1.5	-	mA
VREF	Reference voltage	-	VDDA	-	V
IREFP	Reference current (Avg.)	-	1	-	mA
VIN	Reference voltage	0	-	VREF	V
CIN	Capacitance	-	5	-	pF

7.4.2 Specification of LDO and Power management

PARAMETER	MIN.	TYP.	MAX.	UNIT	NOTE
Input Voltage	2.7	5	5.5	V	V_{DD} input voltage
Output Voltage	-10%	2.5	+10%	V	$V_{DD} > 2.7$ V
Temperature	-40	25	85	℃	
Quiescent Current (PD=0)	-	100	-	uA	
Quiescent Current (PD=1)	-	5	-	uA	
Iload (PD=0)	-	-	100	mA	
Iload (PD=1)	-	-	100	uA	
Cbp	-	10	-	uF	Resr=1ohm

Note:

1. It is recommended that a 10uF or higher capacitor and a 100nF bypass capacitor are connected between VDD and the closest VSS pin of the device.

2. For ensuring power stability, a 10uF or higher capacitor must be connected between LDO pin and the closest VSS pin of the device.

7.4.3 Specification of Low Voltage Reset

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Operation voltage	-	1.7	-	5.5	V
Quiescent current	VDD5V=5.5 V	-	-	5	uA
Temperature	-	-40	25	85	℃
Threshold voltage	Temperature=25°	1.7	2.0	2.3	V
	Temperature=-40°	-	2.4	-	V
	Temperature=85°	-	1.6	-	V
Hysteresis	-	0	0	0	V

7.4.4 Specification of Brown-Out Detector

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Operation voltage	-	2.5	-	5.5	V
Quiescent current	AVDD=5.5 V	-	-	125	μA
Temperature	-	-40	25	85	℃
Brown-out voltage	BOV_VL[1:0]=11	4.3	4.5	4.7	V
	BOV_VL [1:0]=10	3.6	3.8	4.0	V
	BOV_VL [1:0]=01	2.6	2.7	2.8	V
	BOV_VL [1:0]=00	2.1	2.2	2.3	V
Hysteresis	-	30	-	150	mV

7.4.5 Specification of Power-On Reset (5 V)

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Temperature	-	-40	25	85	℃
Reset voltage	V+	-	2	-	V
Quiescent current	Vin>reset voltage	-	1	-	nA

7.4.6 Specification of Temperature Sensor

PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
Supply voltage[1]		2.5	-	5.5	V
Temperature		-40	-	125	℃
Current consumption		6.4	-	10.5	uA
Gain			-1.76		mV/℃
Offset	Temp=0 ℃		720		mV

Note: Internal operation voltage comes from LDO.

7.4.7 Specification of Comparator

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Temperature	-	-40	25	85	℃
VDD	-	2.4	3	5.5	V
VDD current	20 uA@VDD=3 V	-	20	40	uA
Input offset voltage	-	-	5	15	mV
Output swing	-	0.1	-	VDD-0.1	V
Input common mode range	-	0.1	-	VDD-1.2	V
DC gain	-	-	70	-	dB
Propagation delay	@VCM=1.2 V and VDIFF=0.1 V	-	200	-	ns
Comparison voltage	20 mV@VCM=1 V 50 mV@VCM=0.1 V 50 mV@VCM=VDD-1.2 @10 mV for non-hysteresis	10	20	-	mV
Hysteresis	One bit control W/O and W. hysteresis @VCM=0.4 V ~ VDD-1.2 V	-	±10	-	mV
Wake-up time	@CINP=1.3 V CINN=1.2 V	-	-	2	us

7.4.8 Specification of USB PHY

7.4.8.1 USB DC Electrical Characteristics

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
V_{IH}	Input high (driven)		2.0			V
V_{IL}	Input low				0.8	V
V_{DI}	Differential input sensitivity	|PADP-PADM|	0.2			V
V_{CM}	Differential common-mode range	Includes V_{DI} range	0.8		2.5	V
V_{SE}	Single-ended receiver threshold		0.8		2.0	V
	Receiver hysteresis			200		mV
V_{OL}	Output low (driven)		0		0.3	V
V_{OH}	Output high (driven)		2.8		3.6	V
V_{CRS}	Output signal cross voltage		1.3		2.0	V
R_{PU}	Pull-up resistor		1.425		1.575	kΩ
R_{PD}	Pull-down resistor		14.25		15.75	kΩ
V_{TRM}	Termination Voltage for upstream port pull up (RPU)		3.0		3.6	V
Z_{DRV}	Driver output resistance	Steady state drive*		10		Ω
C_{IN}	Transceiver capacitance	Pin to GND			20	pF

*Driver output resistance doesn’t include series resistor resistance.

