NuMicro™ Family
NUC123 Series
Technical Reference Manual

The information described in this document is the exclusive intellectual property of Nuvoton Technology Corporation and shall not be reproduced without permission from Nuvoton.

Nuvoton is providing this document only for reference purposes of NuMicro microcontroller based system design. Nuvoton assumes no responsibility for errors or omissions.

All data and specifications are subject to change without notice.

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

www.nuvoton.com

TABLE OF CONTENTS

List of Figures

List of Tables

1 GENERAL DESCRIPTION

The NuMicro™ NUC123 series 32-bit microcontrollers are embedded with Cortex™-M0 core running up to 72 MHz, up to 36K/68K-byte embedded flash, 12K/20K-byte embedded SRAM, and 4K-byte loader ROM for the ISP. It also integrates Timers, Watchdog Timer, Windowed Watchdog Timer, PDMA with CRC calculation unit, UART, SPI/MICROWIRE, I^{2}C, I^{2}S, PWM Timer, GPIO, PS/2, USB 2.0 FS Device, 10-bit ADC, Low Voltage Reset Controller and Brown-out Detector.

Product Line	UART	SPI	I^{2}C	USB	LIN	CAN	PS/2	I^{2}S
NUC123	●	●	●	●	-	-	●	●

Table 1-1 Connectivity Support Table

2 FEATURES

2.1 NuMicro™ NUC123 Features

Core

- ARM® Cortex™-M0 core runs up to 72 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
- Supports Serial Wire Debug with 2 watchpoints/4 breakpoints

Built-in LDO for wide operating voltage ranges from 2.5 V to 5.5 V

Flash Memory

- 36K/68K bytes Flash for program code
- 4KB flash for ISP loader
- Supports In-system program (ISP) application code update
- 512 byte page erase for flash
- Configurable data flash address and size for both 36KB and 68KB system
- Supports 2 wire ICP update through SWD/ICE interface
- Supports fast parallel programming mode by external programmer

SRAM Memory

- 12K/20K bytes embedded SRAM
- Supports PDMA mode

PDMA (Peripheral DMA)

- Supports 6 channels PDMA for automatic data transfer between SRAM and peripherals such as SPI, UART, I^2S, USB 2.0 FS device, PWM and ADC
- Supports CRC calculation with four common polynomials, CRC-CCITT, CRC-8, CRC-16 and CRC-32

Clock Control

- Flexible selection for different applications
- Built-in 22.1184 MHz high speed oscillator (Trimmed to 1%) for system operation, and low power 10 kHz low speed oscillator for watchdog and wake-up operation
- Supports one PLL, up to 144 MHz, for high performance system operation
- External 4~24 MHz high speed crystal input for precise timing 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 configured as interrupt source with edge/level setting
- Supports High Driver and High Sink IO mode

Timer

- Supports 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
- Supports event counting function

Watchdog/Windowed-Watchdog Timer

- Multiple clock sources
- 8 selectable time out period from 1.6 ms ~ 26.0 sec (depending on clock source)
- Wake-up from Power-down or Idle mode
- Interrupt or reset selectable on watchdog timer time-out
- Interrupt on windowed-watchdog timer time-out
- Reset on windowed-watchdog timer time out or reload in an unexpected time window

PWM/Capture

- Up to two built-in 16-bit PWM generators provide four PWM outputs or two 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 four 16-bit digital Capture timers (shared with PWM timers) providing four rising/falling capture inputs
- Supports capture interrupt

UART

- Up to two UART controllers
- UART ports with flow control (TXD, RXD, CTS and RTS)
- UART0/1 with 16-byte FIFO for standard device
- Supports IrDA (SIR) function
- Supports RS-485 9-bit mode and direction control
- Programmable baud-rate generator up to 1/16 system clock
- Supports PDMA mode

SPI

- Up to three sets of SPI controller

- Master up to 32 MHz, and Slave up to 16 MHz (chip working at 3.3V)

- Supports SPI master/Slave mode
- Full duplex synchronous serial data transfer
- Variable length of transfer data from 8 to 32 bits
- MSB or LSB first data transfer
- Two slave/device select lines when it is selected as the master, and one slave/device select line when it is selected as the slave
- Supports Byte Suspend mode in 16/24/32-bit transmission
- Supports PDMA mode

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 allowing devices with different bit rates to communicate

via one serial bus
- Serial clock synchronization used as a handshake mechanism to suspend and resume serial transfer
- Programmable clocks allow versatile rate control
- Supports multiple address recognition (four slave address with mask option)
- Supports wake-up by address recognition (for 1st slave address only)

I^2S

- Interface with external audio CODEC
- Operate as either master or Slave mode
- Capable of handling 8-, 16-, 24- and 32-bit word sizes
- Supports Mono and stereo audio data
- Supports I^2S and MSB justified data format
- Provides two 8 word FIFO data buffers, one for transmitting and the other for receiving
- Generates interrupt requests when buffer levels cross a programmable boundary
- Supports two DMA requests, one for transmitting and the other for receiving

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
- Provides 1 interrupt source with 4 interrupt events
- Supports Control, Bulk In/Out, Interrupt and Isochronous transfers
- Auto suspend function when no bus signaling for 3 ms
- Provides 8 programmable endpoints
- Includes 512 bytes internal SRAM as USB buffer
- Provides remote wake-up capability

ADC

- 10-bit SAR ADC with 150K SPS
- Up to 8-ch single-end 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
- Supports PDMA mode

Brown-out detector

- With 4 levels: 4.4 V/3.7 V/2.7 V/2.2 V
- Supports Brown-out Interrupt and Reset option

Low Voltage Reset

- Threshold voltage levels: 2.0 V

One built-in LDO

Operating Temperature: -40℃~85℃

Packages:

- All Green package (RoHS)
- LQFP 64-pin
- LQFP 48-pin
- QFN 33-pin

3 PARTS INFORMATION LIST AND PIN CONFIGURATION

3.1 NuMicro™ NUC123xxxANx Selection Guide

3.1.1 NuMicro™ NUC123 Selection Guide

Part number	Flash	SRAM	ISP ROM	I/O	Timer	Connectivity						I^2S	Comp.	PWM	ADC	RTC	EBI	ISP ICP IAP	Package
						UART	SPI	I^2C	USB	LIN	PS/2								
NUC123ZD4AN0	68 KB	20 KB	4 KB	Up to 20	4x32-bit	1	3	1	1	-	-	1	-	2	-	-	-	v	QFN33
NUC123ZC2AN1	36 KB	12 KB	4 KB	up to 20	4x32-bit	1	3	1	1	-	-	1	-	2	-	-	-	v	QFN33
NUC123LD4AN0	68 KB	20 KB	4 KB	up to 36	4x32-bit	2	3	2	1	-	1	1	-	4	8x10-bit	-	-	v	LQFP48
NUC123LC2AN1	36 KB	12 KB	4 KB	up to 36	4x32-bit	2	3	2	1	-	1	1	-	4	8x10-bit	-	-	v	LQFP48
NUC123SD4AN0	68 KB	20 KB	4 KB	up to 47	4x32-bit	2	3	2	1	-	1	1	-	4	8x10-bit	-	-	v	LQFP64
NUC123SC2AN1	36 KB	12 KB	4 KB	up to 47	4x32-bit	2	3	2	1	-	1	1	-	4	8x10-bit	-	-	v	LQFP64

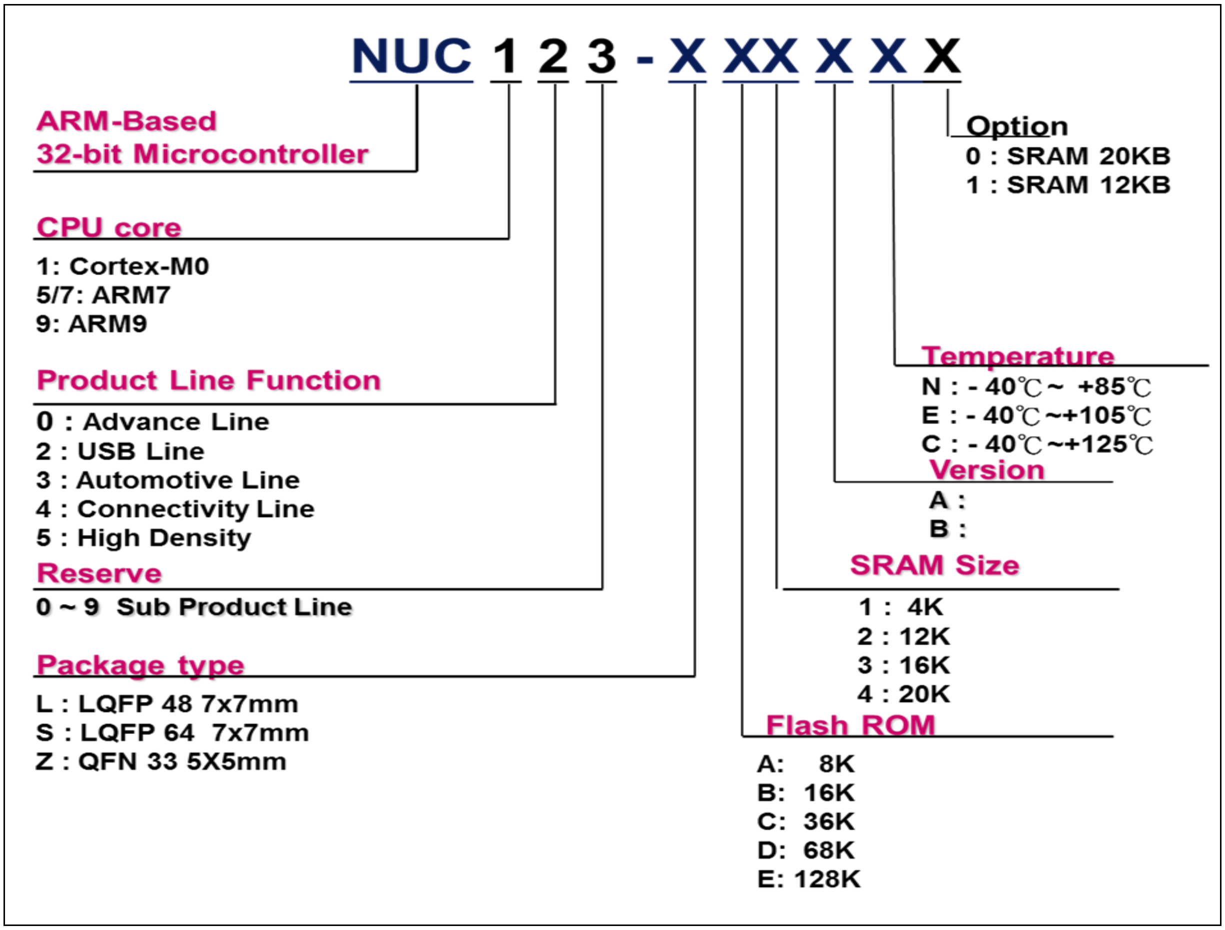


Figure 3-1 NuMicro™ NUC123 Series Selection Code

3.2 Pin Configuration

3.2.1 NuMicro™ NUC123 Pin Diagram

3.2.1.1 *NuMicro™ NUC123SxxANx LQFP 64 pin*

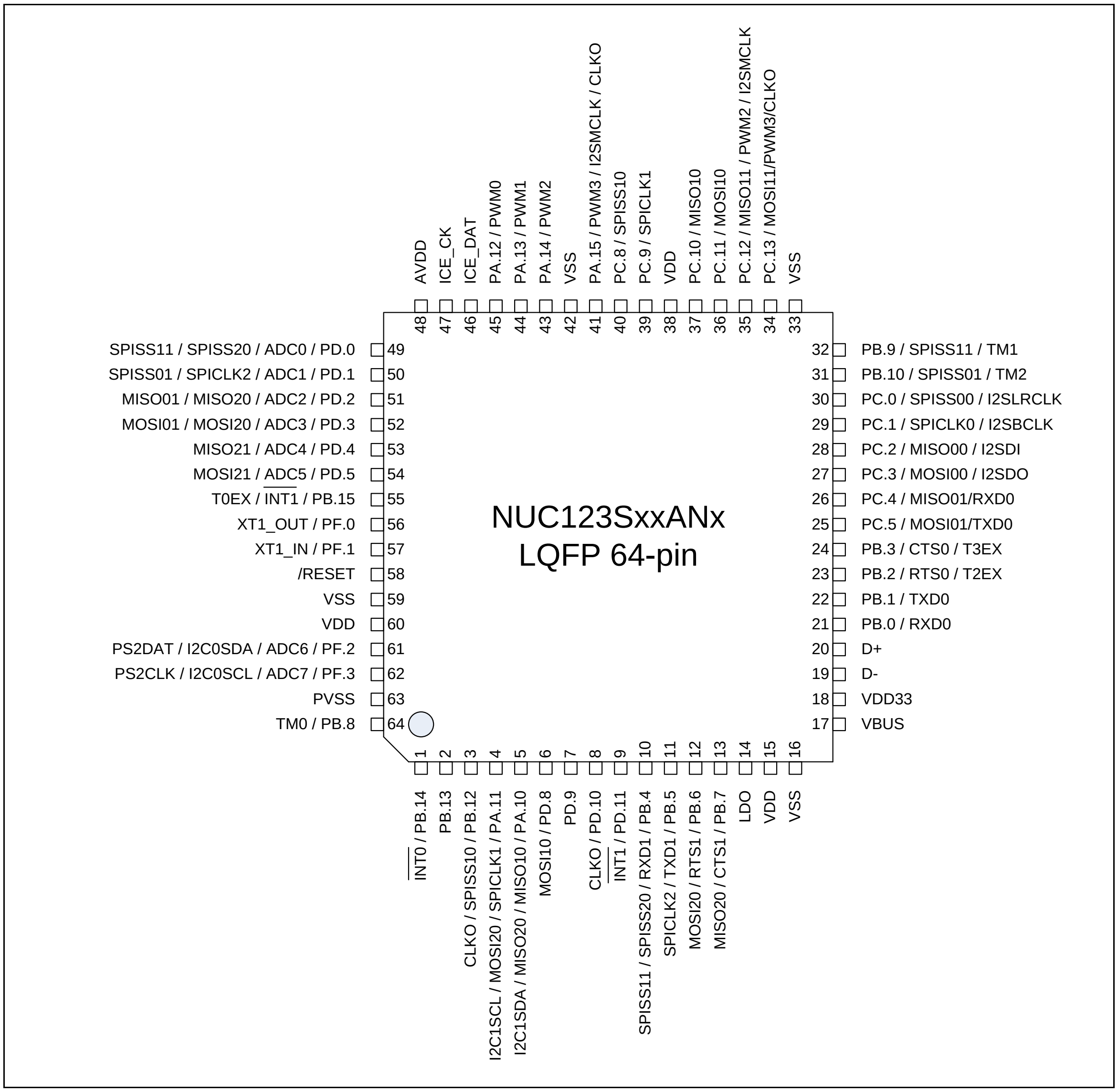


Figure 3-2 NuMicro™ NUC123SxxANx LQFP 64-pin Assignment

3.2.1.2 *NuMicro™ NUC123LxxANx LQFP 48 pin*

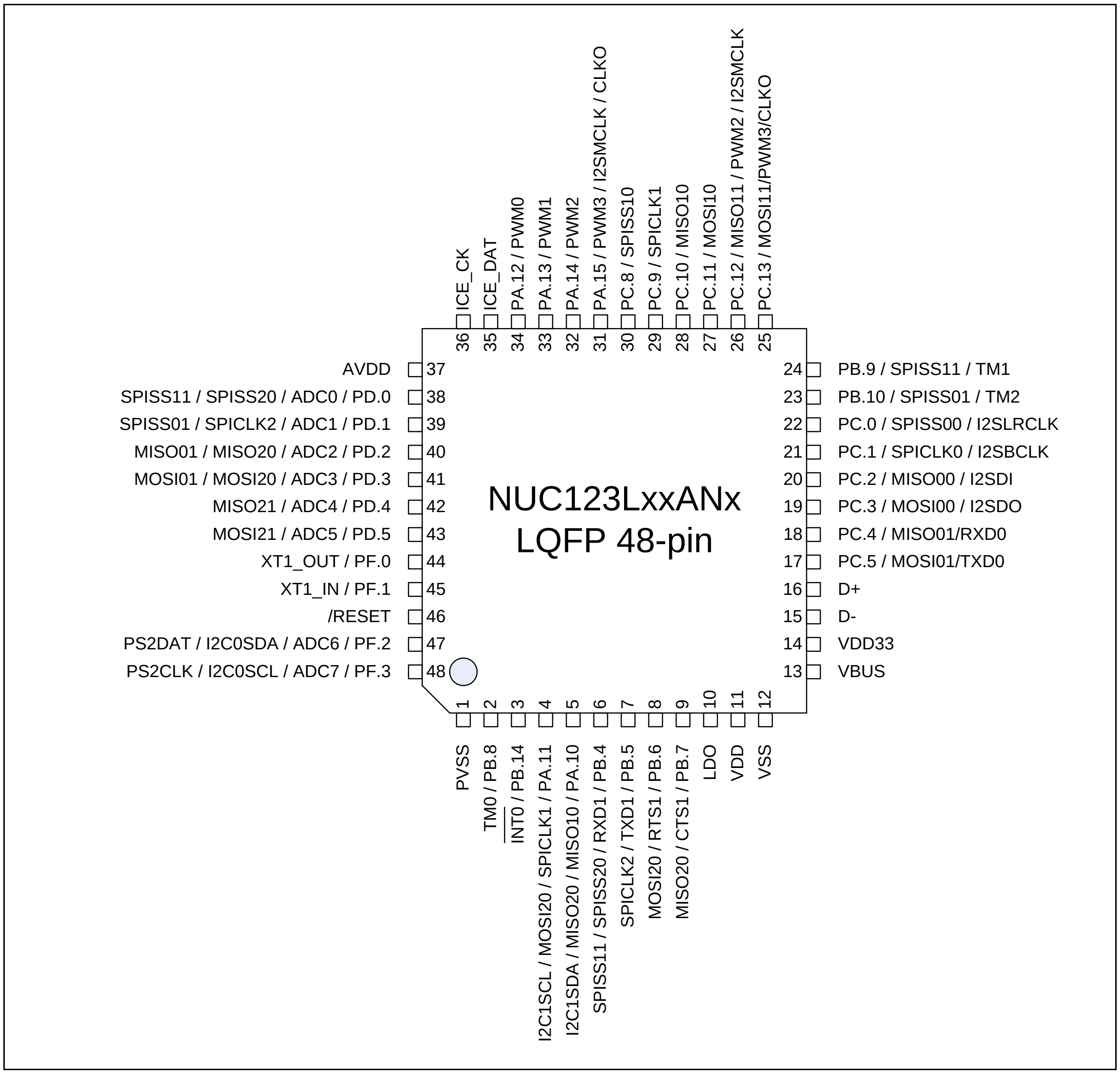


Figure 3-3 NuMicro™ NUC123LxxANx LQFP 48-pin Assignment

3.2.1.3 *NuMicro™ NUC123ZxxANx QFN 33 pin*

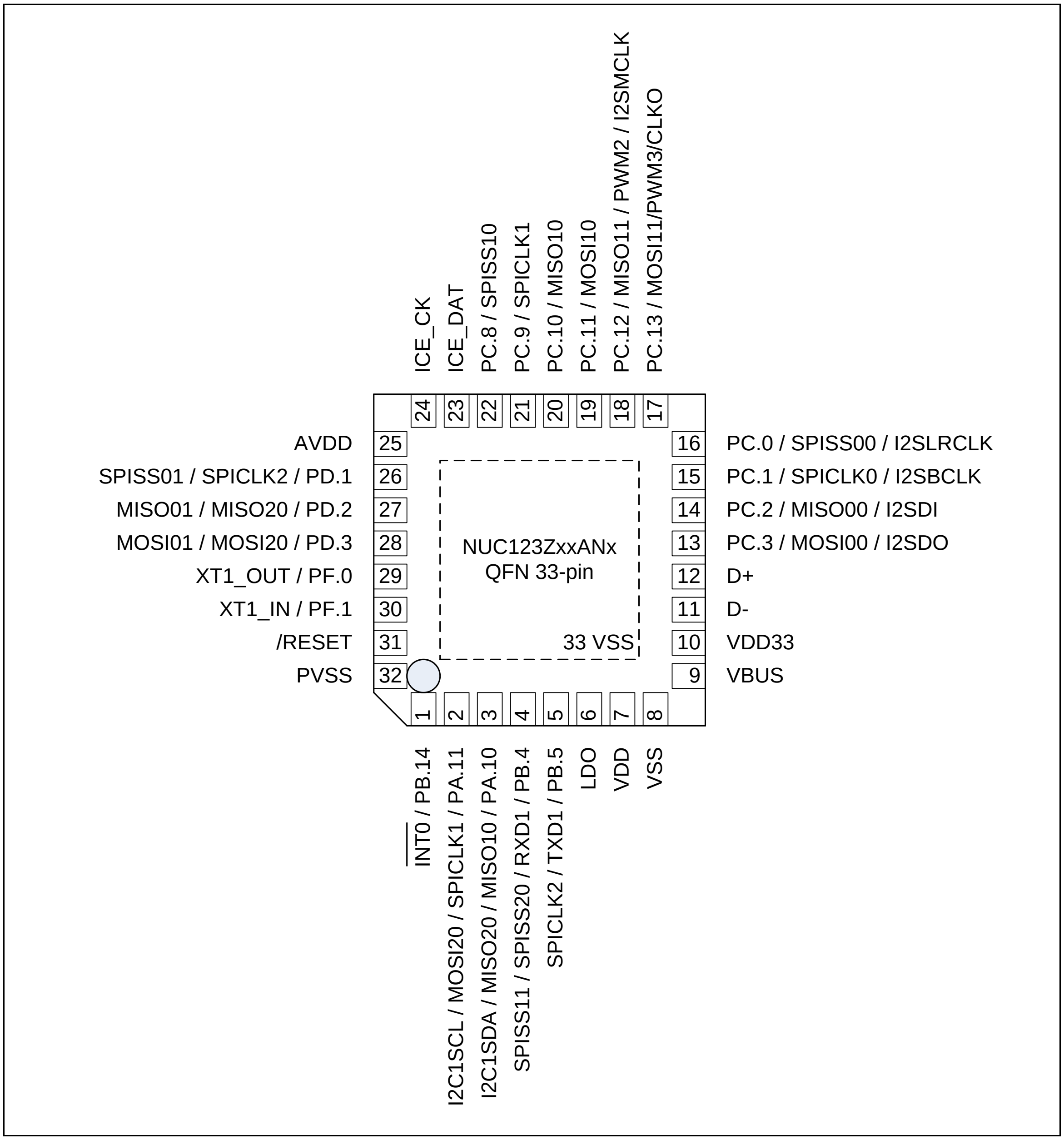


Figure 3-4 NuMicro™ NUC123ZxxANx QFN 33-pin Assignment

3.2.2 Pin Description

3.2.3 NuMicro™ NUC123 Pin Description

Pin No.			Pin Name	Type	Description
LQFP 64-pin	LQFP 48-pin	QFN 33-pin			
1	3	1	PB.14	I/O	Digital GPIO pin
			/INT0	I	External interrupt 0 input pin
2			PB.13	I/O	Digital GPIO pin
3			PB.12	I/O	Digital GPIO pin
			SPISS10	I/O	SPI1 1st slave select pin
			CLKO	O	Frequency Divider output pin
4	4	2	PA.11	I/O	Digital GPIO pin
			SPICLK1	I/O	SPI1 serial clock pin
			MOSI20	I/O	SPI2 1st MOSI (Master Out, Slave In) pin
			I2C1SCL	I/O	I^{2}C1 clock pin
5*	5*	3*	PA.10	I/O	Digital GPIO pin
			MISO10	I/O	SPI1 1st MISO (Master In, Slave Out) pin
			MISO20	I/O	SPI2 1st MISO (Master In, Slave Out) pin
			I2C1SDA	I/O	I^{2}C1 data input/output pin
6			PD.8	I/O	Digital GPIO pin
			MOSI10	I/O	SPI1 1st MOSI (Master Out, Slave In) pin
7			PD.9	I/O	Digital GPIO pin
8			PD.10	I/O	Digital GPIO pin
			CLKO	O	Frequency Divider output pin
9			PD.11	I/O	Digital GPIO pin
			/INT1	I	External interrupt 1 input pin
10	6	4	PB.4	I/O	Digital GPIO pin
			RXD1	I	UART1 data receiver input pin
			SPISS20	I/O	SPI2 1st slave select pin
			SPISS11	I/O	SPI1 2nd slave select pin
11	7	5	PB.5	I/O	Digital GPIO pin

			TXD1	**O**	UART1 data transmitter output pin
			SPICLK2	**I/O**	SPI2 serial clock pin
12	8		PB.6	**I/O**	Digital GPIO pin
			RTS1	**O**	UART1 request to send output pin
			MOSI20	**I/O**	SPI2 1st MOSI (Master Out, Slave In) pin
13	9		PB.7	**I/O**	Digital GPIO pin
			CTS1	**I**	UART1 clear to send input pin
			MISO20	**I/O**	SPI2 1st MISO (Master In, Slave Out) pin
14	10	6	LDO	**P**	LDO output pin
15	11	7	VDD	**P**	Power supply for I/O ports and LDO source for internal PLL and digital function. Voltage range is 2.5-V ~ 5-V.
16	12	8	VSS	**P**	Ground
17	13	9	VBUS	**USB**	Power supply from USB host or hub
18	14	10	VDD33	**USB**	Internal power regulator output 3.3-V decoupling pin
19	15	11	D-	**USB**	USB differential signal D-
20	16	12	D+	**USB**	USB differential signal D+
21			PB.0	**I/O**	Digital GPIO pin
			RXD0	**I**	UART0 data receiver input pin
22			PB.1	**I/O**	Digital GPIO pin
			TXD0	**O**	UART0 data transmitter output pin
23			PB.2	**I/O**	Digital GPIO pin
			RTS0	**O**	UART0 request to send output pin
			T2EX	**I**	Timer2 external capture input pin
24			PB.3	**I/O**	Digital GPIO pin
			CTS0	**I**	UART0 clear to send input pin
			T3EX	**I**	Timer3 external capture input pin
25	17		PC.5	**I/O**	Digital GPIO pin
			MOSI01	**I/O**	SPI0 2nd MOSI (Master Out, Slave In) pin
			TXD0	**O**	UART0 data transmitter output pin
26	18		PC.4	**I/O**	Digital GPIO pin
			MISO01	**I/O**	SPI0 2nd MISO (Master In, Slave Out) pin

			RXD0	**I**	UART0 data receiver input pin
27	19	13	PC.3	**I/O**	Digital GPIO pin
			MOSI00	**I/O**	SPI0 1st MOSI (Master Out, Slave In) pin
			I2SDO	**O**	I^{2}S data output pin
28	20	14	PC.2	**I/O**	Digital GPIO pin
			MISO00	**I/O**	SPI0 1st MISO (Master In, Slave Out) pin
			I2SDI	**I**	I^{2}S data input pin
29	21	15	PC.1	**I/O**	Digital GPIO pin
			SPICLK0	**I/O**	SPI0 serial clock pin
			I2SBCLK	**I/O**	I^{2}S bit clock pin
30	22	16	PC.0	**I/O**	Digital GPIO pin
			SPISS00	**I/O**	SPI0 1st slave select pin
			I2SLRCLK	**I/O**	I^{2}S left/right channel clock pin
31	23		PB.10	**I/O**	Digital GPIO pin
			SPISS01	**I/O**	SPI0 2nd slave select pin
			TM2	**I/O**	Timer2 event counter input / toggle output pin
32	24		PB.9	**I/O**	Digital GPIO pin
			SPISS11	**I/O**	SPI1 2nd slave select pin
			TM1	**I/O**	Timer1 event counter input / toggle output pin
33			VSS	**P**	Ground
34	25	17	PC.13	**I/O**	Digital GPIO pin
			MOSI11	**I/O**	SPI1 2nd MOSI (Master Out, Slave In) pin
			PWM3	**I/O**	PWM3 PWM output / capture input pin
			CLKO	**O**	Frequency Divider output pin
35	26	18	PC.12	**I/O**	Digital GPIO pin
			MISO11	**I/O**	SPI1 2nd MISO (Master In, Slave Out) pin
			PWM2	**I/O**	PWM2 PWM output / capture input pin
			I2SMCLK	**O**	I^{2}S master clock output pin
36	27	19	PC.11	**I/O**	Digital GPIO pin
			MOSI10	**I/O**	SPI1 1st MOSI (Master Out, Slave In) pin
37	28	20	PC.10	**I/O**	Digital GPIO pin

			MISO10	**I/O**	SPI1 1st MISO (Master In, Slave Out) pin
38			VDD	**P**	Power supply for I/O ports and LDO source for internal PLL and digital function. Voltage range is 2.5-V ~ 5-V.
39	29	21	PC.9	**I/O**	Digital GPIO pin
			SPICLK1	**I/O**	SPI1 serial clock pin
40	30	22	PC.8	**I/O**	Digital GPIO pin
			SPISS10	**I/O**	SPI1 1st slave select pin
41	31		PA.15	**I/O**	Digital GPIO pin
			PWM3	**I/O**	PWM3 PWM output / capture input pin
			I2SMCLK	**O**	I^2S master clock output pin
			CLKO	**O**	Frequency Divider output pin
42			VSS	**P**	Ground
43	32		PA.14	**I/O**	Digital GPIO pin
			PWM2	**I/O**	PWM2 PWM output / capture input pin
44	33		PA.13	**I/O**	Digital GPIO pin
			PWM1	**I/O**	PWM1 PWM output / capture input pin
45	34		PA.12	**I/O**	Digital GPIO pin
			PWM0	**I/O**	PWM0 PWM output / capture input pin
46	35	23	ICE_DAT	**I/O**	Serial wired debugger data pin
47	36	24	ICE_CK	**I**	Serial wired debugger clock input pin
48	37		AVDD	**AP**	Power supply for internal analog circuit
49	38	25	PD.0	**I/O**	Digital GPIO pin
			ADC0	**AI**	ADC channel 0 analog input pin
			SPISS20	**I/O**	SPI2 1st slave select pin
			SPISS11	**I/O**	SPI1 2nd slave select pin
50	39	26	PD.1	**I/O**	Digital GPIO pin
			ADC1	**AI**	ADC channel 1 analog input pin
			SPICLK2	**I/O**	SPI2 serial clock pin
			SPISS01	**I/O**	SPI0 2nd slave select pin
51	40	27	PD.2	**I/O**	Digital GPIO pin
			ADC2	**AI**	ADC channel 2 analog input pin

			MISO20	**I/O**	SPI2 1st MISO (Master In, Slave Out) pin
			MISO01	**I/O**	SPI0 2nd MISO (Master In, Slave Out) pin
52	41	28	PD.3	**I/O**	Digital GPIO pin
			ADC3	**AI**	ADC channel 3 analog input pin
			MOSI20	**I/O**	SPI2 1st MOSI (Master Out, Slave In) pin
			MOSI01	**I/O**	SPI0 2nd MOSI (Master Out, Slave In) pin
53	42		PD.4	**I/O**	Digital GPIO pin
			ADC4	**AI**	ADC channel 4 analog input pin
			MISO21	**I/O**	SPI2 2nd MISO (Master In, Slave Out) pin
54	43		PD.5	**I/O**	Digital GPIO pin
			ADC5	**AI**	ADC channel 5 analog input pin
			MOSI21	**I/O**	SPI2 2nd MOSI (Master Out, Slave In) pin
55			PB.15	**I/O**	Digital GPIO pin
			/INT1	**I**	External interrupt 1 input pin
			T0EX	**I**	Timer0 external capture input pin
56	44	29	PF.0	**I/O**	Digital GPIO pin
			XT1_OUT	**O**	External 4~24 MHz high speed crystal output pin
57	45	30	PF.1	**I/O**	Digital GPIO pin
			XT1_IN	**I**	External 4~24 MHz high speed crystal input pin
58	46	31	/RESET	**I**	External reset input: Low active, set this pin low reset chip to initial state. With internal pull-up.
59			VSS	**P**	Ground
60			VDD	**P**	Power supply for I/O ports and LDO source for internal PLL and digital circuit. Voltage range is 2.5 V ~ 5-V.
61	47		PF.2	**I/O**	Digital GPIO pin
			ADC6	**AI**	ADC channel 6 analog input pin
			I2C0SDA	**I/O**	I^{2}C0 data input/output pin
			PS2DAT	**I/O**	PS/2 data pin
62	48		PF.3	**I/O**	Digital GPIO pin
			ADC7	**AI**	ADC channel 7 analog input pin
			I2C0SDA	**I/O**	I^{2}C0 clock pin
			PS2CLK	**I/O**	PS/2 clock pin

63	1	32	PVSS	**P**	PLL ground
64	2		PB.8	**I/O**	Digital GPIO pin
			TM0	**I/O**	Timer2 event counter input / toggle output pin

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

4 BLOCK DIAGRAM

4.1 NuMicro™ NUC123 Block Diagram

4.1.1 NuMicro™ NUC123 Block Diagram

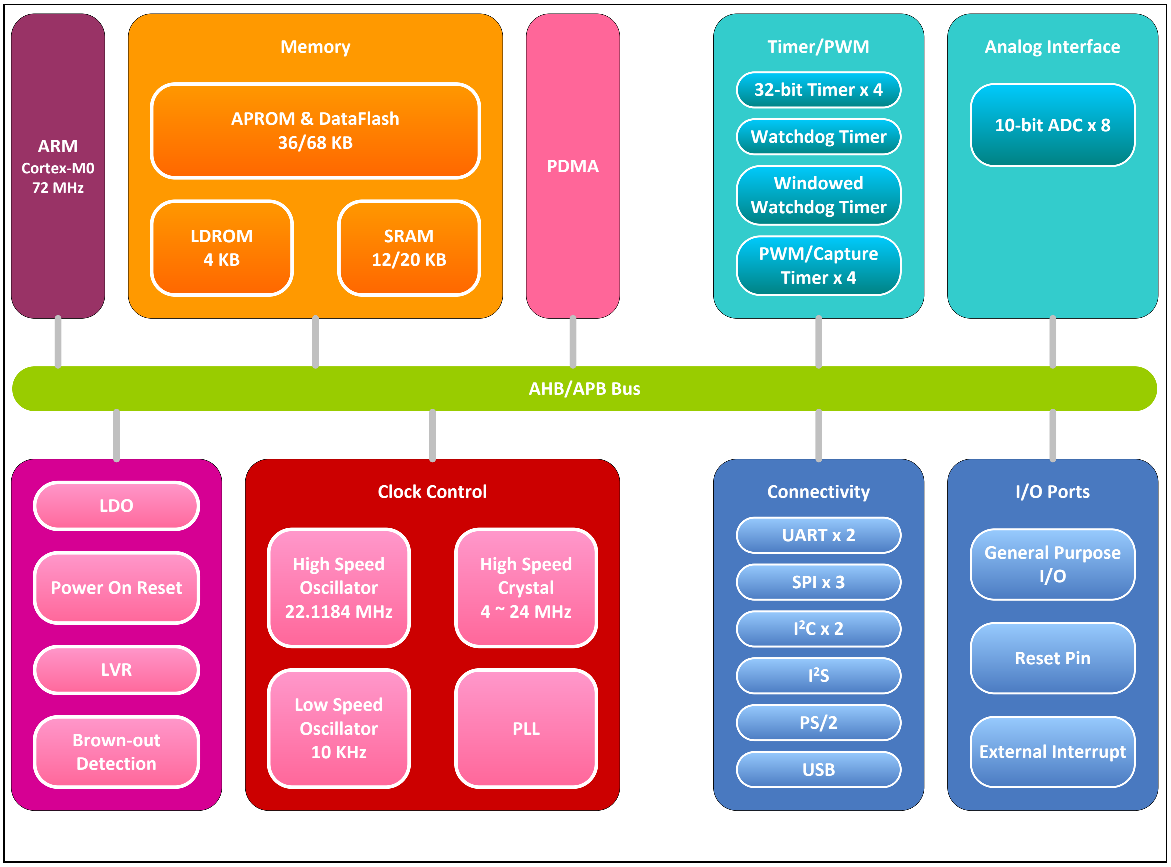


Figure 4-1 NuMicro™ NUC123 Block Diagram

5 FUNCTIONAL DESCRIPTION

5.1 Memory Organization

5.1.1 Overview

The NuMicro™ NUC123 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. The NuMicro™ NUC123 Series only supports little-endian data format.

5.1.2 System Memory Map

Address Space	Token	Controllers
Flash and SRAM Memory Space		
0x0000_0000 – 0x0000_FFFF	FLASH_BA	FLASH Memory Space (64KB)
0x2000_0000 – 0x2000_4FFF	SRAM_BA	SRAM Memory Space (20KB)
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
APB1 Controllers Space (0x4000_0000 ~ 0x400F_FFFF)		
0x4000_4000 – 0x4000_7FFF	WDT_BA	Watchdog/Window Watchdog Timer Control Registers
0x4001_0000 – 0x4001_3FFF	TMR01_BA	Timer0/Timer1 Control Registers
0x4002_0000 – 0x4002_3FFF	I2C0_BA	I^{2}C0 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
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^{2}C1 Interface Control Registers
0x4013_0000 – 0x4013_3FFF	SPI2_BA	SPI2 with master/slave function Control Registers

0x4015_0000 – 0x4015_3FFF	UART1_BA	UART1 Control Registers
0x401A_0000 – 0x401A_3FFF	I2S_BA	I^{2}S 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 Nested Vectored Interrupt Controller (NVIC)

5.2.1 Overview

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

5.2.2 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 “ARM® Cortex™-M0 Technical Reference Manual” and “ARM® v6-M Architecture Reference Manual”.

5.2.3 Exception Model and System Interrupt Map

Table 5-2 lists the exception model supported by the NuMicro™ NUC123 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/Window Watchdog Timer interrupt
18	2	**EINT0**	GPIO	External signal interrupt from PB.14 pin
19	3	**EINT1**	GPIO	External signal interrupt from PB.15 or PD.11 pin
20	4	**GPAB_INT**	GPIO	External signal interrupt from PA[15:0]/PB[13:0]
21	5	**GPCDF_INT**	GPIO	External interrupt from PC[15:0]/PD[15:0]/PF[3:0]
22	6	**PWMA_INT**	PWM0~3	PWM0, PWM1, PWM2 and PWM3 interrupt
23	7	**Reserved**	Reserved	Reserved
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	**UART0_INT**	UART0	UART0 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	**Reserved**	Reserved	Reserved
34	18	**I2C0_INT**	I^{2}C0	I^{2}C0 interrupt
35	19	**I2C1_INT**	I^{2}C1	I^{2}C1 interrupt
36	20	**Reserved**	Reserved	Reserved
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	**Reserved**	Reserved	Reserved
42	26	**PDMA_INT**	PDMA	PDMA interrupt
43	27	**I2S_INT**	I^{2}S	I^{2}S 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	**Reserved**	Reserved	Reserved

Table 5-3 System Interrupt Map

5.2.4 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

5.2.5 Operation Description

NVIC interrupts can be enabled and disabled by writing to their corresponding Interrupt Set-Enable or Interrupt Clear-Enable register bit-field. The registers use a write-1-to-enable and write-1-to-clear policy, both registers reading back the current enabled state of the corresponding interrupts. When an interrupt is disabled, interrupt assertion will cause the interrupt to become Pending; however, the interrupt will not 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.

5.2.6 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	Description	
[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). 1 = Associated interrupt Enabled. 0 = 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	Description	
[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). 1 = Associated interrupt Disabled. 0 = 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	Description	
[31:0]	**SETPEND**	1 = Set pending state of the associated interrupt under software control. Each bit represents an interrupt number from IRQ0 ~ IRQ31 (Vector number from 16 ~ 47). 0 = 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	Description	
[31:0]	**CLRPEND**	1 = Removes the pending state of associated interrupt under software control. Each bit represents an interrupt number from IRQ0 ~ IRQ31 (Vector number from 16 ~ 47). 0 = 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	Description	
[31:30]	**PRI_3**	**Priority of IRQ3** "0" denotes the highest priority and "3" denotes the lowest priority.
[29:24]	**Reserved**	Reserved
[23:22]	**PRI_2**	**Priority of IRQ2** "0" denotes the highest priority and "3" denotes the lowest priority.
[21:16]	**Reserved**	Reserved
[15:14]	**PRI_1**	**Priority of IRQ1** "0" denotes the highest priority and "3" denotes the lowest priority.
[13:8]	**Reserved**	Reserved
[7:6]	**PRI_0**	**Priority of IRQ0** "0" denotes the highest priority and "3" denotes the lowest priority.
[5:0]	**Reserved**	Reserved

IRQ4 ~ IRQ7 Interrupt Priority Register (NVIC_IPR1)

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

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

IRQ8 ~ IRQ11 Interrupt Priority Register (NVIC_IPR2)

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

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

IRQ12 ~ IRQ15 Interrupt Priority Register (NVIC_IPR3)

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

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

IRQ16 ~ IRQ19 Interrupt Priority Register (NVIC_IPR4)

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

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

IRQ20 ~ IRQ23 Interrupt Priority Register (NVIC_IPR5)

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

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

IRQ24 ~ IRQ27 Interrupt Priority Register (NVIC_IPR6)

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

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

IRQ28 ~ IRQ31 Interrupt Priority Register (NVIC_IPR7)

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

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

5.2.7 Interrupt Source Control Registers

Besides the interrupt control registers associated with the NVIC, the NuMicro™ NUC123 series also implement some specific control registers to facilitate the interrupt functions, including "interrupt source identification", "NMI source selection" and "interrupt test mode", which are described 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/F) 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 (Reserved) 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 (Reserved) 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 (Reserved) 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 (Reserved) 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 (Reserved) 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 (Reserved) 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-No	Description
[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 or PD.11
[2:0]	INT_BA+0x10	4	Bit2: 0 Bit1: GPB_INT Bit0: GPA_INT
[2:0]	INT_BA+0x14	5	Bit3: GPF_INT Bit2: 0 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	Reserved
[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	Reserved
[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	Reserved
[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	Reserved
[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	Reserved

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	Description	
[31:8]	**Reserved**	Reserved
[8]	**NMI_EN**	**NMI Interrupt Enable** 1 = NMI interrupt Enabled. 0 = NMI interrupt Disabled.
[7:5]	**Reserved**	Reserved
[4:0]	**NMI_SEL**	**NMI Interrupt Source Selection** 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	Description	
[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, Normal mode and Test mode. The MCU_IRQ collects all interrupts from each peripheral and synchronizes them and 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), setting 1 to the MCU_bit[n] will clear the interrupt and setting MCU_IRQ[n] 0 has no effect.

5.3 System Manager

5.3.1 Overview

System management includes the 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

5.3.2 System Reset

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

- Power-On Reset
- 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 does not reset external crystal circuit and ISPCON.BS bit, but Power-On Reset does.

5.3.3 System Power Distribution

In this chip, 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 1.8 V power for digital operation and I/O pins.
- USB transceiver power from VBUS offers the power for operating the USB transceiver.

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 5-1 shows the power distribution of the NuMicro™ NUC123.

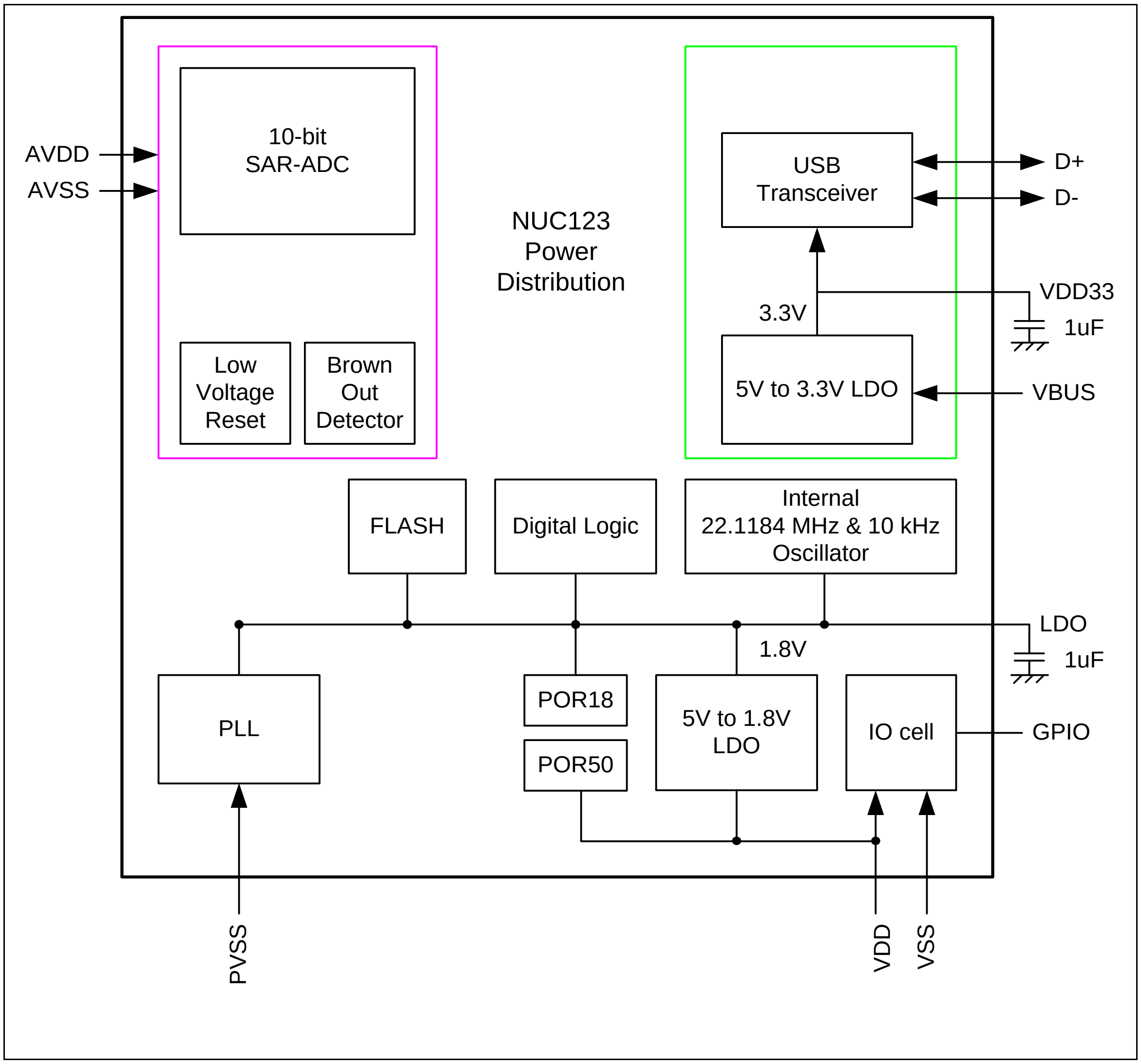


Figure 5-1 NuMicro™ NUC123 Power Distribution Diagram

5.3.4 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	0x0001_23XX[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
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
GPF_MFP	GCR_BA+0x44	R/W	GPIOF Multiple Function and Input Type Control Register	0x0000_000X
ALT_MFP	GCR_BA+0x50	R/W	Alternative Multiple Function Pin Control Register	0x0000_0000
ALT_MFP1	GCR_BA+0x54	R/W	Alternative Multiple Function Pin Control Register 1	0x0000_0000
GPA_IOCR	GCR_BA+0xC0	R/W	GPIOA IO Control Register	0x0000_0000
GPB_IOCR	GCR_BA+0xC4	R/W	GPIOB IO Control Register	0x0000_0000
GPD_IOCR	GCR_BA+0xCC	R/W	GPIOB IO Control Register	0x0000_0000
REGWRPROT	GCR_BA+0x100	R/W	Register Write Protect register	0x0000_0000

Note: [1] Depending on the 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	0x0001_23XX[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	Description	
[31:0]	**PDID**	**Part Device Identification Number** This register reflects the device part number code. S/W can read this register to identify which device is used.

PART NUMBER	PACKAGE	FLASH	SRAM	PDID
NUC123ZD4AN0	QFN-33	68	20	0x0001_2355
NUC123ZC2AN1	QFN-33	36	12	0x0001_2345
NUC123LD4AN0	LQFP-48	68	20	0x0001_2335
NUC123LC2AN1	LQFP-48	36	12	0x0001_2325
NUC123SD4AN0	LQFP-64	68	20	0x0001_2315
NUC123SC2AN1	LQFP-64	36	12	0x0001_2305

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	Description	
[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 = 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 = 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 = 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 = 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 = 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 = 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 = 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					**PDMA_RST**	**CPU_RST**	**CHIP_RST**

Bits	Description	
[31:3]	**Reserved**	Reserved
[2]	**PDMA_RST**	**PDMA Controller Reset (Write-protection**) 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. Refer to 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. Refer to 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. For the difference between CHIP_RST and SYSRESETREQ, please refer to section *5.2.2.* This bit is the protected bit, which means programming it needs to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Refer to 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. User needs 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**		
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PS2_RST	**Reserved**		**PWM03_RST**	**Reserved**		**UART1_RST**	**UART0_RST**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved	**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	Description	
[31:30]	**Reserved**	Reserved
[29]	**I2S_RST**	**I^{2}S Controller Reset** 1 = I^{2}S controller reset 0 = I^{2}S 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
[26:24]	**Reserved**	Reserved
[23]	**PS2_RST**	**PS/2 Controller Reset** 1 = PS/2 controller reset 0 = PS/2 controller normal operation
[22:21]	**Reserved**	Reserved
[20]	**PWM03_RST**	**PWM03 controller Reset** 1 = PWM03 controller reset 0 = PWM03 controller normal operation
[19:18]	**Reserved**	Reserved

[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]	**Reserved**	Reserved
[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. Refer to 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	Description	
[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 by default. 1 = Low Voltage Reset function Enabled– After enabling the bit, the LVR function will be active with 100uS delay for LVR output stable (Default). 0 = Low Voltage Reset function Disabled. This bit is the protected bit, which means programming it needs to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Refer to the register REGWRPROT at address GCR_BA+0x100.
[6]	**BOD_OUT**	**Brown-out Detector Output Status** 1 = Brown-out Detector output status is 1, which means the detected voltage is lower than BOD_VL setting. If the BOD_EN is 0, BOD function disabled , this bit always responds to 0 0 = Brown-out Detector output status is 0, which means the detected voltage is higher than BOD_VL setting or BOD_EN is 0.

<table>
<tr><td>[5]</td><td><b>BOD_LPM</b></td><td><b>Brown-out Detector Low power Mode (Write-protection Bit)</b>
1 = BOD low power mode Enabled.
0 = BOD operated 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, which means programming this needs to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Refer to the register REGWRPROT at address GCR_BA+0x100.</td></tr>
<tr><td>[4]</td><td><b>BOD_INTF</b></td><td><b>Brown-out Detector Interrupt Flag</b>
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.</td></tr>
<tr><td>[3]</td><td><b>BOD_RSTEN</b></td><td><b>Brown-out Reset Enable (Write-protection Bit)</b>
1 = Brown-out “RESET” function Enabled.
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 = Brown-out “INTERRUPT” function Enabled.
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. Refer to the register REGWRPROT at address GCR_BA+0x100.</td></tr>
<tr><td>[2:1]</td><td><b>BOD_VL</b></td><td><b>Brown-out Detector Threshold Voltage Selection (Write-protection Bits)</b>
The default value is set by flash controller user configuration register config0 bit[22:21]
This bit is the protected bit, which means programming it needs to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Refer to 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></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 Enabled. 0 = Brown-out Detector function Disabled. This bit is the protected bit, which means programming it needs to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Refer to the register REGWRPROT at address GCR_BA+0x100.