7.4.8.2 USB Full-Speed Driver Electrical Characteristics

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
T_{FR}	Rise Time	C_L=50p	4		20	ns
T_{FF}	Fall Time	C_L=50p	4		20	ns
T_{FRFF}	Rise and fall time matching	$T_{FRFF}=T_{FR}/T_{FF}$	90		111.11	%

7.4.8.3 USB Power Dissipation

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
I_{VDDREG} (Full Speed)	VDDD and VDDREG Supply Current (Steady State)	Standby		50		uA
		Input mode				uA
		Output mode				uA

7.5 SPI Dynamic Characteristics

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT
SPI master mode (VDD = 4.5V ~ 5.5V, 30pF loading Capacitor)					
t_{DS}	Data setup time	4	2	-	ns
t_{DH}	Data hold time	0	-	-	ns
t_V	Data output valid time	-	7	11	ns
SPI master mode (VDD = 3.0V ~ 3.6V, 30pF loading Capacitor)					
t_{DS}	Data setup time	5	3	-	ns
t_{DH}	Data hold time	0	-	-	ns
t_V	Data output valid time	-	13	18	ns
SPI slave mode (VDD = 4.5V ~ 5.5V, 30pF loading Capacitor)					
t_{DS}	Data setup time	0	-	-	ns
t_{DH}	Data hold time	2*PCLK+4	-	-	ns
t_V	Data output valid time	-	2*PCLK+11	2*PCLK+19	ns
SPI slave mode (VDD = 3.0V ~ 3.6V, 30pF loading Capacitor)					
t_{DS}	Data setup time	0	-	-	ns
t_{DH}	Data hold time	2*PCLK+6	-	-	ns
t_V	Data output valid time	-	2*PCLK+19	2*PCLK+25	ns