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	Description	
[31:16]	**Reserved**	Reserved
[15:0]	**POR_DIS_CODE**	**The register is used for the Power-On-Reset enable control (Write-protection Bits)** When powered on, the POR circuit generates a reset signal to reset the whole chip function, but noise on the power may cause the POR active again. User can disable internal POR circuit to avoid unpredictable noise to cause chip reset by writing 0x5AA5 to this field. The POR function will become 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, which means programming it needs to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Refer to 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

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

<table>
<tr><th>Bits</th><th colspan="2">Description</th></tr>
<tr><td>[31:16]</td><td><b>GPA_TYPEn</b></td><td>1 = GPIOA[15:0] I/O input Schmitt Trigger function Enabled.
0 = GPIOA[15:0] I/O input Schmitt Trigger function Disabled.
GPA[9:0] are reserved in this chip.</td></tr>
<tr><td>[15]</td><td><b>GPA_MFP15</b></td><td><b>PA.15 Pin Function Selection</b>
The pin function depends on GPA_MFP[15] and PA15_MFP1 (ALT_MFP[9]).
<table>
<tr><th>PA15_MFP1
(ALT_MFP[9])</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>0</td><td>CLKO (Clock Driver output)</td></tr>
<tr><td>1</td><td>1</td><td>I2SMCLK (I^{2}S)</td></tr>
</table></td></tr>
<tr><td>[14]</td><td><b>GPA_MFP14</b></td><td><b>PA.14 Pin Function Selection</b>
The pin function depends on GPA_MFP[14].
<table>
<tr><th>GPA_MFP[14]</th><th>PA.14 Function</th></tr>
<tr><td>0</td><td>GPIO</td></tr>
<tr><td>1</td><td>PWM2 (PWM)</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>[13]</td><td><b>GPA_MFP13</b></td><td><b>PA.13 Pin Function Selection</b>
The pin function depends on GPA_MFP[13].
<table>
<tr><th>GPA_MFP[13]</th><th>PA.13 Function</th></tr>
<tr><td>0</td><td>GPIO</td></tr>
<tr><td>1</td><td>PWM1 (PWM)</td></tr>
</table></td></tr>
<tr><td>[12]</td><td><b>GPA_MFP12</b></td><td><b>PA.12 Pin Function Selection</b>
The pin function depends on GPA_MFP[12].
<table>
<tr><th>GPA_MFP[12]</th><th>PA.12 Function</th></tr>
<tr><td>0</td><td>GPIO</td></tr>
<tr><td>1</td><td>PWM0 (PWM)</td></tr>
</table></td></tr>
<tr><td>[11]</td><td><b>GPA_MFP11</b></td><td><b>PA.11 Pin Function Selection</b>
The pin function depends on GPA_MFP[11] and PA11_MFP1 (ALT_MFP[11]).
<table>
<tr><th>PA11_MFP1
(ALT_MFP[11])</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>I2C1SCL (I^{2}C1)</td></tr>
<tr><td>1</td><td>0</td><td>SPICLK1 (SPI1)</td></tr>
<tr><td>1</td><td>1</td><td>MOSI20 (SPI2)</td></tr>
</table></td></tr>
<tr><td>[10]</td><td><b>GPA_MFP10</b></td><td><b>PA.10 Pin Function Selection</b>
The pin function depends on GPA_MFP[10] and PA10_MFP1 (ALT_MFP[12]).
<table>
<tr><th>PA10_MFP1
(ALT_MFP[12])</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>I2C1SDA (I^{2}C1)</td></tr>
<tr><td>1</td><td>0</td><td>MISO10 (SPI1)</td></tr>
<tr><td>1</td><td>1</td><td>MISO20 (SPI2)</td></tr>
</table></td></tr>
<tr><td>[9:0]</td><td><b>Reserved</b></td><td>Reserved</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">Description</th></tr>
<tr><td>[31:16]</td><td><b>GPB_TYPEn</b></td><td>1 = GPIOB[15:0] I/O input Schmitt Trigger function Enabled.
0 = GPIOB[15:0] I/O input Schmitt Trigger function Disabled.</td></tr>
<tr><td>[15]</td><td><b>GPB_MFP15</b></td><td><b>PB.15 Pin Function Selection</b>
Bits PB15_MFP1 (ALT_MFP[24]) and GPB_MFP[15] determine the PB.15 function.
<table>
<tr><th>PB15_MFP1
(ALT_MFP[24])</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>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>T0EX (TMR0)</td></tr>
</table></td></tr>
<tr><td>[14]</td><td><b>GPB_MFP14</b></td><td><b>PB.14 Pin Function Selection</b>
The pin function depends on GPB_MFP[14].
<table>
<tr><th>GPB_MFP[14]</th><th>PB.14 Function</th></tr>
<tr><td>0</td><td>GPIO</td></tr>
<tr><td>1</td><td>/INT0</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>[13]</td><td><b>GPB_MFP13</b></td><td><b>PB.13 Pin Function Selection</b>
The pin is a dedicated GPIO pin.
<table>
<tr><th>GPB_MFP[13]</th><th>PB.13 Function</th></tr>
<tr><td>0</td><td>GPIO</td></tr>
</table></td></tr>
<tr><td>[12]</td><td><b>GPB_MFP12</b></td><td><b>PB.12 Pin Function Selection</b>
The pin function depends on GPB_MFP[12] and PB12_MFP1 (ALT_MFP[10]).
<table>
<tr><th>PB12_MFP1
(ALT_MFP[10])</th><th>GPB_MFP[12]</th><th>PB.12 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>SPISS10 (SPI1)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>CLKO (Clock Driver output)</td></tr>
</table></td></tr>
<tr><td>[11]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[10]</td><td><b>GPB_MFP10</b></td><td><b>PB.10 Pin Function Selection</b>
The pin function depends on GPB_MFP[10] and PB10_MFP1 (ALT_MFP[0]).
<table>
<tr><th>PB10_MFP1
(ALT_MFP[0])</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>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>SPISS01 (SPI0)</td></tr>
</table></td></tr>
<tr><td>[9]</td><td><b>GPB_MFP9</b></td><td><b>PB.9 Pin Function Selection</b>
The pin function depends on GPB_MFP[9] and PB9_MFP1 (ALT_MFP[1]).
<table>
<tr><th>PB9_MFP1
(ALT_MFP[1])</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>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>SPISS11 (SPI1)</td></tr>
</table></td></tr>
<tr><td>[8]</td><td><b>GPB_MFP8</b></td><td><b>PB.8 Pin Function Selection</b>
GPB_MFP[8] determines the PB.8 function.
<table>
<tr><th>GPB_MFP[8]</th><th>PB.8 Function</th></tr>
<tr><td>0</td><td>GPIO</td></tr>
<tr><td>1</td><td>TM0</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>[7]</td><td><b>GPB_MFP7</b></td><td><b>PB.7 Pin Function Selection</b>
The pin function depends on GPB_MFP[7] and PB7_MFP1 (ALT_MFP[16]).
<table>
<tr><th>PB7_MFP1
(ALT_MFP[16])</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>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>MISO20 (SPI2)</td></tr>
</table></td></tr>
<tr><td>[6]</td><td><b>GPB_MFP6</b></td><td><b>PB.6 Pin Function Selection</b>
The pin function depends on GPB_MFP[6] and PB6_MFP1 (ALT_MFP[17]).
<table>
<tr><th>PB6_MFP1
(ALT_MFP[17])</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>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>MOSI20 (SPI2)</td></tr>
</table></td></tr>
<tr><td>[5]</td><td><b>GPB_MFP5</b></td><td><b>PB. 5 Pin Function Selection</b>
The pin function depends on GPB_MFP[5] and PB5_MFP1 (ALT_MFP[18]).
<table>
<tr><th>PB5_MFP1
(ALT_MFP[18])</th><th>GPB_MFP[5]</th><th>PB.5 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>TXD1 (UART1)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>SPICLK2 (SPI2)</td></tr>
</table></td></tr>
<tr><td>[4]</td><td><b>GPB_MFP4</b></td><td><b>PB.4 Pin Function Selection</b>
The pin function depends on GPB_MFP[4] and PB4_MFP1 (ALT_MFP[15]).
<table>
<tr><th>PB4_MFP1
(ALT_MFP[15])</th><th>GPB_MFP[4]</th><th>PB.4 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>RXD1 (UART1)</td></tr>
<tr><td>1</td><td>0</td><td>SPISS20 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>SPISS11 (SPI1)</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>[3]</td><td><b>GPB_MFP3</b></td><td><b>PB.3 Pin Function Selection</b>
The pin function depends on GPB_MFP[3] and PB3_MFP1 (ALT_MFP[27]).
<table>
<tr><th>PB3_MFP1
(ALT_MFP[27])</th><th>GPB_MFP[3]</th><th>PB.3 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>CTS0 (UART0)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>T3EX (TMR3)</td></tr>
</table></td></tr>
<tr><td>[2]</td><td><b>GPB_MFP2</b></td><td><b>PB.2 Pin Function Selection</b>
The pin function depends on GPB_MFP[2] and PB2_MFP1 (ALT_MFP[26]).
<table>
<tr><th>PB2_MFP1
(ALT_MFP[26])</th><th>GPB_MFP[2]</th><th>PB.2 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>RTS0 (UART0)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>T2EX (TMR2)</td></tr>
</table></td></tr>
<tr><td>[1]</td><td><b>GPB_MFP1</b></td><td><b>PB.1 Pin Function Selection</b>
1 = UART0 TXD0 function is selected to the pin PB.1.
0 = GPIOB[1] is selected to the pin PB.1.</td></tr>
<tr><td>[0]</td><td><b>GPB_MFP0</b></td><td><b>PB.0 Pin Function Selection</b>
1 = UART0 RXD0 function is selected to the pin PB.0.
0 = 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]							

<table>
<tr><th>Bits</th><th colspan="2">Description</th></tr>
<tr><td>[31:16]</td><td><b>GPC_TYPEn</b></td><td>1 = GPIOC[15:0] I/O input Schmitt Trigger function Enabled.
0 = GPIOC[15:0] I/O input Schmitt Trigger function Disabled.</td></tr>
<tr><td>[15:14]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[13]</td><td><b>GPC_MFP13</b></td><td><b>PC.13 Pin Function Selection</b>
Bits PC13_MFP1 (ALT_MFP[21]) and GPC_MFP[13] determine the PC.13 function.
<table>
<tr><th>PC13_MFP1
(ALT_MFP[21])</th><th>GPC_MFP[13]</th><th>PC.13 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MOSI11 (SPI1)</td></tr>
<tr><td>1</td><td>0</td><td>CLKO (Clock Driver output)</td></tr>
<tr><td>1</td><td>1</td><td>PWM3 (PWM)</td></tr>
</table></td></tr>
<tr><td>[12]</td><td><b>GPC_MFP12</b></td><td><b>PC.12 Pin Function Selection</b>
Bits PC12_MFP1 (ALT_MFP[20]) and GPC_MFP[12] determine the PC.12 function.
<table>
<tr><th>PC12_MFP1
(ALT_MFP[20])</th><th>GPC_MFP[12]</th><th>PC.12 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MISO11 (SPI1)</td></tr>
<tr><td>1</td><td>0</td><td>I2SMCLK (I^{2}S)</td></tr>
<tr><td>1</td><td>1</td><td>PWM2 (PWM)</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>[11]</td><td><b>GPC_MFP11</b></td><td><b>PC.11 Pin Function Selection</b>
1 = MOSI10 (SPI1 master output, slave input pin-0) function is selected to the pin PC.11.
0 = GPIOC[11] is selected to the pin PC.11.</td></tr>
<tr><td>[10]</td><td><b>GPC_MFP10</b></td><td><b>PC.10 Pin Function Selection</b>
1 = MISO10 (SPI1 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><b>GPC_MFP9</b></td><td><b>PC.9 Pin Function Selection</b>
1 = SPICLK1 (SPI1) function is selected to the pin PC.9.
0 = GPIOC[9] is selected to the pin PC.9.</td></tr>
<tr><td>[8]</td><td><b>GPC_MFP8</b></td><td><b>PC.8 Pin Function Selection</b>
The pin function depends on GPC_MFP[8].
<table>
<tr><th>GPC_MFP[8]</th><th>PC.8 Function</th></tr>
<tr><td>0</td><td>GPIO</td></tr>
<tr><td>1</td><td>SPISS10 (SPI1)</td></tr>
</table></td></tr>
<tr><td>[7:6]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[5]</td><td><b>GPC_MFP5</b></td><td><b>PC.5 Pin Function Selection</b>
Bits PC5_MFP1 (ALT_MFP[30]) and GPC_MFP[5] determine the PC.5 function.
<table>
<tr><th>PC5_MFP1
(ALT_MFP[30])</th><th>GPC_MFP[5]</th><th>PC.5 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MOSI01 (SPI0)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>TXD0 (UART0)</td></tr>
</table></td></tr>
<tr><td>[4]</td><td><b>GPC_MFP4</b></td><td><b>PC.4 Pin Function Selection</b>
Bits PC4_MFP1 (ALT_MFP[29]) and GPC_MFP[4] determine the PC.4 function.
<table>
<tr><th>PC4_MFP1
(ALT_MFP[29])</th><th>GPC_MFP[4]</th><th>PC.4 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MISO01 (SPI0)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>RXD0 (UART0)</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>[3]</td><td><b>GPC_MFP3</b></td><td><b>PC.3 Pin Function Selection</b>
Bits PC3_MFP1 (ALT_MFP[8]) and GPC_MFP[3] determine the PC.3 function.
<table>
<tr><th>PC3_MFP1
(ALT_MFP[8])</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>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>I2SDO (I^{2}S)</td></tr>
</table></td></tr>
<tr><td>[2]</td><td><b>GPC_MFP2</b></td><td><b>PC.2 Pin Function Selection</b>
Bits PC2_MFP1 (ALT_MFP[7]) and GPC_MFP[2] determine the PC.2 function.
<table>
<tr><th>PC2_MFP1
(ALT_MFP[7])</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>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>I2SDI (I^{2}S)</td></tr>
</table></td></tr>
<tr><td>[1]</td><td><b>GPC_MFP1</b></td><td><b>PC.1 Pin Function Selection</b>
Bits PC1_MFP1 (ALT_MFP[6]) and GPC_MFP[1] determine the PC.1 function.
<table>
<tr><th>PC1_MFP1
(ALT_MFP[6])</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>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>I2SBCLK (I^{2}S)</td></tr>
</table></td></tr>
<tr><td>[0]</td><td><b>GPC_MFP0</b></td><td><b>PC.0 Pin Function Selection</b>
Bits PC0_MFP1 (ALT_MFP[5]) and GPC_MFP[0] determine the PC.0 function.
<table>
<tr><th>PC0_MFP1
(ALT_MFP[5])</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>0</td><td>Reserved</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]							

<table>
<tr><th>Bits</th><th colspan="2">Description</th></tr>
<tr><td>[31:16]</td><td><b>GPD_TYPEn</b></td><td>1 = GPIOD[15:0] I/O input Schmitt Trigger function Enabled.
0 = GPIOD[15:0] I/O input Schmitt Trigger function Disabled.</td></tr>
<tr><td>[15:12]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[11]</td><td><b>GPD_MFP11</b></td><td><b>PD.11 Pin Function Selection</b>
1 = /INT1 function is selected to the pin PD.11.
0 = GPIOD[11] is selected to the pin PD.11.</td></tr>
<tr><td>[10]</td><td><b>GPD_MFP10</b></td><td><b>PD.10 Pin Function Selection</b>
1 = CLKO function is selected to the pin PD.10.
0 = GPIOD[10] is selected to the pin PD.10.</td></tr>
<tr><td>[9]</td><td><b>GPD_MFP9</b></td><td><b>PD.9 Pin Function Selection</b>
The pin is a dedicated GPIO pin.<table><tr><th>GPD_MFP[9]</th><th>PD.9 Function</th></tr><tr><td>0</td><td>GPIO</td></tr></table></td></tr>
<tr><td>[8]</td><td><b>GPD_MFP8</b></td><td><b>PD.8 Pin Function Selection</b>
1 = MOSI10 (SPI1 master output, slave input pin-0) function is selected to the pin PD8.
0 = GPIOD[8] is selected to the pin PD8.</td></tr>
<tr><td>[7:6]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
</table>

<table>
<tr><td>[5]</td><td><b>GPD_MFP5</b></td><td><b>PD.5 Pin Function Selection</b>
Bits PD5_MFP1 (ALT_MFP1[21]) and GPD_MFP[5] determine the PD.5 function.
<table>
<tr><th>PD5_MFP1
(ALT_MFP1[21])</th><th>GPD_MFP[5]</th><th>PD.5 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>1</td><td>0</td><td>MOSI21 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC5</td></tr>
</table></td></tr>
<tr><td>[4]</td><td><b>GPD_MFP4</b></td><td><b>PD.4 Pin Function Selection</b>
Bits PD4_MFP1 (ALT_MFP1[20]) and GPD_MFP[4] determine the PD.4 function.
<table>
<tr><th>PD4_MFP1
(ALT_MFP1[20])</th><th>GPD_MFP[4]</th><th>PD.4 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>1</td><td>0</td><td>MISO21 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC4</td></tr>
</table></td></tr>
<tr><td>[3]</td><td><b>GPD_MFP3</b></td><td><b>PD.3 Pin Function Selection</b>
Bits PD3_MFP1 (ALT_MFP1[19]) and GPD_MFP[3] determine the PD.3 function.
<table>
<tr><th>PD3_MPF1
(ALT_MFP1[19])</th><th>GPD_MFP[3]</th><th>PD.3 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MOSI01 (SPI0)</td></tr>
<tr><td>1</td><td>0</td><td>MOSI20 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC3</td></tr>
</table></td></tr>
<tr><td>[2]</td><td><b>GPD_MFP2</b></td><td><b>PD.2 Pin Function Selection</b>
Bits PD2_MFP1 (ALT_MFP1[18]) and GPD_MFP[2] determine the PD.2 function.
<table>
<tr><th>PD2_MFP1
(ALT_MFP1[18])</th><th>GPD_MFP[2]</th><th>PD.2 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MISO01 (SPI0)</td></tr>
<tr><td>1</td><td>0</td><td>MISO20 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC2</td></tr>
</table></td></tr>
</table>

<table>
<tr>
<td>[1]</td>
<td><b>GPD_MFP1</b></td>
<td><b>PD.1 Pin Function Selection</b>
Bits PD1_MFP1 (ALT_MFP1[17]) and GPD_MFP[1] determine the PD.1 function.
<table>
<tr><th>PD1_MFP1
(ALT_MFP1[17])</th><th>GPD_MFP[1]</th><th>PD.1 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>SPISS01 (SPI0)</td></tr>
<tr><td>1</td><td>0</td><td>SPICLK2 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC1</td></tr>
</table>
</td>
</tr>
<tr>
<td>[0]</td>
<td><b>GPD_MFP0</b></td>
<td><b>PD.0 Pin Function Selection</b>
Bits PD0_MFP1 (ALT_MFP1[16]) and GPD_MFP[0] determine the PD.0 function.
<table>
<tr><th>PD0_MFP1
(ALT_MFP1[16])</th><th>GPD_MFP[0]</th><th>PD.0 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>1</td><td>0</td><td>SPISS20 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC0</td></tr>
</table>
</td>
</tr>
</table>

Multiple Function Pin GPIOF Control Register (GPF_MFP)

Register	Offset	R/W	Description	Reset Value
GPF_MFP	GCR_BA+0x44	R/W	GPIOF Multiple Function and Input Type Control Register	0x0000_000X

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved				**GPF_TYPE[3:0]**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**GPF_MFP3**	**GPF_MFP2**	**GPF_MFP1**	**GPF_MFP0**

<table>
<tr><th>Bits</th><th colspan="2">Description</th></tr>
<tr><td>[31:20]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[19:16]</td><td><b>GPF_TYPEn</b></td><td>1 = GPIOF[3:0] I/O input Schmitt Trigger function Enabled.
0 = GPIOF[3:0] I/O input Schmitt Trigger function Disabled.</td></tr>
<tr><td>[15:4]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[3]</td><td><b>GPF_MFP3</b></td><td><b>PF.3 Pin Function Selection</b>
PF3_MFP1 (ALT_MFP1[27:26]) and GPF_MFP[3] determine the PF.3 function.
The reset value of this bit is 1.
<table>
<tr><th>PF3_MFP1
(ALT_MFP1[27:26])</th><th>GPF_MFP[3]</th><th>PF.3 Function</th></tr>
<tr><td>00</td><td>0</td><td>GPIO</td></tr>
<tr><td>00</td><td>1</td><td>PS2CLK</td></tr>
<tr><td>01</td><td>1</td><td>Reserved</td></tr>
<tr><td>10</td><td>1</td><td>I2C0SCL (I^{2}C0)</td></tr>
<tr><td>11</td><td>1</td><td>ADC7</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>[2]</td><td><b>GPF_MFP2</b></td><td><b>PF.2 Pin Function Selection</b>
PF2_MFP1 (ALT_MFP1[25:24]) and GPF_MFP[2] determine the PF.2 function.
The reset value of this bit is 1.
<table>
<tr><th>PF2_MFP1
(ALT_MFP1[25:24])</th><th>GPF_MFP[2]</th><th>PF.2 Function</th></tr>
<tr><td>00</td><td>0</td><td>GPIO</td></tr>
<tr><td>00</td><td>1</td><td>PS2DAT</td></tr>
<tr><td>01</td><td>1</td><td>Reserved</td></tr>
<tr><td>10</td><td>1</td><td>I2C0SDA (I^{2}C0)</td></tr>
<tr><td>11</td><td>1</td><td>ADC6</td></tr>
</table></td></tr>
<tr><td>[1]</td><td><b>GPF_MFP1</b></td><td><b>PF.1 Pin Function Selection</b>
The reset value of this bit controlled by CFOSC of Config0. If CFOSC=000, the reset value of this bit is 1. If CFOSC=111, the reset value of this bit is 0.
1 = XT1_IN function is selected to the pin PF.1.
0 = GPIOF[1] is selected to the pin PF.1.</td></tr>
<tr><td>[0]</td><td><b>GPF_MFP0</b></td><td><b>PF.0 Pin Function Selection</b>
The reset value of this bit controlled by CFOSC of Config0. If CFOSC=000, the reset value of this bit is 1. If CFOSC=111, the reset value of this bit is 0.
1 = XT1_OUT function is selected to the pin PF.0.
0 = GPIOF[0] is selected to the pin PF.0.</td></tr>
</table>

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	PC5_MFP1	PC4_MFP1	Reserved	PB3_MFP1	PB2_MFP1	Reserved	PB15_MFP1
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved		PC13_MFP1	PC12_MFP1	Reserved	PB5_MFP1	PB6_MFP1	PB7_MFP1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PB4_MFP1	Reserved		PA10_MFP1	PA11_MFP1	PB12_MFP1	PA15_MFP1	PC3_MFP1
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PC2_MFP1	PC1_MFP1	PC0_MFP1	Reserved			PB9_MFP1	PB10_MFP1

<table>
<tr><th>Bits</th><th colspan="2">Description</th></tr>
<tr><td>[31]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[30]</td><td><b>PC5_MFP1</b></td><td><b>PC.5 Pin Function Selection</b>
Bits PC5_MFP1 (ALT_MFP[30]) and GPC_MFP[5] determine the PC.5 function.
<table>
<tr><th>PC5_MFP1
(ALT_MFP[30])</th><th>GPC_MFP[5]</th><th>PC.5 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MOSI01 (SPI0)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>TXD0 (UART0)</td></tr>
</table></td></tr>
<tr><td>[29]</td><td><b>PC4_MFP1</b></td><td><b>PC.4 Pin Function Selection</b>
Bits PC4_MFP1 (ALT_MFP[29]) and GPC_MFP[4] determine the PC.4 function.
<table>
<tr><th>PC4_MFP1
(ALT_MFP[29])</th><th>GPC_MFP[4]</th><th>PC.4 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MISO01 (SPI0)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>RXD0 (UART0)</td></tr>
</table></td></tr>
<tr><td>[28]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
</table>

<table>
<tr><td>[27]</td><td><b>PB3_MFP1</b></td><td><b>PB.3 Pin Alternate Function Selection</b>
The pin function depends on GPB_MFP[3] and PB3_MFP1 (ALT_MFP[27]).
<table>
<tr><th>PB3_MFP1
(ALT_MFP[27])</th><th>GPB_MFP[3]</th><th>PB.3 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>CTS0 (UART0)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>T3EX (TMR3)</td></tr>
</table></td></tr>
<tr><td>[26]</td><td><b>PB2_MFP1</b></td><td><b>PB.2 Pin Alternate Function Selection</b>
The pin function depends on GPB_MFP[2] and PB2_MFP1 (ALT_MFP[26]).
<table>
<tr><th>PB2_MFP1
(ALT_MFP[26])</th><th>GPB_MFP[2]</th><th>PB.2 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>RTS0 (UART0)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>T2EX (TMR2)</td></tr>
</table></td></tr>
<tr><td>[25]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[24]</td><td><b>PB15_MFP1</b></td><td><b>PB.15 Pin Alternate Function Selection</b>
Bits PB15_MFP1 (ALT_MFP[24]) and GPB_MFP[15] determine the PB.15 function.
<table>
<tr><th>PB15_MFP1
(ALT_MFP[24])</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>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>T0EX (TMR0)</td></tr>
</table></td></tr>
<tr><td>[23:22]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
</table>

<table>
<tr>
<td>[21]</td>
<td><b>PC13_MFP1</b></td>
<td><b>PC.13 Pin Function Selection</b>
Bits PC13_MFP1 (ALT_MFP[21]) and GPC_MFP[13] determine the PC.13 function.
<table>
<tr><th>PC13_MFP1
(ALT_MFP[21])</th><th>GPC_MFP[13]</th><th>PC.13 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MOSI11 (SPI1)</td></tr>
<tr><td>1</td><td>0</td><td>CLKO (Clock Driver output)</td></tr>
<tr><td>1</td><td>1</td><td>PWM3 (PWM)</td></tr>
</table>
</td>
</tr>
<tr>
<td>[20]</td>
<td><b>PC12_MFP1</b></td>
<td><b>PC.12 Pin Function Selection</b>
Bits PC12_MFP1 (ALT_MFP[20]) and GPC_MFP[12] determine the PC.12 function.
<table>
<tr><th>PC12_MFP1
(ALT_MFP[20])</th><th>GPC_MFP[12]</th><th>PC.12 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MISO11 (SPI1)</td></tr>
<tr><td>1</td><td>0</td><td>I2SMCLK (I^{2}S)</td></tr>
<tr><td>1</td><td>1</td><td>PWM2 (PWM)</td></tr>
</table>
</td>
</tr>
<tr>
<td>[19]</td>
<td><b>Reserved</b></td>
<td>Reserved</td>
</tr>
<tr>
<td>[18]</td>
<td><b>PB5_MFP1</b></td>
<td><b>PB. 5 Pin Alternate Function Selection</b>
The pin function depends on GPB_MFP[5] and PB5_MFP1 (ALT_MFP[18]).
<table>
<tr><th>PB5_MFP1
(ALT_MFP[18])</th><th>GPB_MFP[5]</th><th>PB.5 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>TXD1 (UART1)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>SPICLK2 (SPI2)</td></tr>
</table>
</td>
</tr>
<tr>
<td>[17]</td>
<td><b>PB6_MFP1</b></td>
<td><b>PB.6 Pin Alternate Function Selection</b>
The pin function depends on GPB_MFP[6] and PB6_MFP1 (ALT_MFP[17]).
<table>
<tr><th>PB6_MFP1
(ALT_MFP[17])</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>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>MOSI20 (SPI2)</td></tr>
</table>
</td>
</tr>
</table>

<table>
<tr><td>[16]</td><td><b>PB7_MFP1</b></td><td><b>PB.7 Pin Alternate Function Selection</b>
The pin function depends on GPB_MFP[7] and PB7_MFP1 (ALT_MFP[16]).
<table>
<tr><th>PB7_MFP1
(ALT_MFP[16])</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>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>MISO20 (SPI2)</td></tr>
</table></td></tr>
<tr><td>[15]</td><td><b>PB4_MFP1</b></td><td><b>PB.4 Pin Function Selection</b>
The pin function depends on GPB_MFP[4] and PB4_MFP1 (ALT_MFP[15]).
<table>
<tr><th>PB4_MFP1
(ALT_MFP[15])</th><th>GPB_MFP[4]</th><th>PB.4 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>RXD1 (UART1)</td></tr>
<tr><td>1</td><td>0</td><td>SPISS20 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>SPISS11 (SPI1)</td></tr>
</table></td></tr>
<tr><td>[14:13]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[12]</td><td><b>PA10_MFP1</b></td><td><b>PA.10 Pin Function Selection</b>
The pin function depends on GPA_MFP[10] and PA10_MFP1 (ALT_MFP[12]).
<table>
<tr><th>PA10_MFP1
(ALT_MFP[12])</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>I2C1SDA (I^{2}C1)</td></tr>
<tr><td>1</td><td>0</td><td>MISO10 (SPI1)</td></tr>
<tr><td>1</td><td>1</td><td>MISO20 (SPI2)</td></tr>
</table></td></tr>
<tr><td>[11]</td><td><b>PA11_MFP1</b></td><td><b>PA.11 Pin Function Selection</b>
The pin function depends on GPA_MFP[11] and PA11_MFP1 (ALT_MFP[11]).
<table>
<tr><th>PA11_MFP1
(ALT_MFP[11])</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>I2C1SCL (I^{2}C1)</td></tr>
<tr><td>1</td><td>0</td><td>SPICLK1 (SPI1)</td></tr>
<tr><td>1</td><td>1</td><td>MOSI20 (SPI2)</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>[10]</td><td><b>PB12_MFP1</b></td><td><b>PB.12 Pin Alternate Function Selection</b>
The pin function depends on GPB_MFP[12] and PB12_MFP1 (ALT_MFP[10]).
<table>
<tr><th>PB12_MFP1
(ALT_MFP[10])</th><th>GPB_MFP[12]</th><th>PB.12 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>SPISS10 (SPI1)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>CLKO (Clock Driver output)</td></tr>
</table></td></tr>
<tr><td>[9]</td><td><b>PA15_MFP1</b></td><td><b>PA.15 Pin Alternate Function Selection</b>
The pin function depends on GPA_MFP[15] and PA15_MFP1 (ALT_MFP[9]).
<table>
<tr><th>PA15_MFP1
(ALT_MFP[9])</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>0</td><td>CLKO (Clock Driver output)</td></tr>
<tr><td>1</td><td>1</td><td>I2SMCLK (I^{2}S)</td></tr>
</table></td></tr>
<tr><td>[8]</td><td><b>PC3_MFP1</b></td><td><b>PC.3 Pin Alternate Function Selection</b>
Bits PC3_MFP1 (ALT_MFP[8]) and GPC_MFP[3] determine the PC.3 function.
<table>
<tr><th>PC3_MFP1
(ALT_MFP[8])</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>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>I2SDO (I^{2}S)</td></tr>
</table></td></tr>
<tr><td>[7]</td><td><b>PC2_MFP1</b></td><td><b>PC.2 Pin Alternate Function Selection</b>
Bits PC2_MFP1 (ALT_MFP[7]) and GPC_MFP[2] determine the PC.2 function.
<table>
<tr><th>PC2_MFP1
(ALT_MFP[7])</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>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>I2SDI (I^{2}S)</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>[6]</td><td><b>PC1_MFP1</b></td><td><b>PC.1 Pin Alternate Function Selection</b>
Bits PC1_MFP1 (ALT_MFP[6]) and GPC_MFP[1] determine the PC.1 function.
<table>
<tr><th>PC1_MFP1
(ALT_MFP[6])</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>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>I2SBCLK (I^{2}S)</td></tr>
</table></td></tr>
<tr><td>[5]</td><td><b>PC0_MFP1</b></td><td><b>PC.0 Pin Function Selection</b>
Bits PC0_MFP1 (ALT_MFP[5]) and GPC_MFP[0] determine the PC.0 function.
<table>
<tr><th>PC0_MFP1
(ALT_MFP[5])</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>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>I2SLRCLK (I^{2}S)</td></tr>
</table></td></tr>
<tr><td>[4:2]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[1]</td><td><b>PB9_MFP1</b></td><td><b>PB.9 Pin Alternate Function Selection</b>
The pin function depends on GPB_MFP[9] and PB9_MFP1 (ALT_MFP[1]).
<table>
<tr><th>PB9_MFP1
(ALT_MFP[1])</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>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>SPISS11 (SPI1)</td></tr>
</table></td></tr>
<tr><td>[0]</td><td><b>PB10_MFP1</b></td><td><b>PB.10 Pin Alternate Function Selection</b>
The pin function depends on GPB_MFP[10] and PB10_MFP1 (ALT_MFP[0]).
<table>
<tr><th>PB10_MFP1
(ALT_MFP[0])</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>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>SPISS01 (SPI0)</td></tr>
</table></td></tr>
</table>