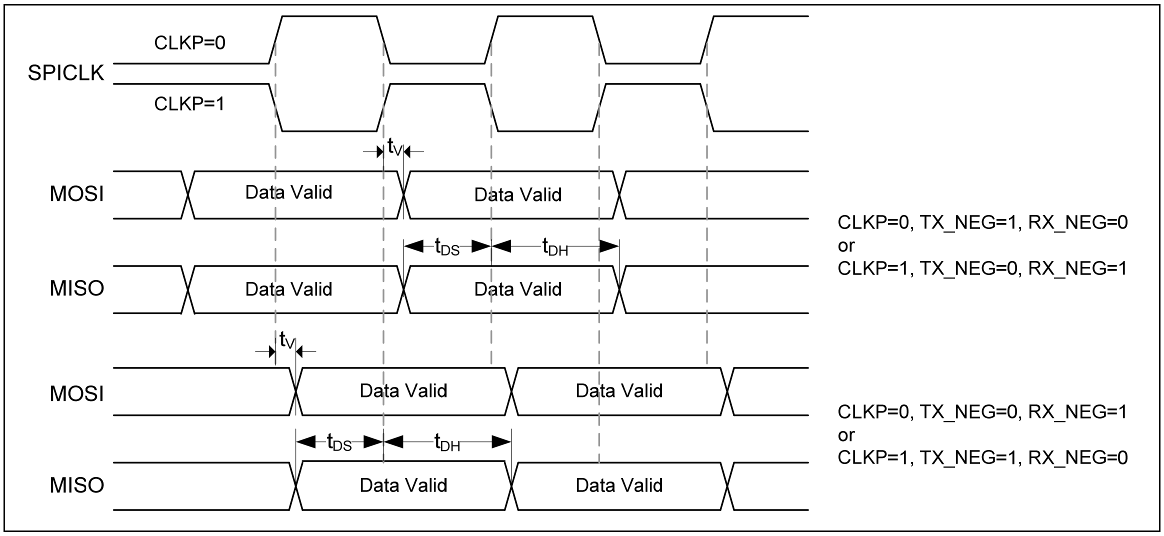


Figure 7-2 SPI Master dynamic characteristics tiMINg

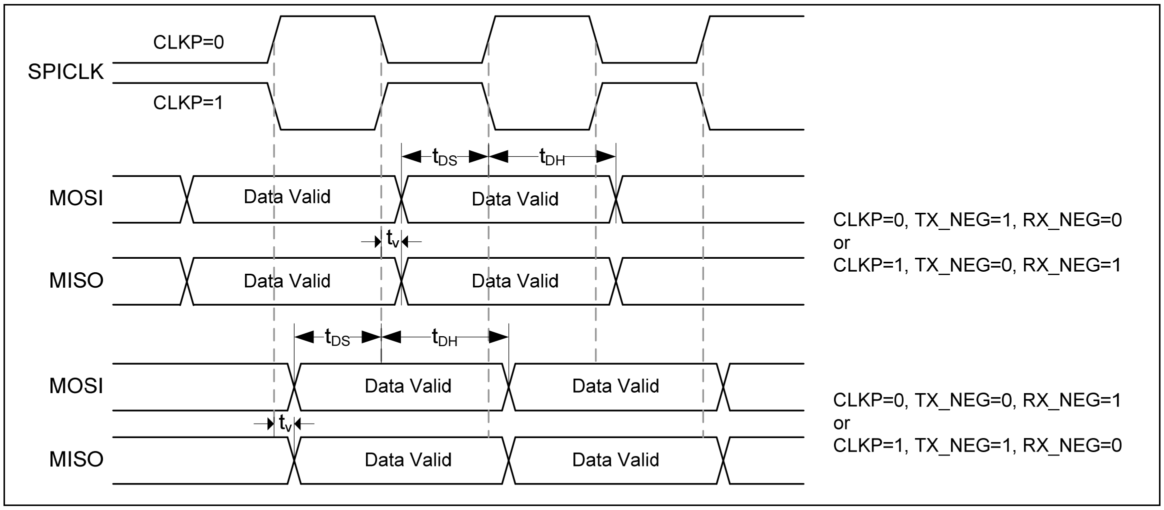
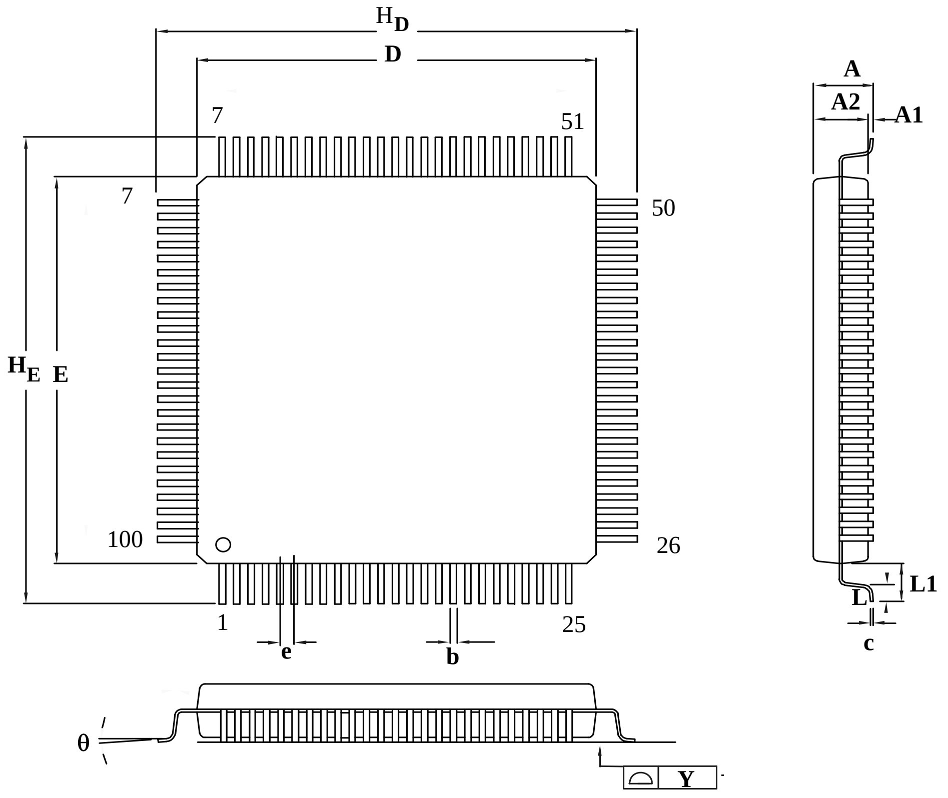


Figure 7-3 SPI Slave dynamic characteristics timing

8 PACKAGE DIMENSIONS

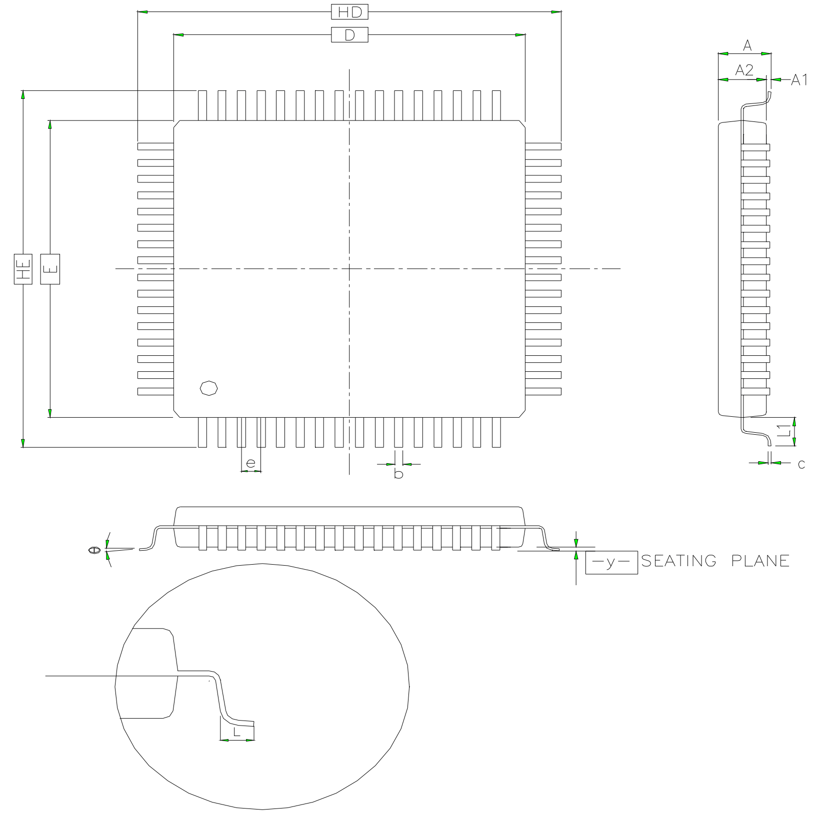
8.1 100L LQFP (14x14x1.4 mm footprint 2.0mm)