Alternative Multiple Function Pin Control Register1 (ALT_MFP1)

Register	Offset	R/W	Description	Reset Value
ALT_MFP1	GCR_BA+0x54	R/W	Alternative Multiple Function Pin Control Register1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				PF3_MFP1		PF2_MFP1	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved		PD5_MFP1	PD4_MFP1	PD3_MFP1	PD2_MFP1	PD1_MFP1	PD0_MFP1
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

<table>
<tr><th>Bits</th><th colspan="2">Description</th></tr>
<tr><td>[31:28]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[27:26]</td><td><b>PF3_MFP1</b></td><td><b>PF.3 Pin Function Selection</b>
PF3_MFP1 (ALT_MFP1[27:26]) and GPF_MFP[3] determine the PF.3 function.
<table>
<tr><th>PF3_MFP1
(ALT_MFP1[27:26])</th><th>GPF_MFP[3]</th><th>PF.3 Function</th></tr>
<tr><td>00</td><td>0</td><td>GPIO</td></tr>
<tr><td>00</td><td>1</td><td>PS2CLK</td></tr>
<tr><td>01</td><td>1</td><td>Reserved</td></tr>
<tr><td>10</td><td>1</td><td>I2C0SCL (I^2C0)</td></tr>
<tr><td>11</td><td>1</td><td>ADC7</td></tr>
</table></td></tr>
<tr><td>[25:24]</td><td><b>PF2_MFP1</b></td><td><b>PF.2 Pin Function Selection</b>
PF2_MFP1 (ALT_MFP1[25:24]) and GPF_MFP[2] determine the PF.2 function.
<table>
<tr><th>PF2_MFP1
(ALT_MFP1[25:24])</th><th>GPF_MFP[2]</th><th>PF.2 Function</th></tr>
<tr><td>00</td><td>0</td><td>GPIO</td></tr>
<tr><td>00</td><td>1</td><td>PS2DAT</td></tr>
<tr><td>01</td><td>1</td><td>Reserevd</td></tr>
<tr><td>10</td><td>1</td><td>I2C0SDA (I^2C0)</td></tr>
<tr><td>11</td><td>1</td><td>ADC6</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>[23:22]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[21]</td><td><b>PD5_MFP1</b></td><td><b>PD.5 Pin Function Selection</b>
Bits PD5_MFP1 (ALT_MFP1[21]) and GPD_MFP[5] determine the PD.5 function.
<table>
<tr><th>PD5_MFP1
(ALT_MFP1[21])</th><th>GPD_MFP[5]</th><th>PD.5 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>1</td><td>0</td><td>MOSI21 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC5</td></tr>
</table></td></tr>
<tr><td>[20]</td><td><b>PD4_MFP1</b></td><td><b>PD.4 Pin Function Selection</b>
Bits PD4_MFP1 (ALT_MFP1[20]) and GPD_MFP[4] determine the PD.4 function.
<table>
<tr><th>PD4_MFP1
(ALT_MFP1[20])</th><th>GPD_MFP[4]</th><th>PD.4 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>1</td><td>0</td><td>MISO21 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC4</td></tr>
</table></td></tr>
<tr><td>[19]</td><td><b>PD3_MFP1</b></td><td><b>PD.3 Pin Function Selection</b>
Bits PD3_MFP1 (ALT_MFP1[19]) and GPD_MFP[3] determine the PD.3 function.
<table>
<tr><th>PD3_MPF1
(ALT_MFP1[19])</th><th>GPD_MFP[3]</th><th>PD.3 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MOSI01 (SPI0)</td></tr>
<tr><td>1</td><td>0</td><td>MOSI20 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC3</td></tr>
</table></td></tr>
<tr><td>[18]</td><td><b>PD2_MFP1</b></td><td><b>PD.2 Pin Function Selection</b>
Bits PD2_MFP1 (ALT_MFP1[18]) and GPD_MFP[2] determine the PD.2 function.
<table>
<tr><th>PD2_MFP1
(ALT_MFP1[18])</th><th>GPD_MFP[2]</th><th>PD.2 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>MISO01 (SPI0)</td></tr>
<tr><td>1</td><td>0</td><td>MISO20 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC2</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>[17]</td><td><b>PD1_MFP1</b></td><td><b>PD.1 Pin Function Selection</b>
Bits PD1_MFP1 (ALT_MFP1[17]) and GPD_MFP[1] determine the PD.1 function.
<table>
<tr><th>PD1_MFP1
(ALT_MFP1[17])</th><th>GPD_MFP[1]</th><th>PD.1 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>SPISS01 (SPI0)</td></tr>
<tr><td>1</td><td>0</td><td>SPICLK2 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC1</td></tr>
</table></td></tr>
<tr><td>[16]</td><td><b>PD0_MFP1</b></td><td><b>PD.0 Pin Function Selection</b>
Bits PD0_MFP1 (ALT_MFP1[16]) and GPD_MFP[0] determine the PD.0 function.
<table>
<tr><th>PD0_MFP1
(ALT_MFP1[16])</th><th>GPD_MFP[0]</th><th>PD.0 Function</th></tr>
<tr><td>0</td><td>0</td><td>GPIO</td></tr>
<tr><td>0</td><td>1</td><td>Reserved</td></tr>
<tr><td>1</td><td>0</td><td>SPISS20 (SPI2)</td></tr>
<tr><td>1</td><td>1</td><td>ADC0</td></tr>
</table></td></tr>
<tr><td>[15:0]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
</table>

Dynamic Function Pin Conflict Status Register (DFP_CSR)

Register	Offset	R/W	Description	Reset Value
DFP_CSR	GCR_BA+0xA0	R	Pin Conflict Status	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	**DFP6_CST**	**DFP5_CST**	**DFP4_CST**	**DFP3_CST**	**DFP2_CST**	**DFP1_CST**	**DFP0_CST**

Bits	Description	
[31:7]	**Reserved**	Reserved
[6]	**DFP6_CST**	**Conflict Status of PB.7** 1 = PB.7 is conflicted. 0 = PB.7 worked normally.
[5]	**DFP5_CST**	**Conflict Status of PB.6** 1 = PB.6 is conflicted. 0 = PB.6 worked normally.
[4]	**DFP4_CST**	**Conflict Status of PB.5** 1 = PB.5 is conflicted. 0 = PB.5 worked normally.
[3]	**DFP3_CST**	**Conflict Status of PB.4** 1 = PB.4 is conflicted. 0 = PB.4 worked normally.
[2]	**DFP2_CST**	**Conflict Status of PD.11** 1 = PD.11 is conflicted. 0 = PD.11 worked normally.
[1]	**DFP1_CST**	**Conflict Status of PD.10** 1 = PD.10 is conflicted. 0 = PD.10 worked normally.

[0]	**DFP0_CST**	**Conflict Status of PD.9** 1 = PD.9 is conflicted. 0 = PD.9 worked normally.

GPIOA IO Control Register (GPA_IOCR)

Register	Offset	R/W	Description	Reset Value
GPA_IOCR	GCR_BA+0xC0	R/W	GPIOA IO 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				**GPA11_DS**	**GPA10_DS**	**Reserved**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:12]	**Reserved**	Reserved
[11]	**GPA11_DS**	**PA.11 Pin Driving Strength Selection** 1 = PA.11 strong driving strength mode Enabled. 0 = PA.11 strong driving strength mode Disabled.
[10]	**GPA10_DS**	**PA.10 Pin Driving Strength Selection** 1 = PA.10 strong driving strength mode Enabled. 0 = PA.10 strong driving strength mode Disabled.
[9:0]	**Reserved**	Reserved

GPIOB IO Control Register (GPB_IOCR)

Register	Offset	R/W	Description	Reset Value
GPB_IOCR	GCR_BA+0xC4	R/W	GPIOB IO 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	GPB14_DS	GPB13_DS	GPB12_DS	Reserved			GPB8_DS
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
GPB7_DS	GPB6_DS	GPB5_DS	GPB4_DS	Reserved			

Bits	Description	
[31:15]	**Reserved**	Reserved
[14]	**GPB14_DS**	**PB.14 Pin Driving Strength Selection** 1 = PB.14 strong driving strength mode Enabled. 0 = PB.14 strong driving strength mode Disabled.
[13]	**GPB13_DS**	**PB.13 Pin Driving Strength Selection** 1 = PB.13 strong driving strength mode Enabled. 0 = PB.13 strong driving strength mode Disabled.
[12]	**GPB12_DS**	**PB.12 Pin Driving Strength Selection** 1 = PB.12 strong driving strength mode Enabled. 0 = PB.12 strong driving strength mode Disabled.
[11:9]	**Reserved**	Reserved
[8]	**GPB8_DS**	**PB.8 Pin Driving Strength Selection** 1 = PB.8 strong driving strength mode Enabled. 0 = PB.8 strong driving strength mode Disabled.
[7]	**GPB7_DS**	**PB.7 Pin Driving Strength Selection** 1 = PB.7 strong driving strength mode Enabled. 0 = PB.7 strong driving strength mode Disabled.
[6]	**GPB6_DS**	**PB.6 Pin Driving Strength Selection** 1 = PB.6 strong driving strength mode Enabled. 0 = PB.6 strong driving strength mode Disabled.

[5]	**GPB5_DS**	**PB.5 Pin Driving Strength Selection** 1 = PB.5 strong driving strength mode Enabled. 0 = PB.5 strong driving strength mode Disabled.
[4]	**GPB4_DS**	**PB.4 Pin Driving Strength Selection** 1 = PB.4 strong driving strength mode Enabled. 0 = PB.4 strong driving strength mode Disabled.
[3:0]	**Reserved**	Reserved

GPIOD IO Control Register (GPD_IOCR)

Register	Offset	R/W	Description	Reset Value
GPD_IOCR	GCR_BA+0xCC	R/W	GPIOD IO 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				GPD11_DS	GPD10_DS	GPD9_DS	GPD8_DS
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:12]	**Reserved**	Reserved
[11]	**GPD11_DS**	**PD.11 Pin Driving Strength Selection** 1 = PD.11 strong driving strength mode Enabled. 0 = PD.11 strong driving strength mode Disabled.
[10]	**GPD10_DS**	**PD.10 Pin Driving Strength Selection** 1 = PD.10 strong driving strength mode Enabled. 0 = PD.10 strong driving strength mode Disabled.
[9]	**GPD9_DS**	**PD.9 Pin Driving Strength Selection** 1 = PD.9 strong driving strength mode Enabled. 0 = PD.9 strong driving strength mode Disabled.
[8]	**GPD8_DS**	**PD.8 Pin Driving Strength Selection** 1 = PD.8 strong driving strength mode Enabled. 0 = PD.8 strong driving strength mode Disabled.
[7:0]	**Reserved**	Reserved

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	Description	
[31:16]	**Reserved**	Reserved
[7:0]	**REGWRPROT**	**Register Write-Protection Code (Write Only)** Some registers have write-protection function. Writing these registers has 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 Disabled for writing protected registers. 0 = Write-protection 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 enabled) **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

5.4 Clock Controller

5.4.1 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 enters Power-down mode and waits for wake-up interrupt source triggered to exit Power-down mode. In Power-down mode, the clock controller turns off the external 4~24 MHz high speed crystal (HXT) and internal 22.1184 MHz (HIRC) high speed oscillator to reduce the overall system power consumption.

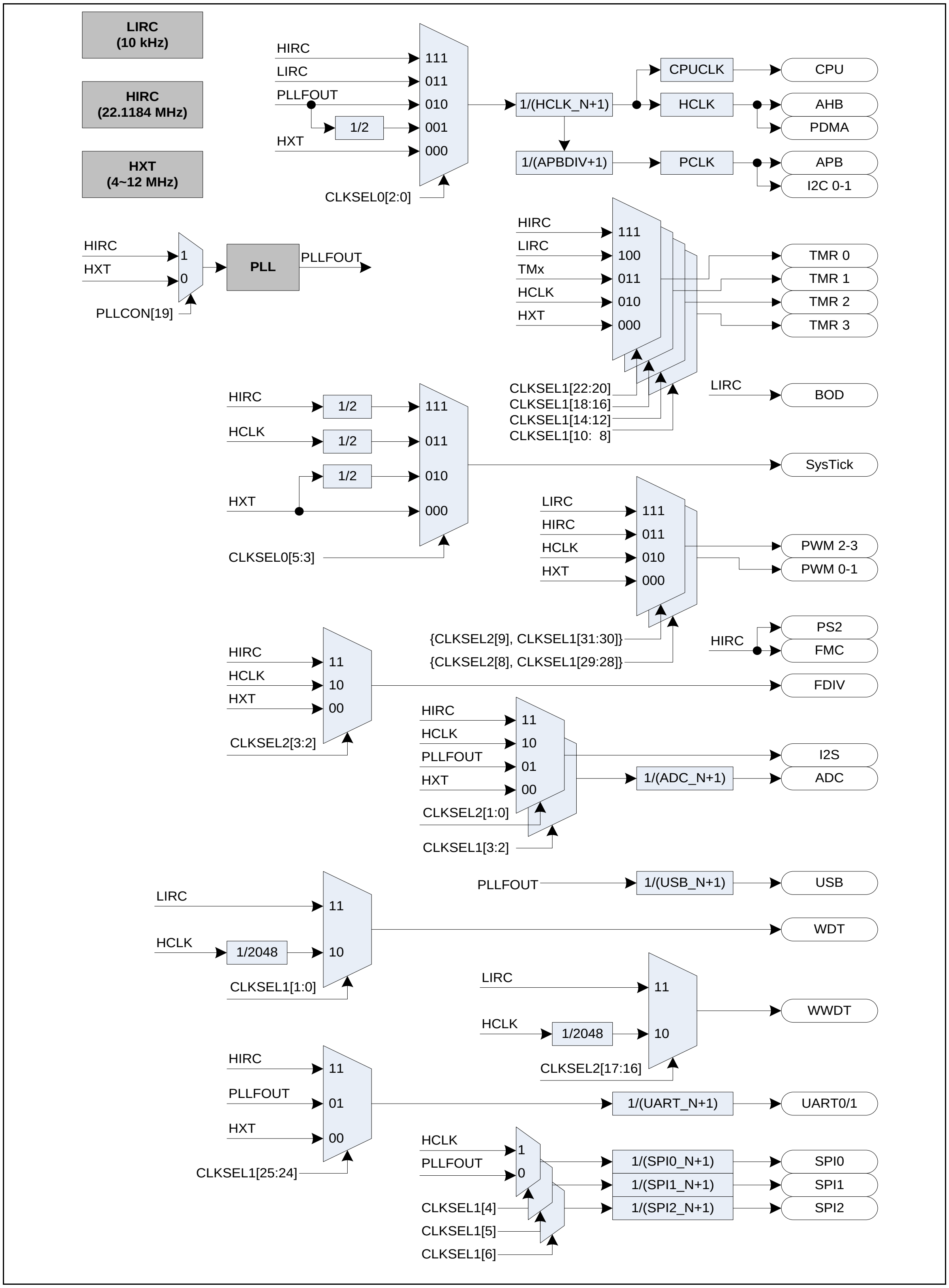


Figure 5-2 Clock Generator Global View Diagram

5.4.2 Clock Generator

The clock generator consists of 4 clock sources as listed below:

- 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

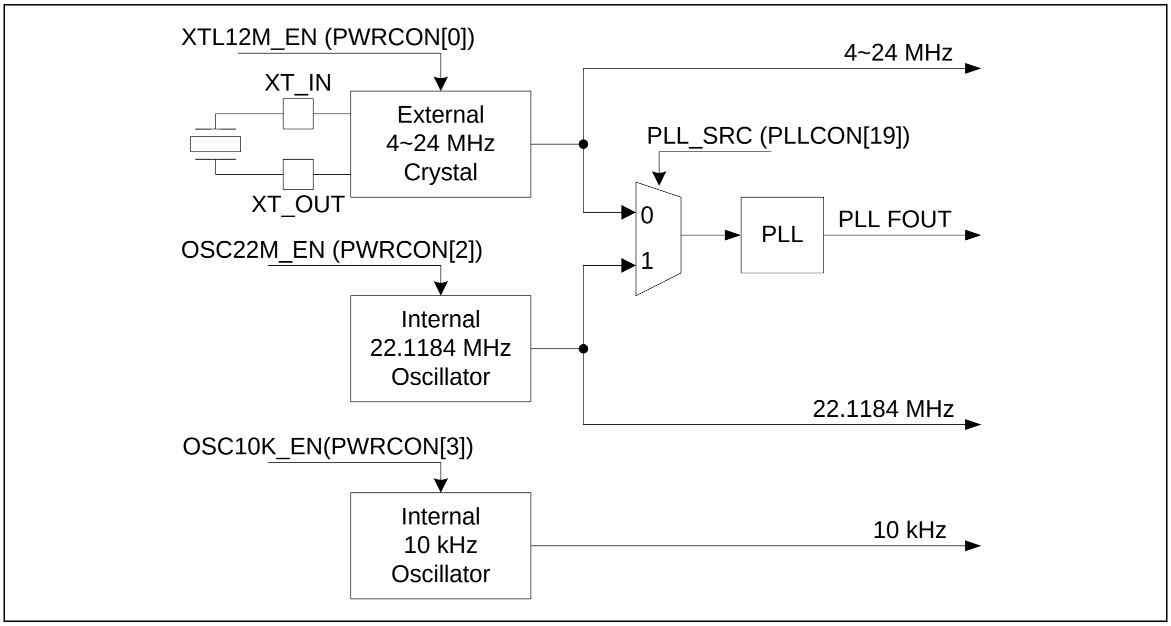


Figure 5-3 Clock Generator Block Diagram

5.4.3 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 shown in Figure 5-4.

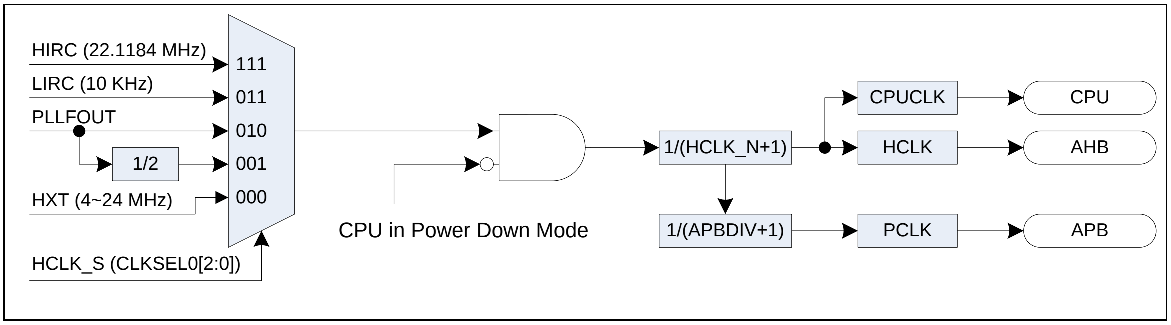


Figure 5-4 System Clock Block Diagram

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 shown in Figure 5-5.

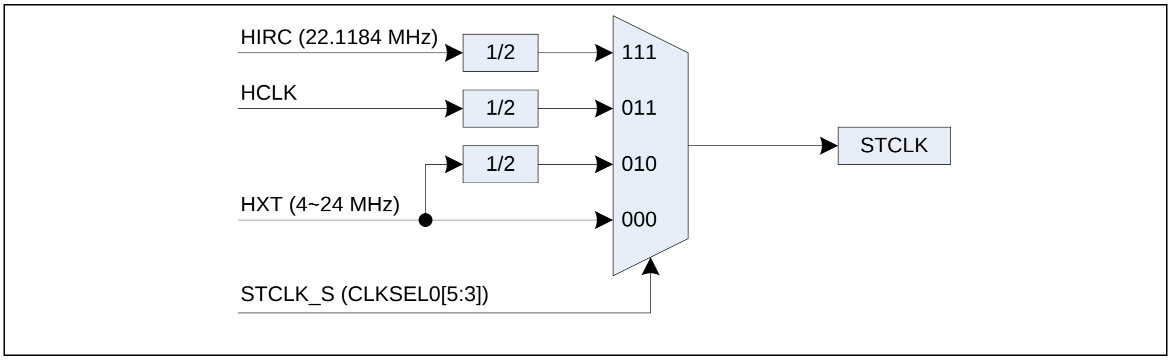


Figure 5-5 SysTick Clock Control Block Diagram

5.4.4 Peripherals Clock

The peripherals clock had different clock source switch setting depending on different peripherals. Please refer to the CLKSEL1 and CLKSEL2 register description in *5.4.8*.

5.4.5 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.

The clocks kept active are listed below:

- Clock Generator
  - Internal 10 kHz low speed oscillator clock
- Peripherals Clock (When these IP adopt 10 kHz low speed oscillator as clock source)

5.4.6 Frequency Divider Output

This device is equipped with a power-of-2 frequency divider which is composed by 16 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.

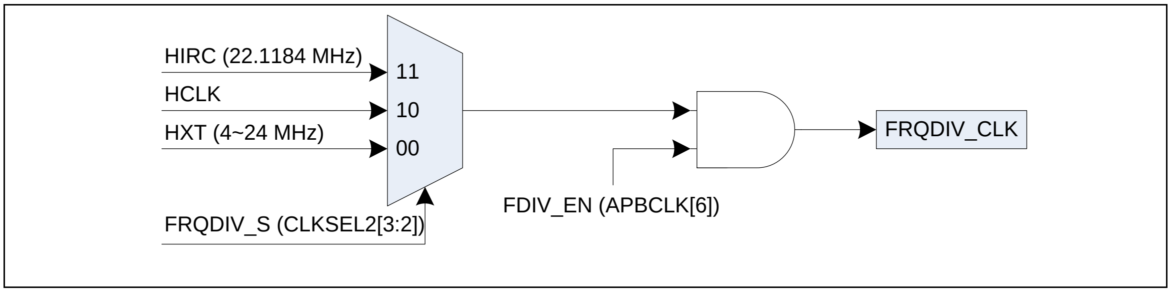


Figure 5-6 Clock Source of Frequency Divider

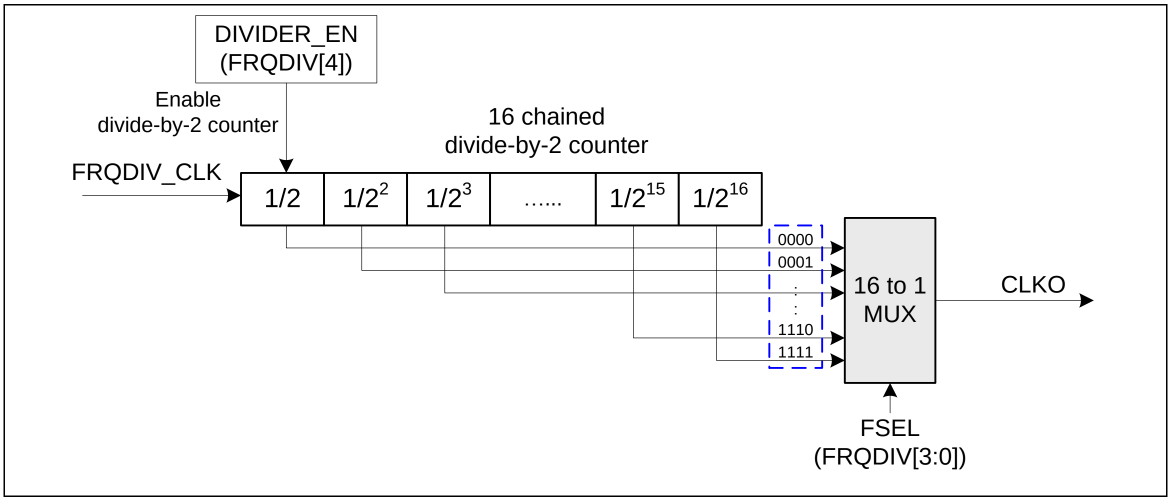


Figure 5-7 Block Diagram of Frequency Divider

5.4.7 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
APBDIV	CLK_BA+0x2C	R/W	APB Divider Control Register	0x0000_0000

5.4.8 Register Description

Power-down Control Register (PWRCON)

Except the BIT[6], all the other bits are protected, and programming these bits needs to write "59h", "16h", "88h" to address 0x5000_0100 to disable register protection. Refer to 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**	**Reserved**	**XTL12M_EN**

Bits	Description	
[31:9]	**Reserved**	Reserve
[8]	**PD_WAIT_CPU**	**Power-down Entry Conditions Control (Write-protection Bit)** 1 = Chip entering 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 entering 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 depend on the PD_WAIT_CPU bit. (a) If the PD_WAIT_CPU is 0, the chip enters Power-down mode immediately after the PWR_DOWN_EN bit set. (b) If the PD_WAIT_CPU is 1, 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. User needs to set this bit again for next Power-down. 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 internal 10 kHz low speed oscillator is not controlled by Power-down mode. In Power-down mode, the PLL and system clock are disabled, and ignored the clock source selection. The clocks of peripheral are not controlled by Power-down mode, if the peripheral clock source is from the internal 10 kHz low speed oscillator. 1 = Chip entering the Power-down mode instantly 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”, which indicates that resume from Power-down mode” The flag is set if the GPIO, USB, UART, WDT, CAN, ACMP or BOD wake-up occurred. Write 1 to clear the bit to zero. **Note:** This bit works only when 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 = Disabled. 1 = Enabled. 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 = Clock cycles delay Enabled. 0 = Clock cycles delay Disabled.
[3]	**OSC10K_EN**	**Internal 10 kHz Low Speed Oscillator Enable (Write-protection Bit)** 1 = Internal 10 kHz low speed oscillator Enabled. 0 = Internal 10 kHz low speed oscillator Disabled.
[2]	**OSC22M_EN**	**Internal 22.1184 MHz High Speed Oscillator Enable (Write-protection Bit)** 1 = Internal 22.1184 MHz high speed oscillator Enabled. 0 = Internal 22.1184 MHz high speed oscillator Disabled.
[1]	**Reserved**	Reserved

[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 = External 4~24 MHz high speed crystal Enabled 0 = External 4~24 MHz high speed crystal Disabled

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 entering Sleep Mode)**	0	0	YES	Only CPU clock is disabled
Power-down Mode	1	0	NO	Most clocks are disabled except 10 kHz, only WDT/Timer/PWM peripheral clock are still enabled.
Power-down Mode **(CPU entering Deep Sleep Mode)**	1	1	YES	Most clocks are disabled except 10 kHz, only WDT/Timer/PWM peripheral clock are still enabled.

Table 5-5 Power-Down Mode Control Table

When chip enters 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 are used to enable/disable clock for Flash ISP and PDMA Controllers on AHB bus.

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

Bits	Description	
[31:3]	**Reserved**	Reserved
[2]	**ISP_EN**	**Flash ISP Controller Clock Enable Control** 1 = Flash ISP engine clock Enabled. 0 = Flash ISP engine clock Disabled.
[1]	**PDMA_EN**	**PDMA Controller Clock Enable Control** 1 = PDMA engine clock Enabled. 0 = PDMA engine clock Disabled.
[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	**Reserved**	**I2S_EN**	**ADC_EN**	**USBD_EN**	**Reserved**	**Reserved**	**Reserved**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved	**Reserved**	**PWM23_EN**	**PWM01_EN**	**Reserved**	**Reserved**	**UART1_EN**	**UART0_EN**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved	**SPI2_EN**	**SPI1_EN**	**SPI0_EN**	**Reserved**	**Reserved**	**I2C1_EN**	**I2C0_EN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**FDIV_EN**	**TMR3_EN**	**TMR2_EN**	**TMR1_EN**	**TMR0_EN**	**Reserved**	**WDT_EN**

Bits	Description	
[31]	**PS2_EN**	**PS/2 Clock Enable** 1 = PS/2 clock Enabled. 0 = PS/2 clock Disabled.
[30]	**Reserved**	Reserved
[29]	**I2S_EN**	**I^{2}S Clock Enable** 1 = I^{2}S Clock Enabled. 0 = I^{2}S Clock Disabled.
[28]	**ADC_EN**	**Analog-Digital-Converter (ADC) Clock Enable** 1 = ADC clock Enabled. 0 = ADC clock Disabled.
[27]	**USBD_EN**	**USB 2.0 FS Device Controller Clock Enable** 1 = USB clock Enabled. 0 = USB clock Disabled.
[26:22]	**Reserved**	Reserved
[21]	**PWM23_EN**	**PWM_23 Clock Enable** 1 = PWM23 clock Enabled. 0 = PWM23 clock Disabled.

[20]	**PWM01_EN**	**PWM_01 Clock Enable** 1 = PWM01 clock Enabled. 0 = PWM01 clock Disabled.
[19:18]	**Reserved**	Reserved
[17]	**UART1_EN**	**UART1 Clock Enable** 1 = UART1 clock Enabled. 0 = UART1 clock Disabled.
[16]	**UART0_EN**	**UART0 Clock Enable** 1 = UART0 clock Enabled. 0 = UART0 clock Disabled.
[15]	**Reserved**	Reserved
[14]	**SPI2_EN**	**SPI2 Clock Enable** 1 = SPI2 clock Enabled. 0 = SPI2 clock Disabled.
[13]	**SPI1_EN**	**SPI1 Clock Enable** 1 = SPI1 clock Enabled. 0 = SPI1 clock Disabled.
[12]	**SPI0_EN**	**SPI0 Clock Enable** 1 = SPI0 clock Enabled. 0 = SPI0 clock Disabled.
[11:10]	**Reserved**	Reserved
[9]	**I2C1_EN**	**I^{2}C1 Clock Enable** 1 = I^{2}C1 clock Enabled. 0 = I^{2}C1 clock Disabled.
[8]	**I2C0_EN**	**I^{2}C0 Clock Enable** 1 = I^{2}C0 clock Enabled. 0 = I^{2}C0 clock Disabled.
[7]	**Reserved**	Reserved
[6]	**FDIV_EN**	**Frequency Divider Output Clock Enable** 1 = FDIV clock Enabled. 0 = FDIV clock Disabled.
[5]	**TMR3_EN**	**Timer3 Clock Enable** 1 = Timer3 clock Enabled. 0 = Timer3 clock Disabled.
[4]	**TMR2_EN**	**Timer2 Clock Enable** 1 = Timer2 clock Enabled. 0 = Timer2 clock Disabled.

[3]	**TMR1_EN**	**Timer1 Clock Enable** 1 = Timer1 clock Enabled. 0 = Timer1 clock Disabled.
[2]	**TMR0_EN**	**Timer0 Clock Enable** 1 = Timer0 clock Enabled. 0 = Timer0 clock Disabled.
[1]	**Reserved**	Reserved
[0]	**WDT_EN**	**Watchdog Timer Clock Enable (Write-protection Bit)** This bit is the protected bit, which means programming it needs to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Refer to the register REGWRPROT at address GCR_BA+0x100. 1 = Watchdog Timer Clock Enabled. 0 = Watchdog Timer Clock Disabled.

Clock status Register (CLKSTATUS)

The bits of this register are used to monitor if the chip clock source is stable or not, and if clock switch is 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**	**Reserved**	**XTL12M_STB**

Bits	Description	
[31:8]	**Reserved**	Reserved
[7]	**CLK_SW_FAIL**	**Clock Switching Fail Flag (Write-protection Bit)** 1 = Clock switching failed. 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]	**Reserved**	Reserved
[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	Description	
[31:6]	**Reserved**	Reserved
[5:3]	**STCLK_S**	**Cortex_M0 SysTick Clock Source Selection (Write-protection Bits)** If SYST_CSR[2]= 0, SysTick uses clock source listed below. These bits are protected bit, which means programming them needs to write "59h", "16h", "88h" to address 0x5000_0100 to disable register protection. Refer to the register REGWRPROT at address GCR_BA+0x100. 000 = Clock source from external 4~24 MHz high 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 Selection (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. Refer to the register REGWRPROT at address GCR_BA+0x100. 000 = Clock source from external 4~24 MHz high speed crystal clock 001 = Clock source from PLL clock/2 010 = 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	**SPI2_S**	**SPI1_S**	**SPI0_S**	**ADC_S**		**WDT_S**	

Bits	Description	
[31:30]	**PWM23_S**	**PWM2 and PWM3 Clock Source Select Bit [1:0]** PWM2 and PWM3 use the same Engine clock source, and both of them use the same prescaler. The Engine clock source of PWM2 and PWM3 is defined by PWM23_S[2:0] and this field is combined by CLKSEL2[9] and CLKSEL1[31:30]. 000 = Clock source from external 4~24 MHz high speed crystal clock 010 = Clock source from HCLK 011 = Clock source from internal 22.1184 MHz high speed oscillator clock 111 = Clock source from internal 10 kHz low speed oscillator clock
[29:28]	**PWM01_S**	**PWM0 and PWM1 Clock Source Select Bit [1:0]** PWM0 and PWM1 use the same Engine clock source, and both of them use the same prescaler The Engine clock source of PWM0 and PWM1 is defined by PWM01_S[2:0] and this field is combined by CLKSEL2[8] and CLKSEL1[29:28]. 000 = Clock source from external 4~24 MHz high speed crystal clock. 010 = Clock source from HCLK. 011 = Clock source from internal 22.1184 MHz high speed oscillator clock. 111 = Clock source from internal 10 kHz low speed oscillator clock.
[27:26]	**Reserved**	Reserved

[25:24]	**UART_S**	**UART Clock Source Selection** 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 Selection** 000 = Clock source from external 4~24 MHz high speed crystal clock. 010 = Clock source from HCLK. 011 = Reserved. 100 = Clock source from internal 10 kHz low speed oscillator clock. 111 = Clock source from internal 22.1184 MHz high speed oscillator clock.
[19]	**Reserved**	Reserved
[18:16]	**TMR2_S**	**TIMER2 Clock Source Selection** 000 = Clock source from external 4~24 MHz high speed crystal clock. 010 = Clock source from HCLK. 011 = Clock source from external clock source TM2. 100 = Clock source from internal 10 kHz low speed oscillator clock. 111 = Clock source from internal 22.1184 MHz high speed oscillator clock.
[15]	**Reserved**	Reserved
[14:12]	**TMR1_S**	**TIMER1 Clock Source Selection** 000 = Clock source from external 4~24 MHz high speed crystal clock. 010 = Clock source from HCLK. 011 = Clock source from external clock source TM1. 100 = Clock source from internal 10 kHz low speed oscillator clock. 111 = Clock source from internal 22.1184 MHz high speed oscillator clock.
[11]	**Reserved**	Reserved
[10:8]	**TMR0_S**	**TIMER0 Clock Source Selection** 000 = Clock source from external 4~24 MHz high speed crystal clock. 010 = Clock source from HCLK. 011 = Clock source from external clock source TM0. 100 = Clock source from internal 10 kHz low speed oscillator clock. 111 = Clock source from internal 22.1184 MHz high speed oscillator clock.
[7]	**Reserved**	Reserved
[6]	**SPI2_S**	**SPI2 Clock Source Selection** 1 = Clock source from HCLK. 0 = Clock source from PLL clock.

[5]	**SPI1_S**	**SPI1 Clock Source Selection** 1 = Clock source from HCLK. 0 = Clock source from PLL clock.
[4]	**SPI0_S**	**SPI0 Clock Source Selection** 1 = Clock source from HCLK. 0 = Clock source from PLL clock.
[3:2]	**ADC_S**	**ADC Clock Source Selection** 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 Selection (Write-protection Bits)** These bits are protected bits and programming this needs to write “59h”, “16h”, “88h” to address 0x5000_0100 to disable register protection. Refer to the register REGWRPROT at address GCR_BA+0x100. 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						**WWDT_S**	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved						**PWM23_S[2]**	**PWM01_S[2]**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**FRQDIV_S**		**I2S_S**	

Bits	Description	
[31:18]	**Reserved**	Reserved
[17:16]	**WWDT_S**	**Windowed-Watchdog Timer Clock Source Selection (Write-protection Bits)** 10 = Clock source from HCLK/2048 clock. 11 = Clock source from internal 10 kHz low speed oscillator clock.
[15:10]	**Reserved**	Reserved
[9]	**PWM23_S[2]**	**PWM2 and PWM3 Clock Source Select Bit [2]** PWM2 and PWM3 use the same Engine clock source, and both of them use the same prescaler. The Engine clock source of PWM2 and PWM3 is defined by PWM23_S[2:0] and this field is combined by CLKSEL2[9] and CLKSEL1[31:30]. 000 = Clock source from external 4~24 MHz high speed crystal clock. 010 = Clock source from HCLK. 011 = Clock source from internal 22.1184 MHz high speed oscillator clock. 111 = Clock source from internal 10 kHz low speed oscillator clock.

[8]	**PWM01_S[2]**	**PWM0 and PWM1 Clock Source Select Bit [2]** PWM0 and PWM1 use the same Engine clock source, and both of them use the same prescaler. The Engine clock source of PWM0 and PWM1 is defined by PWM01_S[2:0] and this field is combined by CLKSEL2[8] and CLKSEL1[29:28]. 000 = Clock source from external 4~24 MHz high speed crystal clock. 010 = Clock source from HCLK. 011 = Clock source from internal 22.1184 MHz high speed oscillator clock. 111 = Clock source from internal 10 kHz low speed oscillator clock.
[7:4]	**Reserved**	Reserved
[3:2]	**FRQDIV_S**	**Clock Divider Clock Source Selection** 00 = Clock source from external 4~24 MHz high speed crystal clock. 01 = Reserved. 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 Selection** 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	Description	
[31:24]	**Reserved**	Reserved
[23:16]	**ADC_N**	**ADC Clock Divide Number from ADC Clock Source** 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** UART clock frequency = (UART clock source frequency) / (UART_N + 1)
[7:4]	**USB_N**	**USB Clock Divide Number from PLL Clock** USB clock frequency = (PLL frequency) / (USB_N + 1)
[3:0]	**HCLK_N**	**HCLK Clock Divide Number from HCLK Clock Source** 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 used 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

<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="4">Reserved</td><td>PLL_SRC</td><td>OE</td><td>BP</td><td>PD</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="2">OUT_DV</td><td colspan="5">IN_DV</td><td>FB_DV</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">FB_DV</td></tr>
</table>

Bits	Description	
[31:20]	**Reserved**	Reserved
[19]	**PLL_SRC**	**PLL Source Clock Selection** 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 Enabled. 1 = PLL FOUT is fixed low.
[17]	**BP**	**PLL Bypass Control** 0 = PLL is in Normal mode (default). 1 = PLL clock output is the same as clock input (XTALin).
[16]	**PD**	**Power-down Mode** If the PWR_DOWN_EN bit set to 1 in PWRCON register, the PLL will also enter Power-down mode. 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:

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

Constraint:

1. $3.2MHz < FIN < 150MHz$
2. $800KHz < \frac{FIN}{2 * NR} < 8MHz$
3. $100MHz < FCO = FIN \times \frac{NF}{NR} < 200MHz$
   $120MHz < FCO$ 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	Description	
[31:5]	**Reserved**	Reserved
[4]	**DIVIDER_EN**	**Frequency Divider Enable Bit** 0 = Frequency Divider Disabled. 1 = Frequency Divider Enabled.
[3:0]	**FSEL**	**Divider Output Frequency Select Bits** The formula of output frequency is: $\mathbf{F_{out} = F_{in}/2^{(N+1)}}$ F_{in} is the input clock frequency. F_{out} is the frequency of divider output clock. N is the 4-bit value of FSEL[3:0].

APB Divider Control Register (APBDIV)

Register	Offset	R/W	Description	Reset Value
APBDIV	CLK_BA+ 2C	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							**APBDIV**

Bits	Description	
[31:1]	**Reserved**	Reserved
[0]	**APBDIV**	**APB Divider Enable Bit** 0: PCLK = HCLK 1: PCLK = HCLK / 2

5.5 FLASH MEMORY CONTROLLER (FMC)

5.5.1 Overview

The NuMicro™ NUC123 is equipped with 68/36 K bytes on-chip embedded flash for application program memory (APROM) and data flash that can be updated through ISP procedure. In System Programming (ISP) function enables user to update chip embedded flash when chip is soldered on PCB. After chip is powered on, Cortex-M0 CPU fetches code from APROM or LDROM decided by boot select (CBS) in Config0. By the way, NuMicro™ NUC123 also provides data flash for user, to store some application dependent data before chip power off.

The NuMicro™ NUC123 supports another flexible feature: configurable data flash size. The data flash size is decided by data flash variable size enable (DFVSEN), data flash enable (DFEN) in Config0 and data flash base address (DFBADR) in Config1. When DFVSEN is set to 1, the data flash size is fixed at 4K and the address is started from 0x0001_f000, and the APROM size is become 64/32K. When DFVSEN is set to 0 and DFEN is set to 1, the data flash size is zero and the APROM size is 68/36K bytes.. When DFVSEN is set to 0 and DFEN is set to 0, the APROM and data flash share 68/36K bytes continuous address and the start address of data flash is defined by (DFBADR) in Config1.

5.5.2 Features

- Runs up to 72 MHz and optional up to 50 MHz with zero wait state for continuous address read access
- 68/36 K bytes application program memory (APROM) and data flash
- 4KB in system programming (ISP) loader program memory (LDROM)
- Configurable Data flash size with 512 bytes page erase unit
- In System Program (ISP) to update on chip Flash

5.5.3 Block Diagram

The flash memory controller consists of AHB slave interface, ISP control logic and flash macro interface timing control logic. The block diagram of flash memory controller is shown as follows.

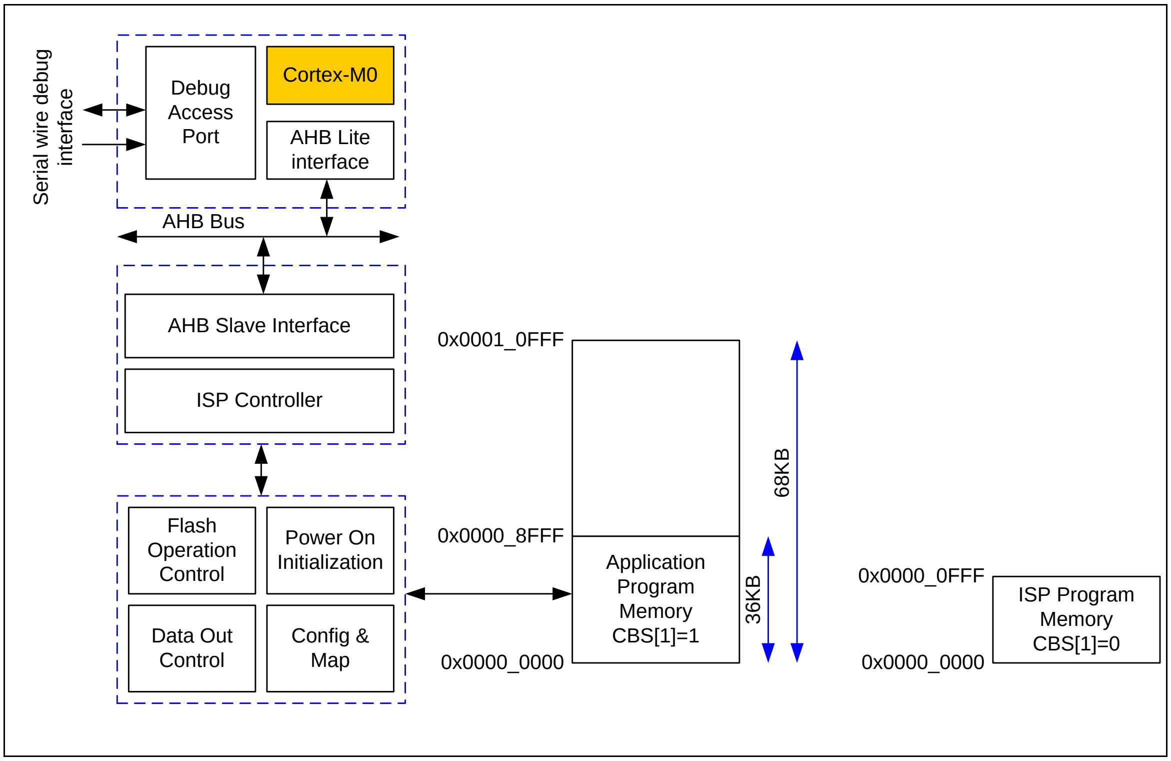


Figure 5-8 Flash Memory Control Block Diagram

5.5.4 Flash Memory Organization

The flash memory consists of application program memory (APROM), data flash, ISP loader program memory (LDROM) and user configuration. User configuration block provides two words to control system logic, such as flash security lock, boot select, Brown-out voltage level, data flash base address, etc. It works like a fuse for power on setting. It is loaded from flash memory to its corresponding control registers during chip powered on. User can set these bits according to different application request. The data flash start address and its size can be defined by user depending on the application.

Block Name	Size	Start Address	End Address
AP-ROM	64 KB 32 KB	0x0000_0000	0x0000_FFFF (64 KB) 0x0000_7FFF (32 KB)
Data Flash	4 KB	0x0001_F000	0x0001_FFFF
LD-ROM	4 KB	0x0010_0000	0x0010_0FFF
User Configuration	2 words	0x0030_0000	0x0030_0004

Table 5-6 Memory Address Map (DFVSEN = 1)

Block Name	Size	Start Address	End Address
AP-ROM	(36-0.5*N)/(68-0.5*N) KB	0x0000_0000	0x0001_0FFF (68KB, if DFEN=1) 0x0000_8FFF (36KB, if DFEN=1) DFBADR-1 (if DFEN=0)
Data Flash	0.5*N KB (if DFEN=0)	DFBADR (if DFEN=0)	0x0001_0FFF (68KB, if DFEN=0) 0x0000_8FFF (36KB, if DFEN=0)
LD-ROM	4 KB	0x0010_0000	0x0010_0FFF
User Configuration	2 words	0x0030_0000	0x0030_0004

Note: N is the page number of configured data flash. One page size is 512 bytes.

Table 5-7 Memory Address Map (DFVSEN = 0)

The flash memory organization is shown below:

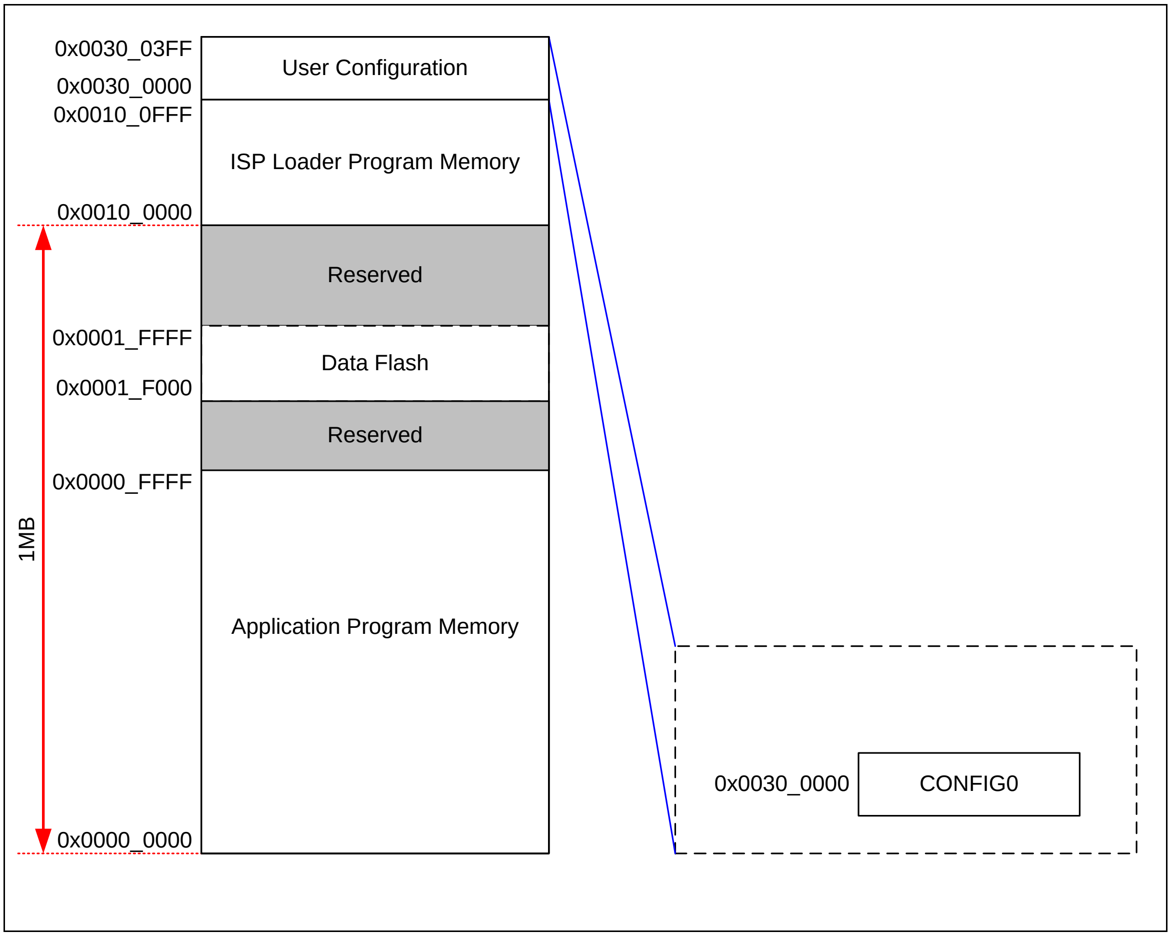


Figure 5-9 Flash Memory Organization (DFVSEN = 1)

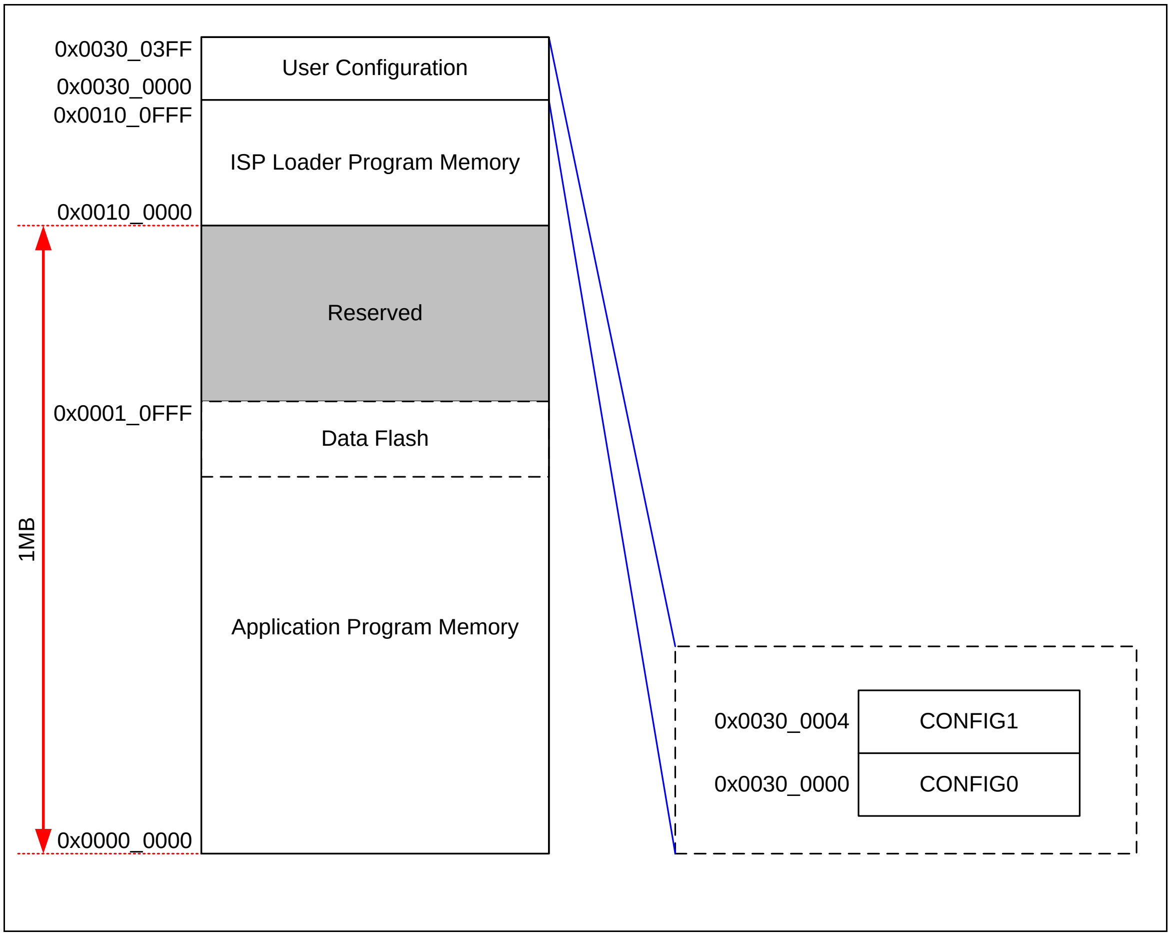


Figure 5-10 Flash Memory Organization (DFVSEN = 0)

5.5.5 Boot Selection

The NuMicro™ NUC123 Series provides the In-System-Programming (ISP) feature to enable user to update program memory by a stand-along ISP firmware. A dedicated 4 KB program memory (LDROM) is used to store ISP firmware. User can select to start program fetch from APROM or LDROM by CBS[1] in Config0.