Controlling Dimension : Millimeters

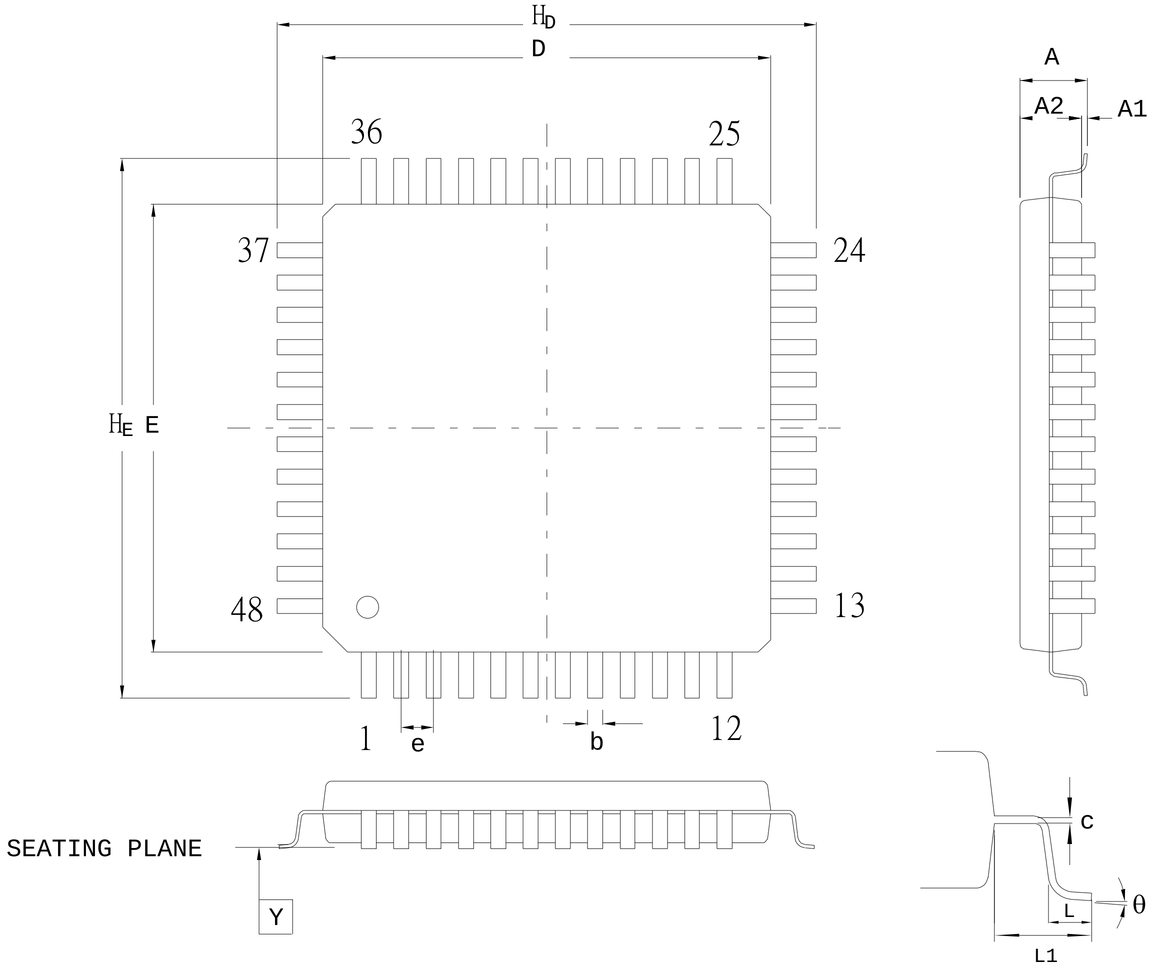
Symbol	Dimension in inch			Dimension in mm		
	Min	Nom	Max	Min	Nom	Max
A	—	—	0.063	—	—	1.60
A1	0.002	—	—	0.05	—	—
A	0.053	0.055	0.057	1.35	1.40	1.45
b	0.007	0.009	0.011	0.17	0.22	0.27
c	0.004	0.006	0.008	0.10	0.15	0.20
D	0.547	0.551	0.556	13.90	14.00	14.10
E	0.547	0.551	0.556	13.90	14.00	14.10
e	—	0.020	—	—	0.50	—
H_D	0.622	0.630	0.638	15.80	16.00	16.20
H_E	0.622	0.630	0.638	15.80	16.00	16.20
L	0.018	0.024	0.030	0.45	0.60	0.75
L1	—	0.039	—	—	1.00	—
y	—	—	0.004	—	—	0.10
θ	0°	—	7°	0°	—	7°

8.2 64L LQFP (10x10x1.4mm footprint 2.0 mm)



Symbol	Dimension in inch			Dimension in mm		
	Min	Nom	Max	Min	Nom	Max
A	—	—	0.063	—	—	1.60
A_1	0.002	—	0.006	0.05	—	0.15
A_2	0.053	0.055	0.057	1.35	1.40	1.45
b	0.007	0.008	0.011	0.17	0.20	0.27
c	0.004	—	0.008	0.09	—	0.20
D	—	0.393	—	—	10.00	—
E	—	0.393	—	—	10.00	—
$\boxed{e}$	—	0.020	—	—	0.50	—
H_D	—	0.472	—	—	12.00	—
H_E	—	0.472	—	—	12.00	—
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°	3.5°	7°	0°	3.5°	7°

8.3 48L LQFP (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 °

9 REVISION HISTORY

VERSION	DATE	PAGE/ CHAPTER	DESCRIPTION
V2.00	May 04, 2011	-	1. This document is divided from "NUC100 series TRM.doc" 2. Update the CAN peripheral to BOSCH C-CAN
V2.01	June 14, 2011	6.10 7.4.6 3.4 7.5 7.4.4	1. Revise ISPCON.BS and ISPCON.ISPFF description 2. modify temperature sensor spec 3. Revise Pin description position for multi-function T2EX, T3EX, nRD, nWR 4. update title of SPI Dynamic Characteristics 5. update BOD spec

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.