In addition to set boot from APROM or LDROM, CBS in Config0 also used to control system memory map after booting. When CBS[0] = 1 and set CBS[1] = 1 to boot from APROM, the application in APROM will not be able to access LDROM by memory read. In other words, when CBS[0] = 1 and set CBS[1] = 0 to boot from LDROM, the software executed in LDROM will not be able to access APROM by memory read. Figure 5-11 shows the memory map when boot from APROM and LDROM.

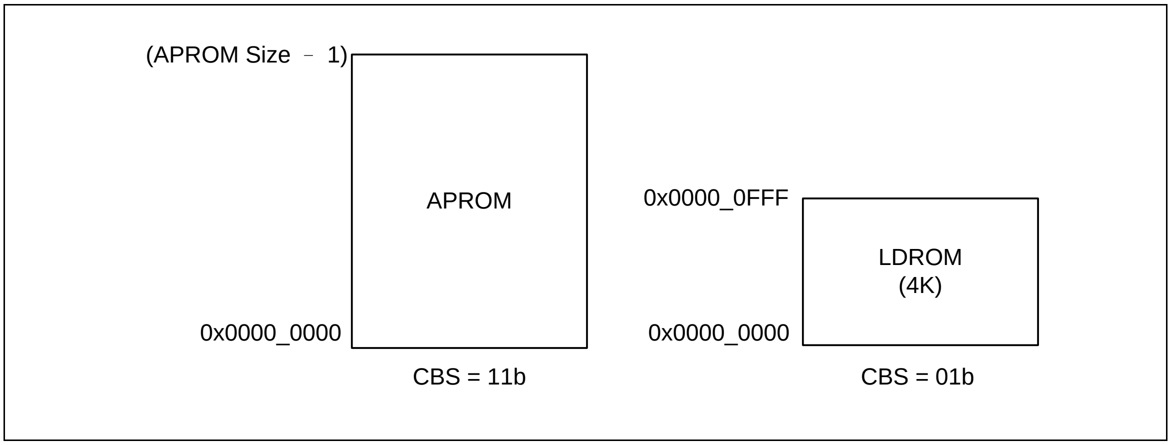


Figure 5-11 Program Executing Range for boot from APROM and boot from LDROM

For the application that software needs to execute code in APROM and call the functions in LDROM or to execute code in LDROM and call the APROM function without changing boot mode, CBS[0] needs to be set as 0 and this is called In-Application-Programming(IAP).

5.5.6 In Application Programming

The NuMicro™ NUC123 Series provides a new In-application-programming (IAP) function for user to switch the code executing between APROM and LDROM without a reset. User can enable the IAP function by re-booting chip and setting the chip boot selection bits in Config0 (CBS[1:0]) as 10b or 00b.

In the case that NuMicro™ NUC123 boots from APROM with the IAP function enabled (CBS[1:0] = 10b), the executable range of code includes all of APROM and LDROM. The address space of APROM is kept as the original size but the address space of the 4 KB LDROM is mapped to 0x0010_0000~ 0x0010_0FFF.

In the case that NuMicro™ NUC123 boots from LDROM with the IAP function enabled (CBS[1:0] = 00b), the executable range of code includes all of LDROM and almost all of APROM except for its first page. User cannot access the first page of APROM because the first page of executable code range becomes the mirror of the first page of LDROM as set by default. Meanwhile, the address space of 4 KB LDROM is mapped to 0x0010_0000~0x0010_0FFF.

Please refer to the following figure for the address map while IAP is activating.

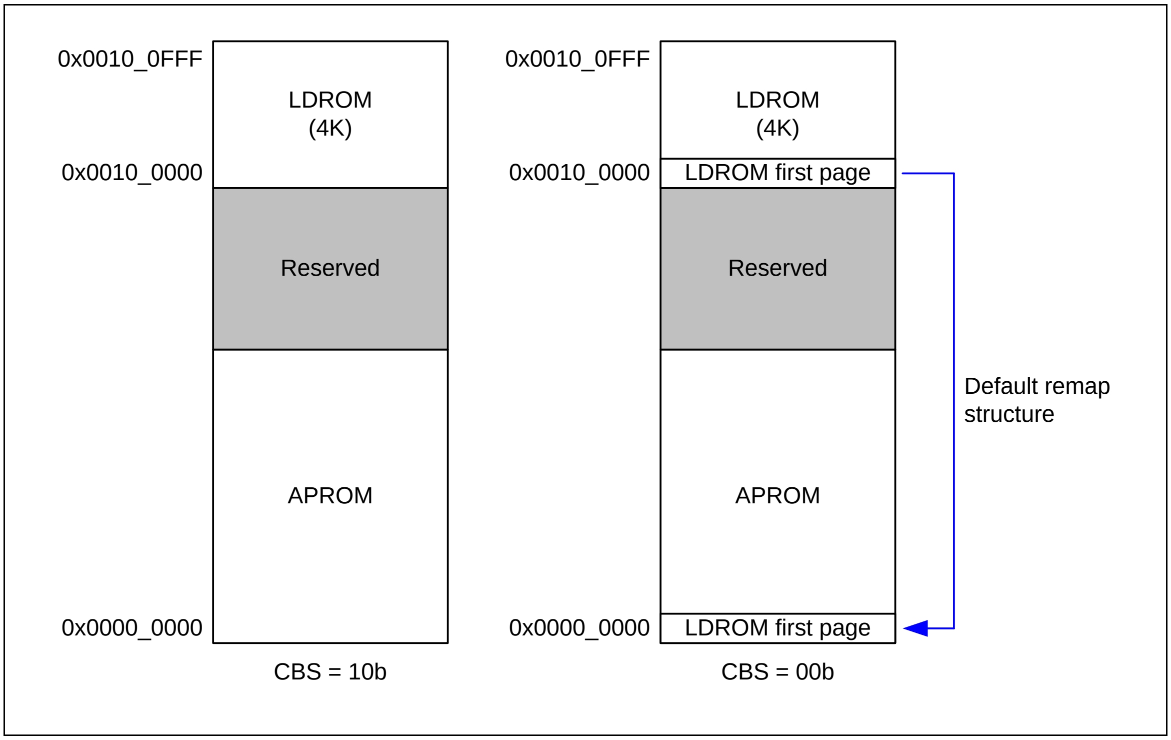


Figure 5-12 Program Executing Range of IAP

When chip boots with the IAP function enabled, any other page within the executable range of code can be mirrored to the first page of executable code (0x0000_0000~0x0000_01FF) any time. User can change the remap address of the first executing page by filling the target remap address to ISPADR and then go through ISP procedure with the Vector Page Re-map command. After the change, user can check if the remap address change is successful by reading the VECMAP field in the ISPSTA register.

5.5.7 Data Flash

The NuMicro™ NUC123 provides data flash for user to store data, which is read or written 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.

When data flash variable size enable (DFVSEN) bit in Config0 is 1, the application program memory (APROM) is 64/32 K bytes and the data flash size is 4 K bytes. The start address of data flash is fixed at 0x0001_F000. When DFVSEN is set to 0, the application program memory (APROM) is 68/36 K bytes and the data flash is shared with APROM with variable size defined by user. If data flash enable (DFEN) bit in Config0 is 1, there is no data flash and all 68/36 K bytes size is used for APROM. If data flash enable (DFEN) bit in Config0 is 0, the data flash share with APROM and its base address is defined by data flash base address (DFBADR) bits in Config1. Under this setting, the application program memory size is (68/36-0.5*N)KB and data flash size is 0.5*N KB.

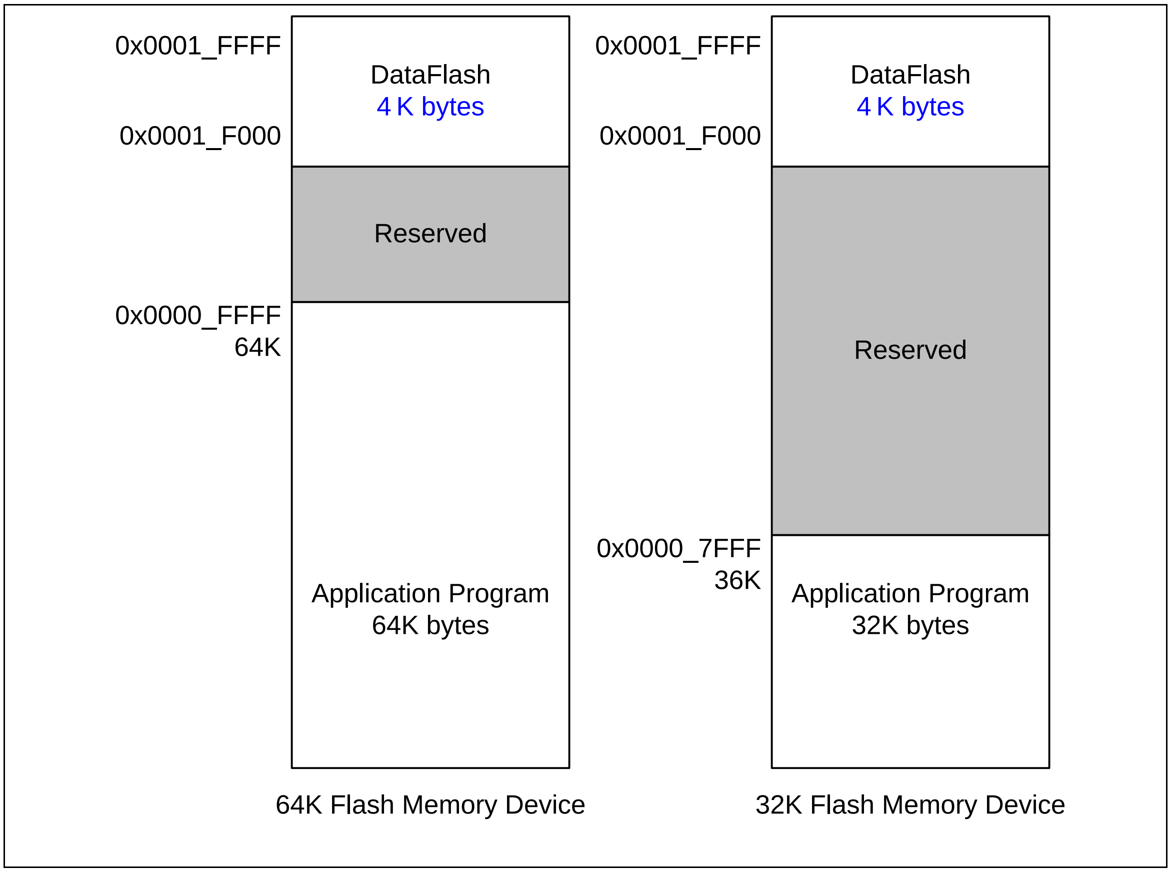


Figure 5-13 Flash Memory Structure (DFVSEN = 1)

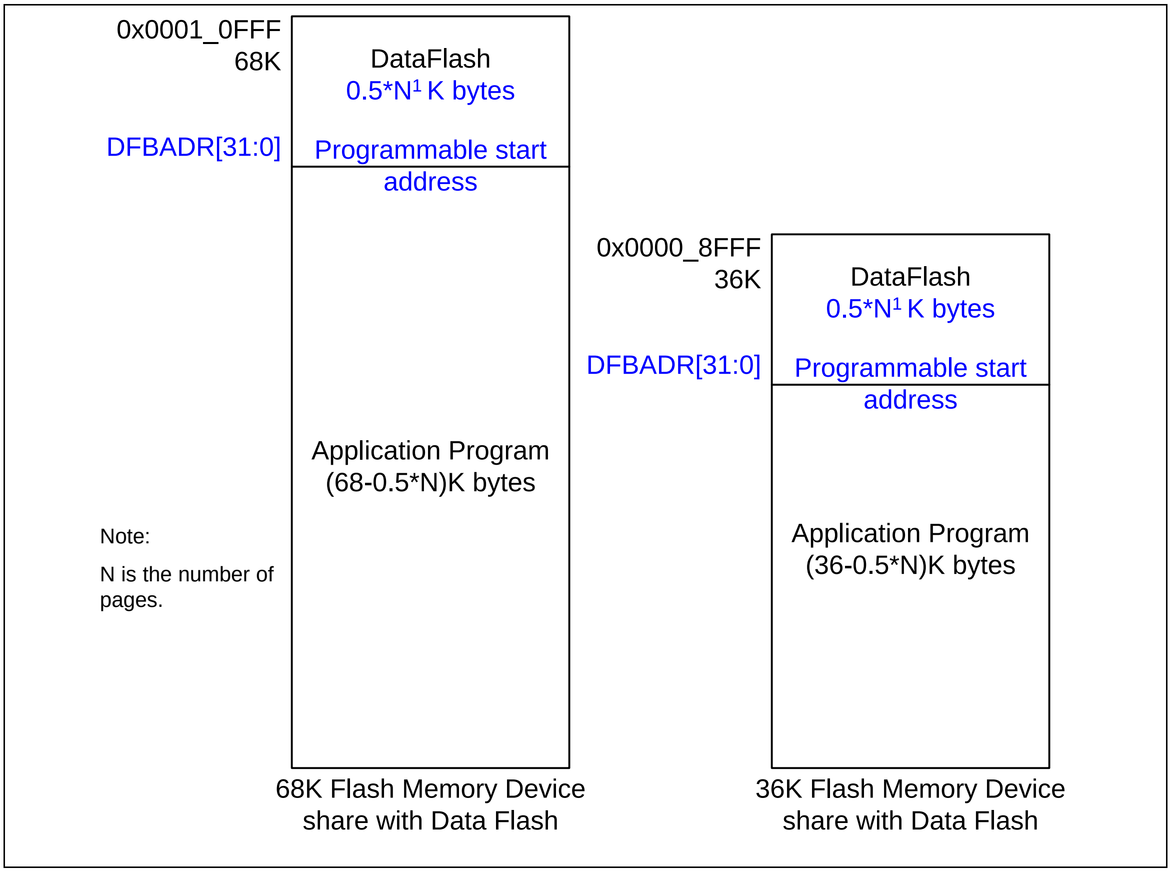


Figure 5-14 Flash Memory Structure (DFVSEN = 0)

5.5.8 User Configuration

Config0 (Address = 0x0030_0000)

31	30	29	28	27	26	25	24
CWDTEN	CWDTPDEN	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			DFVSEN	LOCK	DFEN

<table>
<tr><th>Bits</th><th colspan="2">Description</th></tr>
<tr><td>[31]</td><td><b>CWDTEN</b></td><td><b>Watchdog hardware Enable</b>
0 = Window Watchdog Timer Enabled when chip is powered on.
1 = Window Watchdog Timer Disabled when chip is powered on.</td></tr>
<tr><td>[30]</td><td><b>CWDTPDEN</b></td><td><b>Watchdog Clock Power-down Enable</b>
0 = Watchdog Timer clock kept enabled when chip enters Power-down.
1 = Watchdog Timer clock is controlled by OSC10K_EN (PWRCON[3]) when chip enters Power-down.
<b>Note:</b> This bit only works if CWDTEN is set to 0</td></tr>
<tr><td>[29]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[28]</td><td><b>CKF</b></td><td><b>XT1 Clock Filter Enable</b>
0 = XT1 clock filter Disabled.
1 = XT1 clock filter Enabled.</td></tr>
<tr><td>[27]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[26:24]</td><td><b>CFOSC</b></td><td><b>CPU Clock Source Selection After Reset</b>
<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 loaded to CLKSEL0.HCLK_S[2:0] in system register after any reset occurs.</td></tr>
<tr><td>[23]</td><td><b>CBODEN</b></td><td><b>Brown-out Detector Enable</b>
0= Brown-out detect Enabled after powered on.
1= Brown-out detect Disabled after powered on.</td></tr>
</table>

<table>
<tr><td>[22:21]</td><td><b>CBOV1-0</b></td><td><b>Brown-out Voltage Selection</b>
<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>
<tr><td>[20]</td><td><b>CBORST</b></td><td><b>Brown-out Reset Enable</b>
0 = Brown-out reset Enabled after powered on.
1 = Brown-out reset Disabled after powered on.</td></tr>
<tr><td>[19:8]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[7:6]</td><td><b>CBS</b></td><td><b>Config Boot Selection</b>
<table>
<tr><th>CBS[1:0]</th><th>Boot Selection</th></tr>
<tr><td>11</td><td>Chip booting from APROM and program executing range only including APROM. LDROM cannot be access by program directly, except by through ISP.
APROM is write-protected in this mode.</td></tr>
<tr><td>01</td><td>Chip booting from LDROM, program executing range only including LDROM; APROM cannot be access by program directly, except by through ISP.
APROM can be updated in this mode.</td></tr>
<tr><td>10</td><td>Chip booting from APROM, program executing range including LDROM and APROM
LDROM address is mapping to 0x0010_0000~0x0010_0FFF
APROM can be updated in this mode.
The address 0x0000_0000 ~ 0x0000_01FF can be re-mapping to any other page within executing range though ISP command.[PT1]</td></tr>
<tr><td>00</td><td>Chip booting from LDROM, program executing range including LDROM and most of APROM (all but except first 512 bytes, because the first 512 bytes is mapped from LDROM)
LDROM address is mapping to 0x0010_0000 ~ 0x0010_0FFF, and also the first 512 bytes of LDROM is mapping to the address 0x0000_0000 ~ 0x0000_01FF.
APROM can be updated in this mode.
The address 0x0000_0000 ~ 0x0000_01FF can be re-mapping to any other page within executing range though ISP command.[PT2]</td></tr>
</table></td></tr>
<tr><td>[5:3]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[2]</td><td><b>DFVSEN</b></td><td><b>DATA Flash Variable Size Enable</b>
0 = Data flash size is variable and it base address is based on DFBADR (Config1).
1 = Data flash size is fixed at 4K bytes.</td></tr>
</table>

[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** 0 = Data flash Enabled. 1 = Data flash Disabled. **Note:** This bit only workable if DFVSEN is set to 0. When DFVSEN is 0 and DFEN is 1, there is no data flash and APROM size is 68K bytes. When DFVSEN is 0 and DFEN is 0, the data flash is shared with APROM within 68K bytes, and the base address of data flash is decided by DFBADR (Config1).

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	Description	
[31:20]	**Reserved**	Reserved (It is mandatory to program 0x00 to these Reserved bits.)
[19:0]	**DFBADR**	**Data Flash Base Address** (This register works only when DFEN set to 0) If DFEN is set to 0, the 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.

5.5.9 In System Program (ISP)

The application 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™ NUC123 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.

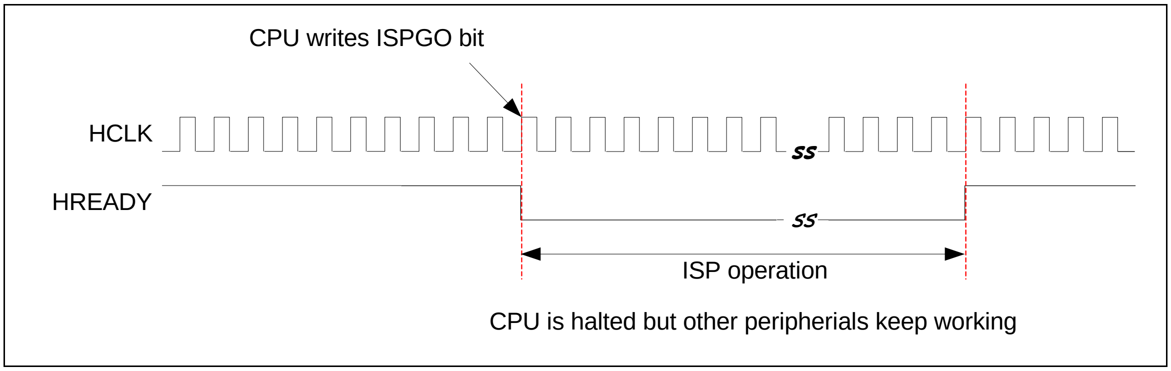
ISP is performed without removing the microcontroller from the system. Various interfaces enable LDROM firmware to get new program code easily. The most common method to perform ISP is via UART along with the firmware in LDROM. General speaking, PC transfers the new APROM code through serial port. Then LDROM firmware receives it and re-programs into APROM through ISP commands. The ISP firmware and PC application program for NuMicro™ NUC123 Series enables user to easily perform ISP through Nuvoton ISP tool.

5.5.10 ISP Procedure

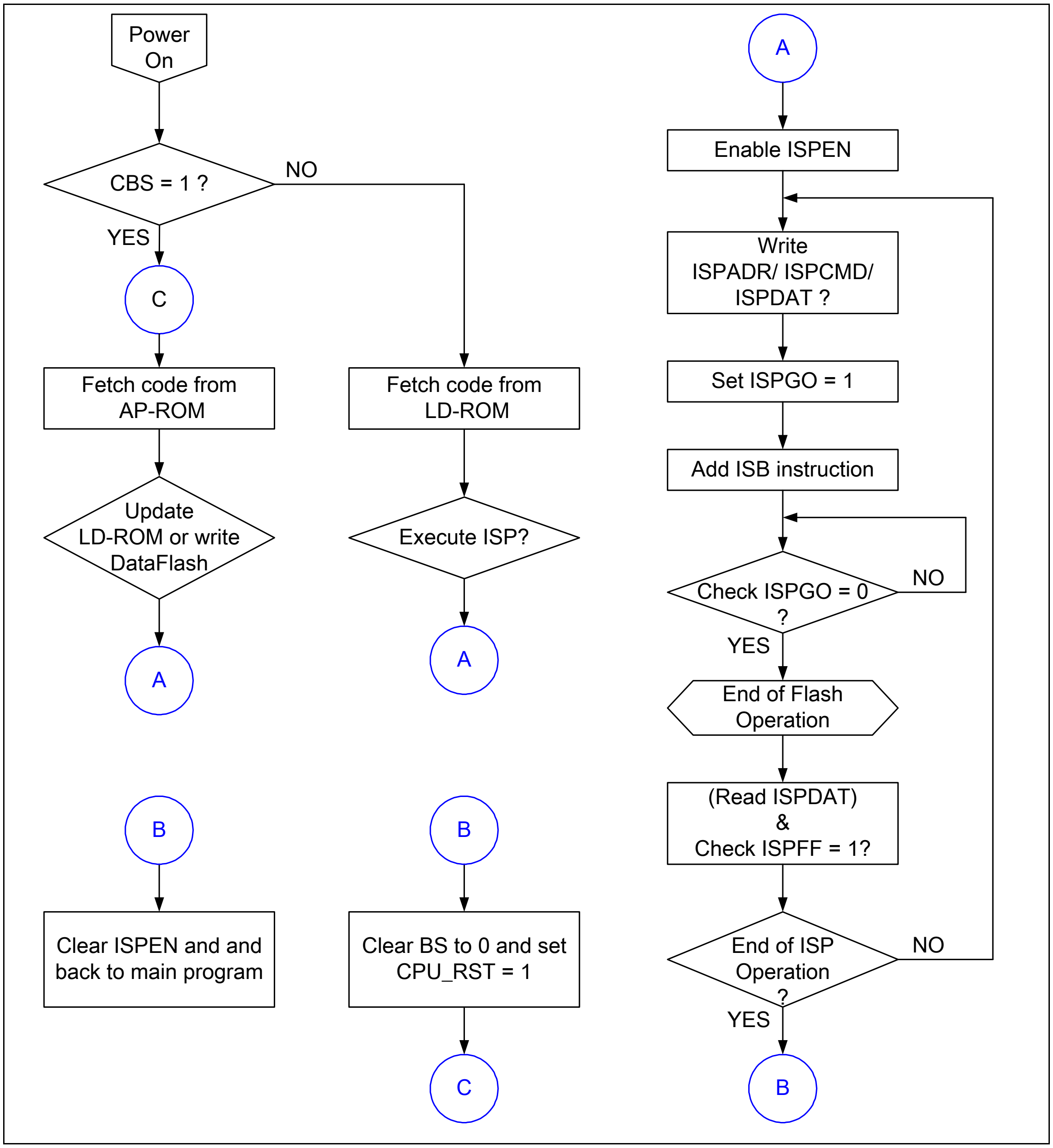
The NuMicro™ NUC123 supports booting from APROM or LDROM initially defined by user configuration bits (CBS). If user wants to update application program in APROM, he can write BS=1 and starts software reset to make chip boot from LDROM. The first step to start ISP function is write ISPEN bit to 1. S/W 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™ NUC123 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 CPU_RST = 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]	0xFFFF_FFFF
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]	0xFFFF_FFFF
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]
Read Unique ID	0	0	0100	0	0	Address in A[19:0] = 0x00_0000 0x00_0004 0x00_0008	Data out D[31:0]

Table 5-8 ISP Mode Command

5.5.11 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 (DFVSEN = 1)	0x0001_F000
DFBADR	FMC_BA+0x14	R	Data Flash Start Address (DFVSEN = 0)	CONFIG1
FATCON	FMC_BA+0x18	R/W	Flash Access Window Control Register	0x0000_0000

5.5.12 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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**ISPFF**	**LDUEN**	**CFGUEN**	**APUEN**	**Reserved**	**BS**	**ISPEN**

Bits	Description	
[31:7]	**Reserved**	Reserved
[6]	**ISPFF**	**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 if LDUEN is set to 0. (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.
[5]	**LDUEN**	**LDROM Update Enable (Write-protection Bit)** LDROM update enable bit. 1 = LDROM can be updated. 0 = LDROM cannot be updated.
[4]	**CFGUEN**	**Enable Config-bits Update by ISP (Write-protection Bit)** 1 = ISP Enabled to update config-bits. 0 = ISP Disabled to update config-bits.
[3]	**APUEN**	**APROM Update Enable (Write-protection Bit)** 1 = APROM can be updated when the chip runs in APROM. 0 = APROM cannot be updated when the chip runs in APROM.
[2]	**Reserved**	Reserved

[1]	**BS**	**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.
[0]	**ISPEN**	**ISP Enable (Write-protection Bit)** ISP function enable bit. Set this bit to enable ISP function. 1 = ISP function Enabled. 0 = ISP function Disabled.

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	Description	
[31:0]	**ISPADR**	**ISP Address** The NuMicro™ NUC123 Series is equipped with an embedded flash, and 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	Description	
[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">Description</th></tr>
<tr><td>[31:6]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[5]</td><td><b>FOEN</b></td><td rowspan="3"><b>ISP Command</b>
ISP command table is shown 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>
<tr><td>Read UID</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td></tr>
</table></td></tr>
<tr><td>[4]</td><td><b>FCEN</b></td></tr>
<tr><td>[3:0]</td><td><b>FCTRL</b></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	Description	
[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 progressed. 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 CONFIG1

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	Description	
[31:0]	**DFBADR**	**Data Flash Base Address** This register indicates data flash start address. It is a read only register. When DFVSEN is set to 0, the data flash is shared with APROM. The data flash size is defined by user configuration and the content of this register is loaded from Config1. When DFVSEN is set to 1, the data flash size is fixed as 4K and the start address can be read from this register 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	**MFOM**	**Reserved**	**LFOM**	**Reserved**			

Bits	Description	
[31:7]	**Reserved**	Reserved
[6]	**MFOM**	**Middle Frequency Optimization Mode (Write-protection Bit)** When chip operation frequency is lower than 50 MHz, chip can work more efficiently by setting this bit to 1 1 = Middle Frequency Optimization mode Enabled. 0 = Middle Frequency Optimization mode Disabled.
[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 = Low Frequency Optimization mode Enabled. 0 = Low Frequency Optimization mode Disabled.
[3:0]	**Reserved**	Reserved

5.6 USB Device Controller (USB)

5.6.1 Overview

There is one set of USB 2.0 full-speed device controller and transceiver in this device. It is compliant with USB 2.0 full-speed device specification and supports control/bulk/interrupt/ isochronous transfer types.

In this device controller, there are two main interfaces: APB bus and USB bus which comes from the USB PHY transceiver. For the APB bus, the CPU can program control registers through it. There are 512 bytes internal SRAM as data buffer in this controller. For IN or OUT transfer, it is necessary to write data to SRAM or read data from SRAM through the APB interface or SIE. User needs to set the effective starting address of SRAM for each endpoint buffer through "buffer segmentation register (BUFSEGx)".

There are 8 endpoints in this controller. Each of the endpoint can be configured as IN or OUT endpoint. All the operations including Control, Bulk, Interrupt and Isochronous transfer are implemented in this block. The block of ENDPOINT CONTROL is also used to manage the data sequential synchronization, endpoint 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, such as IN ACK, OUT ACK, and BUS events, such as suspend and resume, etc. Any event will cause an interrupt, and user just needs 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.

Also refer to *Universal Serial Bus Specification Revision 1.1*.

5.6.2 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
- Provides 1 interrupt vector with 4 different interrupt events (WAKEUP, FLDET, USB and BUS)
- Supports Control/Bulk/Interrupt/Isochronous transfer type
- Supports suspend function when no bus activity existing for 3 ms
- Provides 8 endpoints for configurable Control/Bulk/Interrupt/Isochronous transfer types and maximum 512 bytes buffer size
- Provides remote wake-up capability

5.6.3 Block Diagram

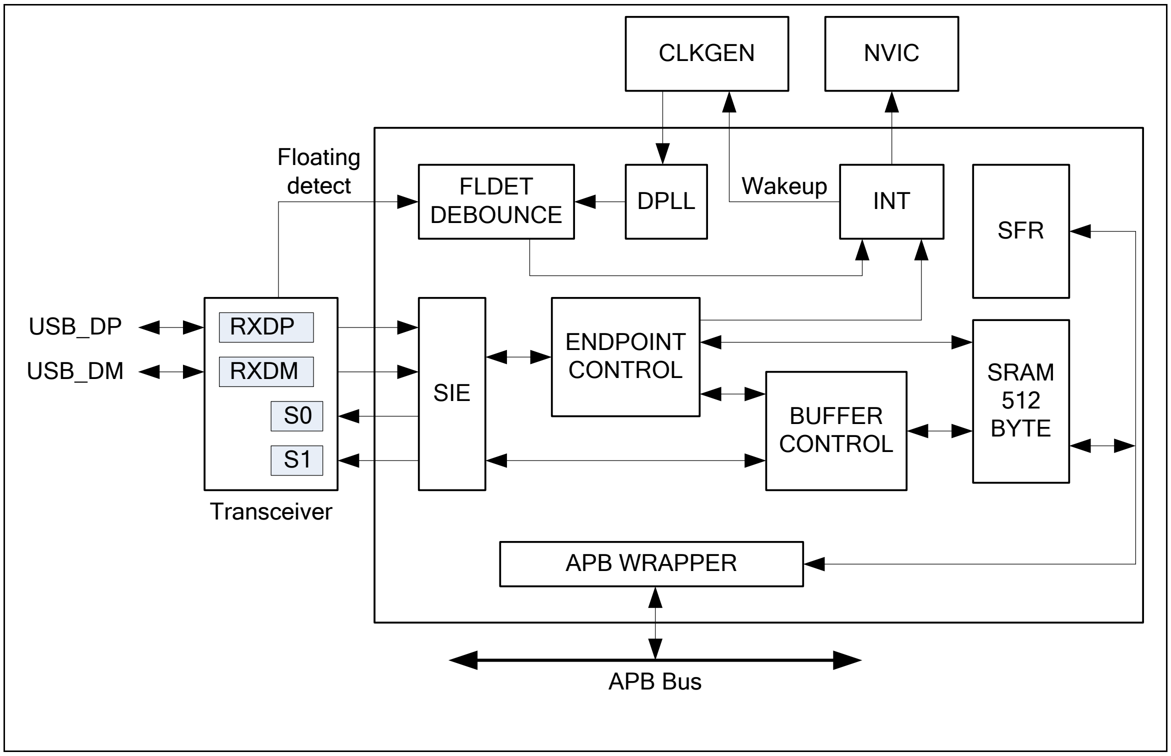


Figure 5-15 USB Block Diagram

5.6.4 Functional Description

5.6.4.1 *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

5.6.4.2 *Endpoint Control*

There are 8 endpoints in this controller. There are 2 different configuration address for endpoint 0~5 (see the register mapping table in section 5.6.5). 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.

5.6.4.3 *Digital Phase Lock Loop*

The bit rate of USB data is 12 MHz. The DPLL uses the 48 MHz which comes from the clock controller to lock the input data RXDP and RXDM. The 12 MHz bit rate clock is also converted from DPLL.

5.6.4.4 *Floating De-bounce*

A USB device may be plugged-in or plugged-out from the USB host. To monitor the state of a USB device when it is detached from the USB host, the device controller provides hardware de-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. 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 user polls this flag to check USB state, he/she must add software de-bounce if necessary.

5.6.4.5 *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 user of some USB requests, like IN ACK, OUT ACK etc., and the BUS event notifies user 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 intentional, 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.

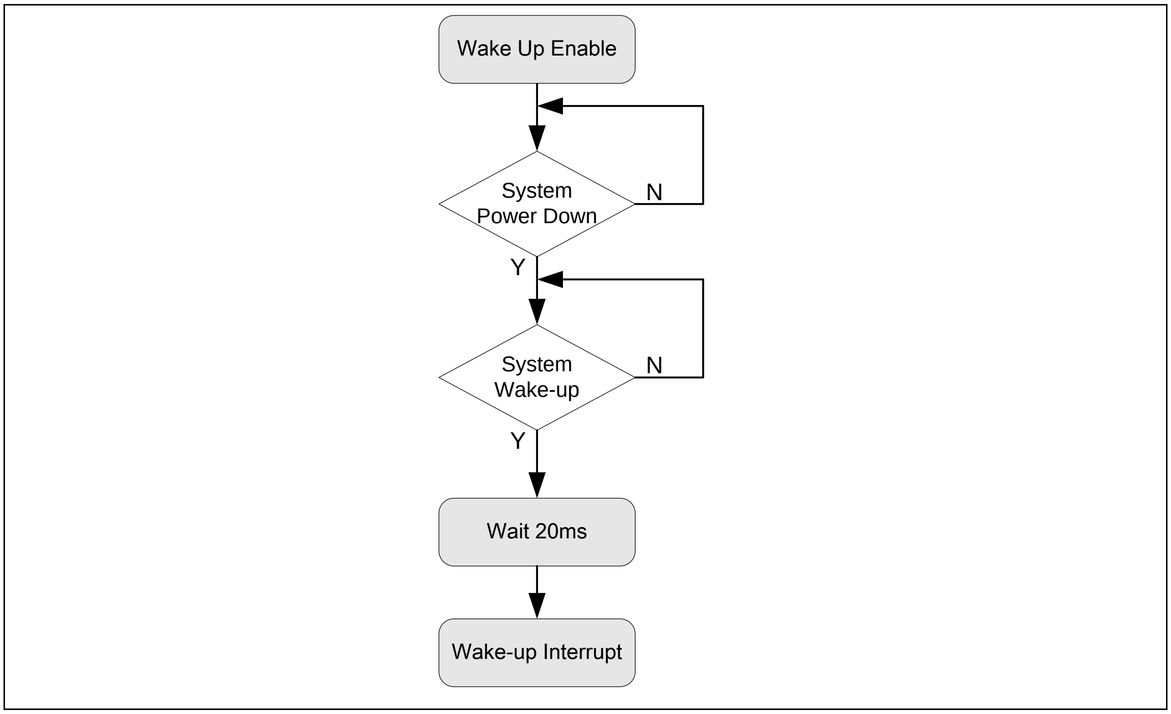


Figure 5-16 Wake-up Interrupt Operation Flow

USB interrupt is used to notify user 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 user of some bus events, such as USB reset, suspend, time-out, and resume. User can read USB_ATTR to acknowledge bus events.

5.6.4.6 *Power Saving*

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

5.6.4.7 *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 5-17 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).

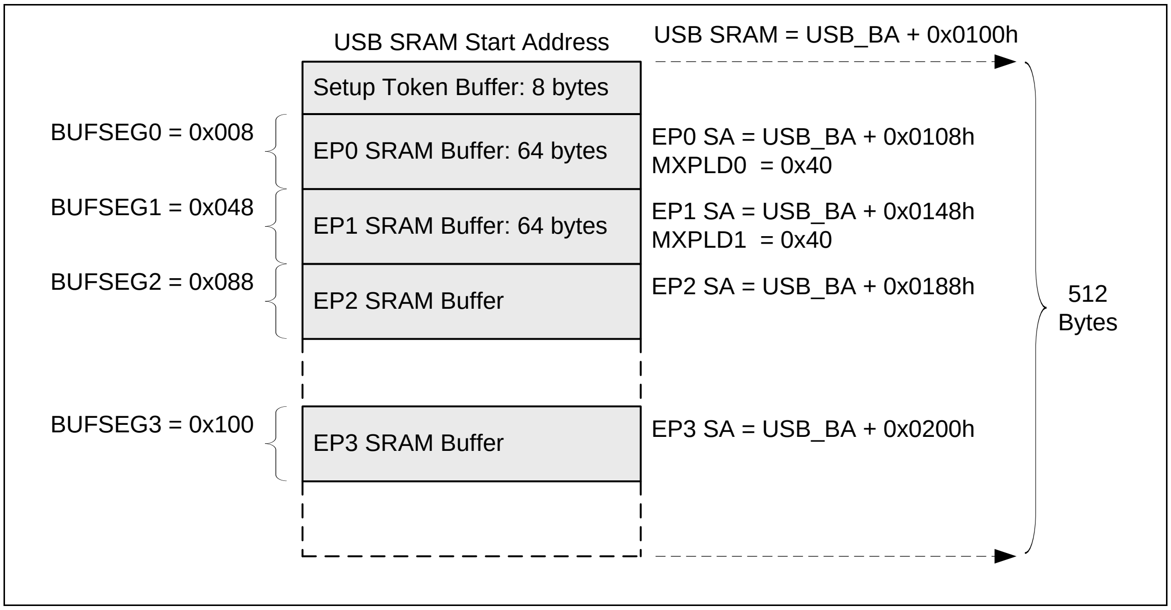

USB SRAM Start Address
USB SRAM = USB_BA + 0x0100h
Setup Token Buffer: 8 bytes
BUFSEG0 = 0x008
EP0 SRAM Buffer: 64 bytes
EP0 SA = USB_BA + 0x0108h
MXPLD0 = 0x40
BUFSEG1 = 0x048
EP1 SRAM Buffer: 64 bytes
EP1 SA = USB_BA + 0x0148h
MXPLD1 = 0x40
BUFSEG2 = 0x088
EP2 SRAM Buffer
EP2 SA = USB_BA + 0x0188h
512 Bytes
BUFSEG3 = 0x100
EP3 SRAM Buffer
EP3 SA = USB_BA + 0x0200h

Figure 5-17 Endpoint SRAM Structure

5.6.4.8 *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, user needs to prepare related data into the specified endpoint buffer in advance. After buffering the required data, user needs 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.

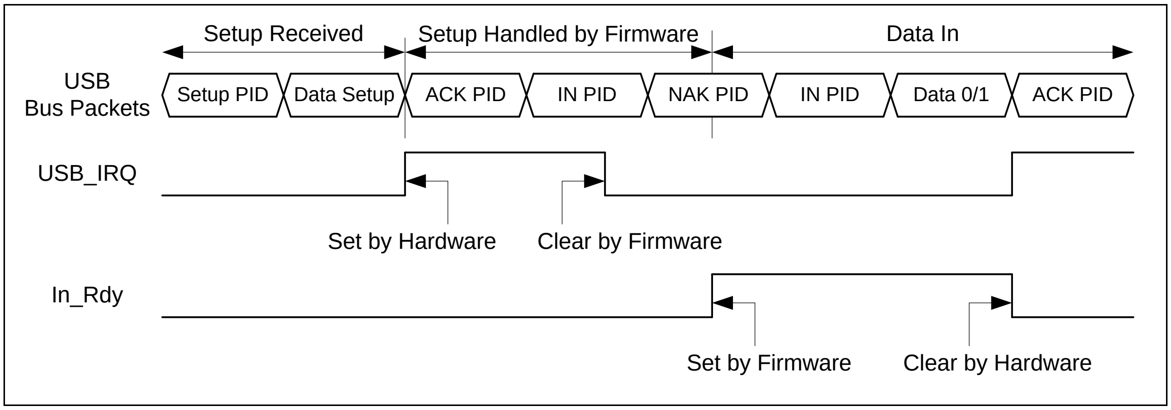


Figure 5-18 Setup Transaction Followed by Data in Transaction

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 user moves out current data in the related endpoint buffer. Once user has processed this transaction, the related register “MAXPLD” needs to be written by firmware to assert the signal “Out_Rdy” again to accept next transaction.

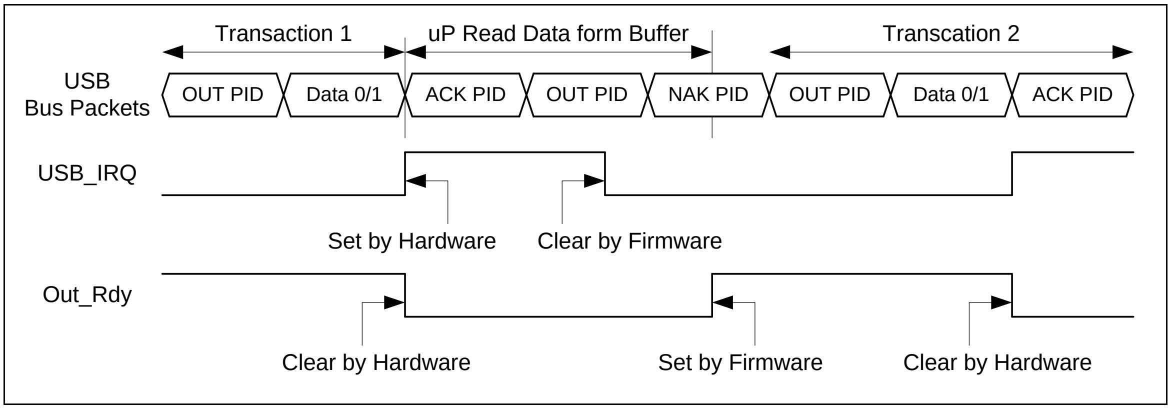


Figure 5-19 Data Out Transfer

5.6.5 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
USB_PDMA	USB_BA+0x0A4	R/W	USB PDMA Control Register	0x0000_0000
USB_BUFSEG0	USB_BA+0x500	R/W	Endpoint 0 Buffer Segmentation Register	0x0000_0000
USB_MXPLD0	USB_BA+0x504	R/W	Endpoint 0 Maximal Payload Register	0x0000_0000
USB_CFG0	USB_BA+0x508	R/W	Endpoint 0 Configuration Register	0x0000_0000
USB_CFGP0	USB_BA+0x50C	R/W	Endpoint 0 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_BUFSEG1	USB_BA+0x510	R/W	Endpoint 1 Buffer Segmentation Register	0x0000_0000
USB_MXPLD1	USB_BA+0x514	R/W	Endpoint 1 Maximal Payload Register	0x0000_0000
USB_CFG1	USB_BA+0x518	R/W	Endpoint 1 Configuration Register	0x0000_0000
USB_CFGP1	USB_BA+0x51C	R/W	Endpoint 1 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_BUFSEG2	USB_BA+0x520	R/W	Endpoint 2 Buffer Segmentation Register	0x0000_0000
USB_MXPLD2	USB_BA+0x524	R/W	Endpoint 2 Maximal Payload Register	0x0000_0000
USB_CFG2	USB_BA+0x528	R/W	Endpoint 2 Configuration Register	0x0000_0000
USB_CFGP2	USB_BA+0x52C	R/W	Endpoint 2 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_BUFSEG3	USB_BA+0x530	R/W	Endpoint 3 Buffer Segmentation Register	0x0000_0000
USB_MXPLD3	USB_BA+0x534	R/W	Endpoint 3 Maximal Payload Register	0x0000_0000
USB_CFG3	USB_BA+0x538	R/W	Endpoint 3 Configuration Register	0x0000_0000
USB_CFGP3	USB_BA+0x53C	R/W	Endpoint 3 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_BUFSEG4	USB_BA+0x540	R/W	Endpoint 4 Buffer Segmentation Register	0x0000_0000
USB_MXPLD4	USB_BA+0x544	R/W	Endpoint 4 Maximal Payload Register	0x0000_0000
USB_CFG4	USB_BA+0x548	R/W	Endpoint 4 Configuration Register	0x0000_0000
USB_CFGP4	USB_BA+0x54C	R/W	Endpoint 4 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_BUFSEG5	USB_BA+0x550	R/W	Endpoint 5 Buffer Segmentation Register	0x0000_0000

USB_MXPLD5	USB_BA+0x554	R/W	Endpoint 5 Maximal Payload Register	0x0000_0000
USB_CFG5	USB_BA+0x558	R/W	Endpoint 5 Configuration Register	0x0000_0000
USB_CFGP5	USB_BA+0x55C	R/W	Endpoint 5 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_BUFSEG6	USB_BA+0x560	R/W	Endpoint 6 Buffer Segmentation Register	0x0000_0000
USB_MXPLD6	USB_BA+0x564	R/W	Endpoint 6 Maximal Payload Register	0x0000_0000
USB_CFG6	USB_BA+0x568	R/W	Endpoint 6 Configuration Register	0x0000_0000
USB_CFGP6	USB_BA+0x56C	R/W	Endpoint 6 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_BUFSEG7	USB_BA+0x570	R/W	Endpoint 7 Buffer Segmentation Register	0x0000_0000
USB_MXPLD7	USB_BA+0x574	R/W	Endpoint 7 Maximal Payload Register	0x0000_0000
USB_CFG7	USB_BA+0x578	R/W	Endpoint 7 Configuration Register	0x0000_0000
USB_CFGP7	USB_BA+0x57C	R/W	Endpoint 7 Set Stall and Clear In/Out Ready Control Register	0x0000_0000

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.

5.6.6 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	Description	
[31:16]	**Reserved**	Reserved
[15]	**INNAK_EN**	**Active NAK Function and its Status in IN Token** 1 = 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 enables the interrupt event when the device responds NAK after receiving IN token 0 = NAK status wasn't updated into the endpoint status register when it was set to 0. It also disables the interrupt event when device responds NAK after receiving IN token
[14:9]	**Reserved**	Reserved
[8]	**WAKEUP_EN**	**Wake-up Function Enable** 1 = USB wake-up function Enabled. 0 = USB wake-up function Disabled.
[7:4]	**Reserved**	Reserved
[3]	**WAKEUP_IE**	**USB Wake-up Interrupt Enable** 1 = Wake-up Interrupt Enabled. 0 = Wake-up Interrupt Disabled.
[2]	**FLDET_IE**	**Floating Detected Interrupt Enable** 1 = Floating detect Interrupt Enabled. 0 = Floating detect Interrupt Disabled.

[1]	**USB_IE**	**USB Event Interrupt Enable** 1 = USB event interrupt Enabled. 0 = USB event interrupt Disabled.
[0]	**BUS_IE**	**Bus Event Interrupt Enable** 1 = BUS event interrupt Enabled. 0 = BUS event interrupt Disabled.

USB Interrupt Event Status Register (USB_INTSTS)

This register is USB Interrupt Event Status register; clear by writing '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**
EPEVT7	EPEVT6	EPEVT5	EPEVT4	EPEVT3	EPEVT2	EPEVT1	EPEVT0
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				WAKEUP_STS	FLDET_STS	USB_STS	BUS_STS

Bits	Description	
[31]	**SETUP**	**Setup Event Status** 1 = Setup event occurred, and cleared by writing 1 to USB_INTSTS[31]. 0 = No Setup event.
[30:24]	**Reserved**	Reserved
[23]	**EPEVT7**	**Endpoint 7's USB Event Status** 1 = USB event occurred on Endpoint 7, check USB_EPSTS[31:29] to know which kind of USB event was occurred, cleared by writing 1 to USB_INTSTS[23] or USB_INTSTS[1]. 0 = No event occurred in endpoint 7.
[22]	**EPEVT6**	**Endpoint 6's USB Event Status** 1 = USB event occurred on Endpoint 6, check USB_EPSTS[28:26] to know which kind of USB event was occurred, cleared by writing 1 to USB_INTSTS[22] or USB_INTSTS[1] 0 = No event occurred on Endpoint 6.
[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 writing 1 to USB_INTSTS[21] or USB_INTSTS[1] 0 = No event occurred on 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 writing 1 to USB_INTSTS[20] or USB_INTSTS[1] 0 = No event occurred on 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 writing 1 to USB_INTSTS[19] or USB_INTSTS[1] 0 = No event occurred on 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 writing 1 to USB_INTSTS[18] or USB_INTSTS[1] 0 = No event occurred on 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 writing 1 to USB_INTSTS[17] or USB_INTSTS[1] 0 = No event occurred on 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 writing 1 to USB_INTSTS[16] or USB_INTSTS[1] 0 = No event occurred on Endpoint 0.
[15:4]	**Reserved**	Reserved
[3]	**WAKEUP_STS**	**Wake-up Interrupt Status** 1 = Wake-up event occurred, cleared by writing 1 to USB_INTSTS[3] 0 = No Wake-up event occurred.
[2]	**FLDET_STS**	**Floating Detected Interrupt Status** 1 = There is attached/detached event in the USB bus and it is cleared by writing 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 writing 1 to USB_INTSTS[1] or EPSTS0~5 and SETUP (USB_INTSTS[31]) 0 = No USB event 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 writing 1 to USB_INTSTS[0]. 0 = No BUS event 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	Description	
[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
EPSTS7[2:1]			EPSTS6[2:1]			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	Description	
[31:29]	**EPSTS7**	**Endpoint 7 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
[28:26]	**EPSTS6**	**Endpoint 6 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

[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 = Out data is more than the Max Payload in MXPLD register or the Setup data is 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	TIME-OUT	RESUME	SUSPEND	USBRST

Bits	Description	
[31:11]	**Reserved**	Reserved
[10]	**BYTEM**	**CPU access USB SRAM Size Mode Selection** 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 = Pull-up resistor in USB_DP bus Enabled. 0 = the pull-up resistor in USB_DP bus Disabled.
[7]	**USB_EN**	**USB Controller Enable** 1 = USB Controller Enabled. 0 = USB Controller Disabled.
[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 = PHY transceiver function Enabled. 0 = PHY transceiver function Disabled.

[3]	**TIME-OUT**	**Time Out Status** 1 = No Bus 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 = No bus suspend. It is a read only bit.
[0]	**USBRST**	**USB Reset Status** 1 = Bus reset when SE0 (single-ended 0) more than 2.5 us. 0 = No bus 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	Description	
[31:1]	**Reserved**	Reserved
[0]	**FLDET**	**Device Floating Detected** 1 = Controller is attached to the BUS. 0 = Controller isn’t attached to 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	Description	
[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 = 0~7

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
USB_BUFSEG0	USB_BA+0x500	R/W	Endpoint 0 Buffer Segmentation Register	0x0000_0000
USB_BUFSEG1	USB_BA+0x510	R/W	Endpoint 1 Buffer Segmentation Register	0x0000_0000
USB_BUFSEG2	USB_BA+0x520	R/W	Endpoint 2 Buffer Segmentation Register	0x0000_0000
USB_BUFSEG3	USB_BA+0x530	R/W	Endpoint 3 Buffer Segmentation Register	0x0000_0000
USB_BUFSEG4	USB_BA+0x540	R/W	Endpoint 4 Buffer Segmentation Register	0x0000_0000
USB_BUFSEG5	USB_BA+0x550	R/W	Endpoint 5 Buffer Segmentation Register	0x0000_0000
USB_BUFSEG6	USB_BA+0x560	R/W	Endpoint 6 Buffer Segmentation Register	0x0000_0000
USB_BUFSEG7	USB_BA+0x570	R/W	Endpoint 7 Buffer Segmentation Register	0x0000_0000

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

Bits	Description	
[31:9]	**Reserved**	Reserved
[8:3]	**BUFSEGx**	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.
[2:0]	**Reserved**	Reserved

Maximal Payload Register (USB_MXPLDx) x = 0~7

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
USB_MXPLD0	USB_BA+0x504	R/W	Endpoint 0 Maximal Payload Register	0x0000_0000
USB_MXPLD1	USB_BA+0x514	R/W	Endpoint 1 Maximal Payload Register	0x0000_0000
USB_MXPLD2	USB_BA+0x524	R/W	Endpoint 2 Maximal Payload Register	0x0000_0000
USB_MXPLD3	USB_BA+0x534	R/W	Endpoint 3 Maximal Payload Register	0x0000_0000
USB_MXPLD4	USB_BA+0x544	R/W	Endpoint 4 Maximal Payload Register	0x0000_0000
USB_MXPLD5	USB_BA+0x554	R/W	Endpoint 5 Maximal Payload Register	0x0000_0000
USB_MXPLD6	USB_BA+0x564	R/W	Endpoint 6 Maximal Payload Register	0x0000_0000
USB_MXPLD7	USB_BA+0x574	R/W	Endpoint 7 Maximal Payload Register	0x0000_0000

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

Bits	Description	
[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 is 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 indicates the data length be transmitted to host For OUT token, the value of MXPLD indicates the actual data length receiving from host. **Note:** Once MXPLD is written, the data packets will be transmitted/received immediately after IN/OUT token arrived.

Configuration Register (USB_CFGx) x = 0~7

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
USB_CFG0	USB_BA+0x508	R/W	Endpoint 0's Configuration Register	0x0000_0000
USB_CFG1	USB_BA+0x518	R/W	Endpoint 1's Configuration Register	0x0000_0000
USB_CFG2	USB_BA+0x528	R/W	Endpoint 2's Configuration Register	0x0000_0000
USB_CFG3	USB_BA+0x538	R/W	Endpoint 3's Configuration Register	0x0000_0000
USB_CFG4	USB_BA+0x548	R/W	Endpoint 4's Configuration Register	0x0000_0000
USB_CFG5	USB_BA+0x558	R/W	Endpoint 5's Configuration Register	0x0000_0000
USB_CFG6	USB_BA+0x568	R/W	Endpoint 6's Configuration Register	0x0000_0000
USB_CFG7	USB_BA+0x578	R/W	Endpoint 7'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	Description	
[31:10]	**Reserved**	Reserved
[9]	**CSTALL**	**Clear STALL Response** 1 = Clear the device to respond STALL handshake in the setup stage. 0 = Device Disabled to clear the STALL handshake in the 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 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 = 0~7

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
USB_CFGP0	USB_BA+0x50C	R/W	Endpoint 0 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_CFGP1	USB_BA+0x51C	R/W	Endpoint 1 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_CFGP2	USB_BA+0x52C	R/W	Endpoint 2 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_CFGP3	USB_BA+0x53C	R/W	Endpoint 3 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_CFGP4	USB_BA+0x54C	R/W	Endpoint 4 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_CFGP5	USB_BA+0x55C	R/W	Endpoint 5 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_CFGP6	USB_BA+0x56C	R/W	Endpoint 6 Set Stall and Clear In/Out Ready Control Register	0x0000_0000
USB_CFGP7	USB_BA+0x57C	R/W	Endpoint 7 Set Stall and Clear In/Out Ready 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="6">Reserved</td><td>SSTALL</td><td>CLRRDY</td></tr>
</table>

Bits	Description	
[31:2]	**Reserved**	Reserved
[1]	**SSTALL**	**Set STALL** 1 = Set the device to respond STALL automatically. 0 = Device Disabled to respond 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, user can set this bit to 1 to turn it off and it is automatically cleared to 0. For IN token, writing '1' is used to clear the IN token had ready to transmit the data to USB. For OUT token, writing '1' is used to clear the OUT token had ready to receive the data from USB. This bit is written 1 only and 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	Description	
[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

USB PDMA Controller Register (USB_PDMA)

Register	Offset	R/W	Description	Reset Value
USB_PDMA	USB_BA+0x0A4	R/W	USB PDMA Control Regiseter	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						PDMA_EN	PDMA_RW

Bits	Description	
[31:2]	**Reserved**	Reserved
[1]	**PDMA_EN**	**PDMA Function Enable** 1 = The PDMA function in USB is active 0 = The PDMA function is not active This bit will be automatically cleared after PDMA transfer done.
[0]	**PDMA_RW**	**PDMA_RW** 1 = The PDMA will read data from USB buffer to memory 0 = The PDMA will read data from memory to USB buffer

5.7 General Purpose I/O (GPIO)

5.7.1 Overview

The NuMicro™ NUC123 series has up to 47 General Purpose I/O pins shared with other function pins depending on the chip configuration. These 47 pins are arranged in 5 ports named GPIOA, GPIOB, GPIOC, GPIOD and GPIOF. GPIOA has 6 pins on PA[15:10]. GPIOB has 15 pins on PB[15:12] and PB[10:0]. GPIOC has 12 pins on PC[13:8] and PC[5:0]. GPIOD has 10 pins on PD[11:8] and PD[5:0]. GPIOF has 4 pins on PF[3:0]. Each one of the 47 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 has a very weakly individual pull-up resistor which is about 110 KΩ~300 KΩ for V_{DD} is from 5.0 V to 2.5 V.

5.7.2 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
- Supports High Driver and High Sink IO mode

5.7.3 Functional Description

5.7.3.1 *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.

5.7.3.2 *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.

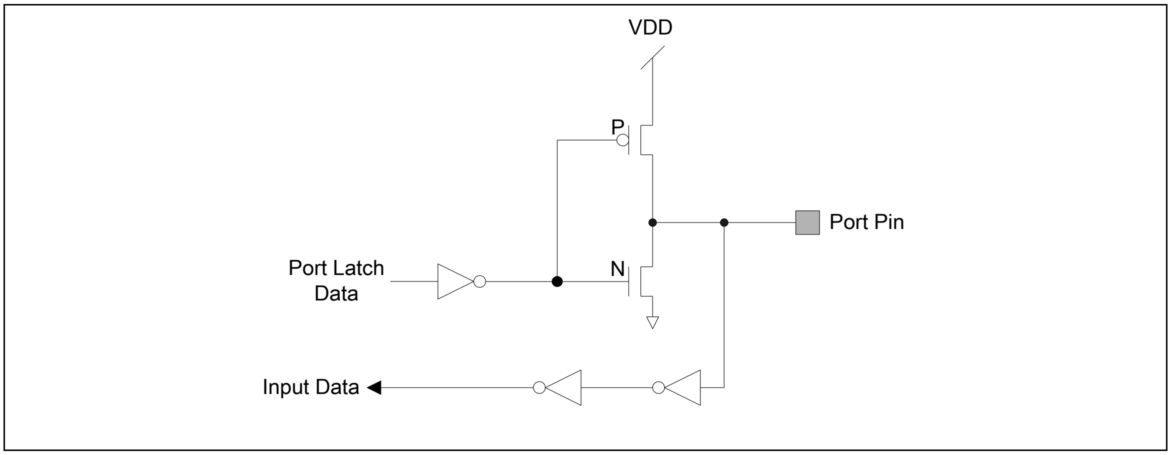


Figure 5-20 Push-Pull Output

5.7.3.3 *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.

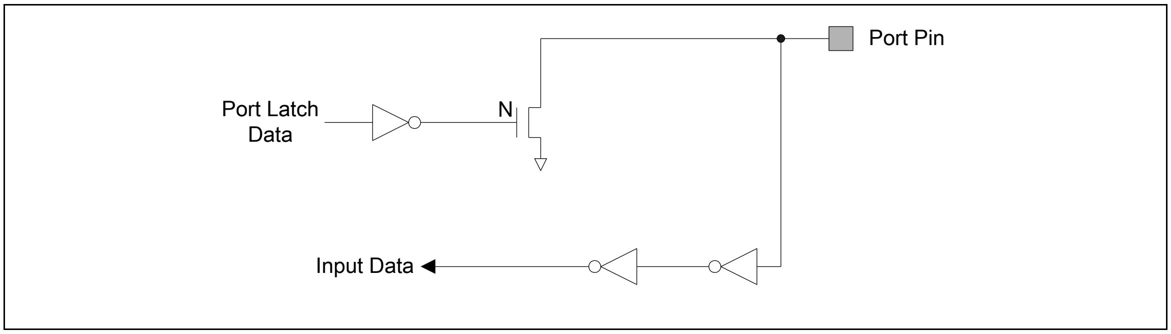


Figure 5-21 Open-Drain Output

5.7.3.4 *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, the 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.

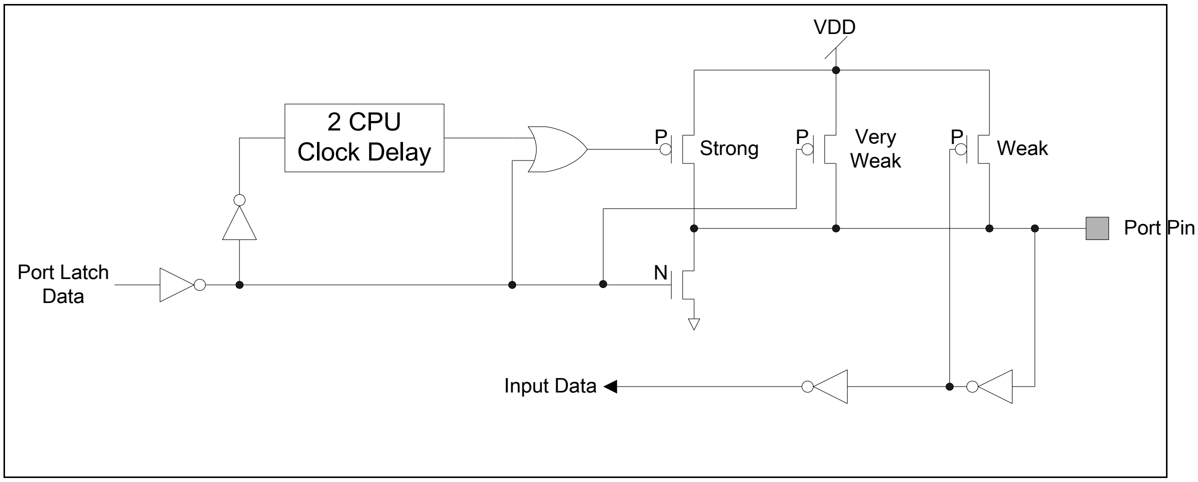


Figure 5-22 Quasi-bidirectional I/O Mode

5.7.4 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	0x0000_0000
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	0x0000_0000
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	0x0000_0000
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	0x0000_0000
GPIOF_PMD	GP_BA+0x140	R/W	GPIO Port F Pin I/O Mode Control	0x0000_00FF
GPIOF_OFFD	GP_BA+0x144	R/W	GPIO Port F Pin OFF Digital Enable	0x0000_0000
GPIOF_DOUT	GP_BA+0x148	R/W	GPIO Port F Data Output Value	0x0000_000F
GPIOF_DMASK	GP_BA+0x14C	R/W	GPIO Port F Data Output Write Mask	0x0000_0000
GPIOF_PIN	GP_BA+0x150	R	GPIO Port F Pin Value	0x0000_000X
GPIOF_DBEN	GP_BA+0x154	R/W	GPIO Port F De-bounce Enable	0x0000_0000
GPIOF_IMD	GP_BA+0x158	R/W	GPIO Port F Interrupt Mode Control	0x0000_0000
GPIOF_IEN	GP_BA+0x15C	R/W	GPIO Port F Interrupt Enable	0x0000_0000
GPIOF_ISRC	GP_BA+0x160	R/W	GPIO Port F Interrupt Source Flag	0x0000_0000
DBNCECON	GP_BA+0x180	R/W	De-bounce Cycle Control	0x0000_0020
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

Register	Offset	R/W	Description	Reset Value
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
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
GPIOC5_DOUT	GP_BA+0x294	R/W	GPIO PC.5 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
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

Register	Offset	R/W	Description	Reset Value
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
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
GPIOF0_DOUT	GP_BA+0x340	R/W	GPIO PF.0 Bit Output/Input Control	0x0000_0001
GPIOF1_DOUT	GP_BA+0x344	R/W	GPIO PF.1 Bit Output/Input Control	0x0000_0001
GPIOF2_DOUT	GP_BA+0x348	R/W	GPIO PF.2 Bit Output/Input Control	0x0000_0001
GPIOF3_DOUT	GP_BA+0x34C	R/W	GPIO PF.3 Bit Output/Input Control	0x0000_0001

5.7.5 Register Description

GPIO Port [A/B/C/D/F] 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
GPIOF_PMD	GP_BA+0x140	R/W	GPIO Port F Pin I/O Mode Control	0x0000_00FF

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	Description	
[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 Note: GPIOA: valid n are 15~10. Others are reserved. GPIOB: valid n are 15~12, 10~0. Others are reserved. GPIOC: valid n are 13~8, 5~0. Others are reserved. GPIOD: valid n are 11~8, 5~0. Others are reserved. GPIOF: valid n are 3~0. Others are reserved.

GPIO Port [A/B/C/D/F] 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
GPIOF_OFFD	GP_BA+0x144	R/W	GPIO Port F 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	Description	
[n+16]	**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, user can OFF digital input path to avoid creepage 1 = IO digital input path Disabled (digital input tied to low). 0 = IO digital input path Enabled. **Notes:** GPIOA: valid n are 15~10. Others are reserved. GPIOB: valid n are 15~12, 10~0. Others are reserved. GPIOC: valid n are 13~8, 5~0. Others are reserved. GPIOD: valid n are 11~8, 5~0. Others are reserved. GPIOF: valid n are 3~0. Others are reserved.
[0:15]	**Reserved**	Reserved

GPIO Port [A/B/C/D/F] 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
GPIOF_DOUT	GP_BA+0x148	R/W	GPIO Port F Data Output Value	0x0000_000F

31	30	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	Description	
[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/DF] 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/ F] Pin[n] will drive Low if the GPIO pin is configures as output, open-drain and quasi-mode. Note: GPIOA: valid n are 15~10. Others are reserved. GPIOB: valid n are 15~12, 10~0. Others are reserved. GPIOC: valid n are 13~8, 5~0. Others are reserved. GPIOD: valid n are 11~8, 5~0. Others are reserved. GPIOF: valid n are 3~0. Others are reserved.

GPIO Port [A/B/C/D/F] 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	0x0000_0000
GPIOB_DMASK	GP_BA+0x04C	R/W	GPIO Port B Data Output Write Mask	0x0000_0000
GPIOC_DMASK	GP_BA+0x08C	R/W	GPIO Port C Data Output Write Mask	0x0000_0000
GPIOD_DMASK	GP_BA+0x0CC	R/W	GPIO Port D Data Output Write Mask	0x0000_0000
GPIOF_DMASK	GP_BA+0x14C	R/W	GPIO Port F Data Output Write Mask	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved
[n]	**DMASK[n]**	**Port [A/B/C/D/ F] 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 and GPIOFx_DOUT). **Note:** GPIOA: valid n are 15~10. Others are reserved. GPIOB: valid n are 15~12, 10~0. Others are reserved. GPIOC: valid n are 13~8, 5~0. Others are reserved. GPIOD: valid n are 11~8, 5~0. Others are reserved. GPIOF: valid n are 3~0. Others are reserved.

GPIO Port [A/B/C/D/F] 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
GPIOF_PIN	GP_BA+0x150	R	GPIO Port F Pin Value	0x0000_000X

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

Bits	Description	
[31:16]	**Reserved**	Reserved
[n]	**PIN[n]**	**Port [A/B/C/D/ F] 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 Note: GPIOA: valid n are 15~10. Others are reserved. GPIOB: valid n are 15~12, 10~0. Others are reserved. GPIOC: valid n are 13~8, 5~0. Others are reserved. GPIOD: valid n are 11~8, 5~0. Others are reserved. GPIOF: valid n are 3~0. Others are reserved.

GPIO Port [A/B/C/D/F] 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	0x0000_0000
GPIOB_DBEN	GP_BA+0x054	R/W	GPIO Port B De-bounce Enable	0x0000_0000
GPIOC_DBEN	GP_BA+0x094	R/W	GPIO Port C De-bounce Enable	0x0000_0000
GPIOD_DBEN	GP_BA+0x0D4	R/W	GPIO Port D De-bounce Enable	0x0000_0000
GPIOF_DBEN	GP_BA+0x154	R/W	GPIO Port F De-bounce 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**
DBEN[15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DBEN[7:0]							

Bits	Description	
[31:16]	**Reserved**	Reserved
[n]	**DBEN[n]**	**Port [A/B/C/D/ F] 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. Note: GPIOA: valid n are 15~10. Others are reserved. GPIOB: valid n are 15~12, 10~0. Others are reserved. GPIOC: valid n are 13~8, 5~0. Others are reserved. GPIOD: valid n are 11~8, 5~0. Others are reserved. GPIOF: valid n are 3~0. Others are reserved.

GPIO Port [A/B/C/D/F] 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	0x0000_0000
GPIOB_IMD	GP_BA+0x058	R/W	GPIO Port B Interrupt Mode Control	0x0000_0000
GPIOC_IMD	GP_BA+0x098	R/W	GPIO Port C Interrupt Mode Control	0x0000_0000
GPIOD_IMD	GP_BA+0x0D8	R/W	GPIO Port D Interrupt Mode Control	0x0000_0000
GPIOF_IMD	GP_BA+0x158	R/W	GPIO Port F Interrupt Mode Control	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved
[n]	**IMD[n]**	**Port [A/B/C/D/ F] 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 pin is set as the level trigger interrupt, 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. **Note:** GPIOA: Valid n are 15~10. Others are reserved. GPIOB: Valid n are 15~12, 10~0. Others are reserved. GPIOC: Valid n are 13~8, 5~0. Others are reserved. GPIOD: Valid n are 11~8, 5~0. Others are reserved.

		GPIOF: Valid n are 3~0. Others are reserved.

GPIO Port [A/B/C/D/F] 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
GPIOF_IEN	GP_BA+0x15C	R/W	GPIO Port F 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	Description	
[n+16]	**IR_EN[n]**	**Port [A/B/C/D/ F] Interrupt Enable by Input Rising Edge or Input Level High** IR_EN[n] is 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 the IR_EN[n] bit is set to 1: If the interrupt is level triggered, the input PIN[n] state at level “high” will generate the interrupt. If the interrupt is edge triggered, the input PIN[n] state changes from “low-to-high” will generate the interrupt. 1 = PIN[n] level-high or low-to-high interrupt Enabled. 0 = PIN[n] level-high or low-to-high interrupt Disabled. **Note:** GPIOA: Valid n are 15~10. Others are reserved. GPIOB: Valid n are 15~12, 10~0. Others are reserved. GPIOC: Valid n are 13~8, 5~0. Others are reserved. GPIOD: Valid n are 11~8, 5~0. Others are reserved. GPIOF: Valid n are 3~0. Others are reserved.

[n]	**IF_EN[n]**	**Port [A/B/C/D/ F] Interrupt Enable by Input Falling Edge or Input Level Low** IF_EN[n] is 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 triggered, the input PIN[n] state at level “low” will generate the interrupt. If the interrupt is edge triggered, the input PIN[n] state changes from “high-to-low” will generate the interrupt. 1 = PIN[n] state low-level or high-to-low change interrupt Enabled. 0 = PIN[n] state low-level or high-to-low change interrupt Disabled. **Note:** GPIOA: Valid n are 15~10. Others are reserved. GPIOB: Valid n are 15~12, 10~0. Others are reserved. GPIOC: Valid n are 13~8, 5~0. Others are reserved. GPIOD: Valid n are 11~8, 5~0. Others are reserved. GPIOF: Valid n are 3~0. Others are reserved.

GPIO Port [A/B/C/D/F] 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_XXXX
GPIOB_ISRC	GP_BA+0x060	R/W	GPIO Port B Interrupt Trigger Source Indicator	0x0000_XXXX
GPIOC_ISRC	GP_BA+0x0A0	R/W	GPIO Port C Interrupt Trigger Source Indicator	0x0000_XXXX
GPIOD_ISRC	GP_BA+0x0E0	R/W	GPIO Port D Interrupt Trigger Source Indicator	0x0000_XXXX
GPIOF_ISRC	GP_BA+0x160	R/W	GPIO Port F Interrupt Trigger Source Indicator	0x0000_000X

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

Bits	Description	
[31:16]	**Reserved**	Reserved
[n]	**ISRC[n]**	**Port [A/B/C/D/ F] Interrupt Trigger Source Indicator** Read: 1 = Indicates GPIOx[n] generate an interrupt. 0 = No interrupt at GPIOx[n]. Write : 1= Clear the corresponding pending interrupt. 0= No action. **Note:** GPIOA: Valid n are 15~10. Others are reserved. GPIOB: Valid n are 15~12, 10~0. Others are reserved. GPIOC: Valid n are 13~8, 5~0. Others are reserved. GPIOD: Valid n are 11~8, 5~0. Others are reserved. GPIOF: Valid n are 3~0. Others are reserved.

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**			

Bits	Description	
[5]	**ICLK_ON**	**Interrupt clock On mode** Setting this bit to 0 will disable the interrupt generate circuit clock if the pin[n] interrupt is disabled 1 = Interrupt generated circuit clock always Enabled. 0 = Clock Disabled if the GPIOA/B/C/D/F[n] interrupt is disabled.
[4]	**DBCLKSRC**	**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.

<table>
<tr>
<td>[3:0]</td>
<td><b>DBCLKSEL</b></td>
<td><b>De-bounce sampling cycle selection</b>
<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>
<tr><td>11</td><td>Sample interrupt input once per 8*256 clocks</td></tr>
<tr><td>12</td><td>Sample interrupt input once per 16*256 clocks</td></tr>
<tr><td>13</td><td>Sample interrupt input once per 32*256 clocks</td></tr>
<tr><td>14</td><td>Sample interrupt input once per 64*256 clocks</td></tr>
<tr><td>15</td><td>Sample interrupt input once per 128*256 clocks</td></tr>
</table>
</td>
</tr>
</table>

GPIO Port [A/B/C/D/F] 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
GPIOFx_DOUT	GP_BA+0x340 - GP_BA+0x34C	R/W	GPIO Port F 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	Description	
[0]	**GPIOxx_DOUT**	**GPIOxx I/O Pin Bit Output/Input Control** Writing this bit can control one GPIO pin output value. 1 = Set the corresponding GPIO pin to high. 0 = Set the corresponding GPIO pin to low. Read this register to get IO pin status. For example: Writing GPIOA0_DOUT will reflect the written value to bit GPIOA_DOUT[0], read GPIOA0_DOUT will return the value of GPIOA_PIN[0]

5.8 I^{2}C Serial Interface Controller (Master/Slave) (I^{2}C)

5.8.1 Overview

I^{2}C is a two-wire, bi-directional serial bus that provides a simple and efficient method of data exchange between devices. The I^{2}C 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 5-23 for more detail I^{2}C BUS Timing.

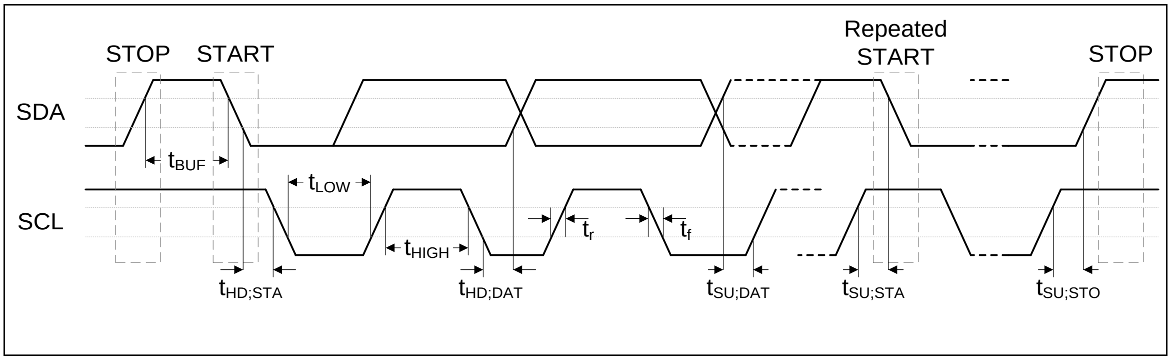


Figure 5-23 I^{2}C Bus Timing

The device's on-chip I^{2}C logic provides the serial interface that meets the I^{2}C bus standard mode specification. The I^{2}C port handles byte transfers autonomously. To enable this port, the bit ENS1 in I2CON should be set to '1'. The I^{2}C H/W interfaces to the I^{2}C bus via two pins: SDA and SCL. Pull up resistor is needed for I^{2}C operation as these are open drain pins. When the I/O pins are used as I^{2}C port, user must set the pins function to I^{2}C in advance.

5.8.2 Features

The I^{2}C 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 I²C interrupt if the I²C 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
- I²C-bus controllers supporting multiple address recognition (four slave address with mask option)
- Supports Power-down wake-up function

5.8.3 Functional Description

5.8.3.1 *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

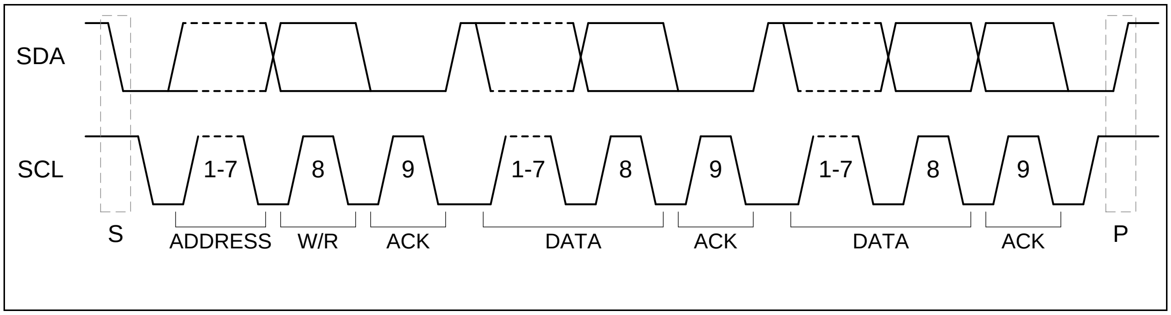


Figure 5-24 I²C Protocol

5.8.3.2 *Data transfer on the I²C-bus*

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

The transfer direction is not changed

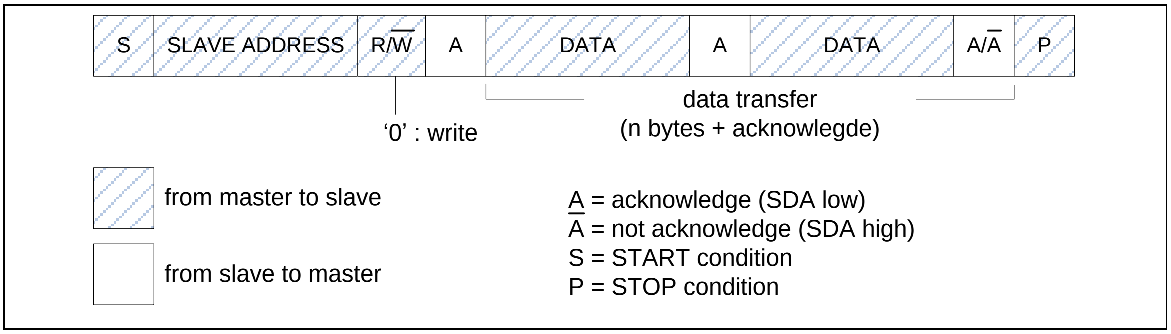


Figure 5-25 Master Transmits Data to Slave

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

The transfer direction is changed.

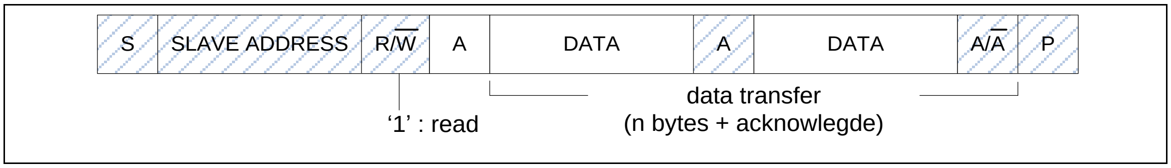


Figure 5-26 Master Reads Data from Slave

5.8.3.3 *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.

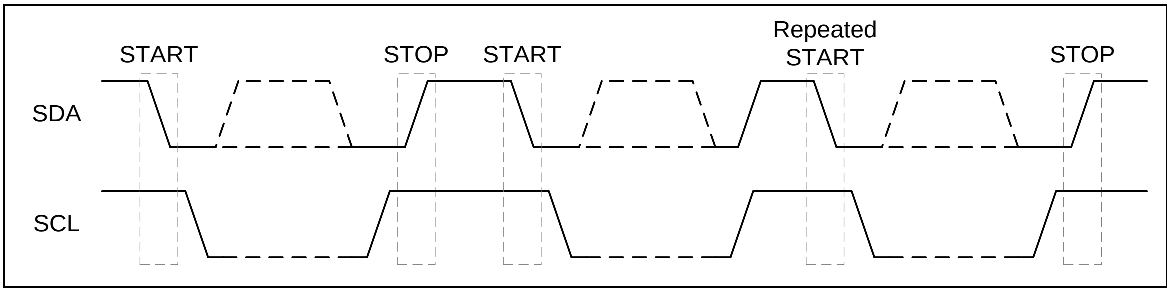


Figure 5-27 START and STOP Conditions

5.8.3.4 *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.

5.8.3.5 *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.

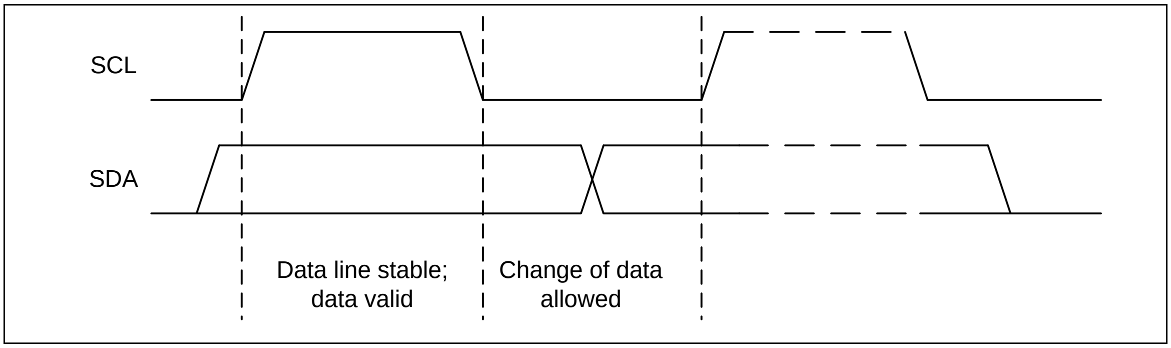


Figure 5-28 Bit Transfer on the I²C Bus

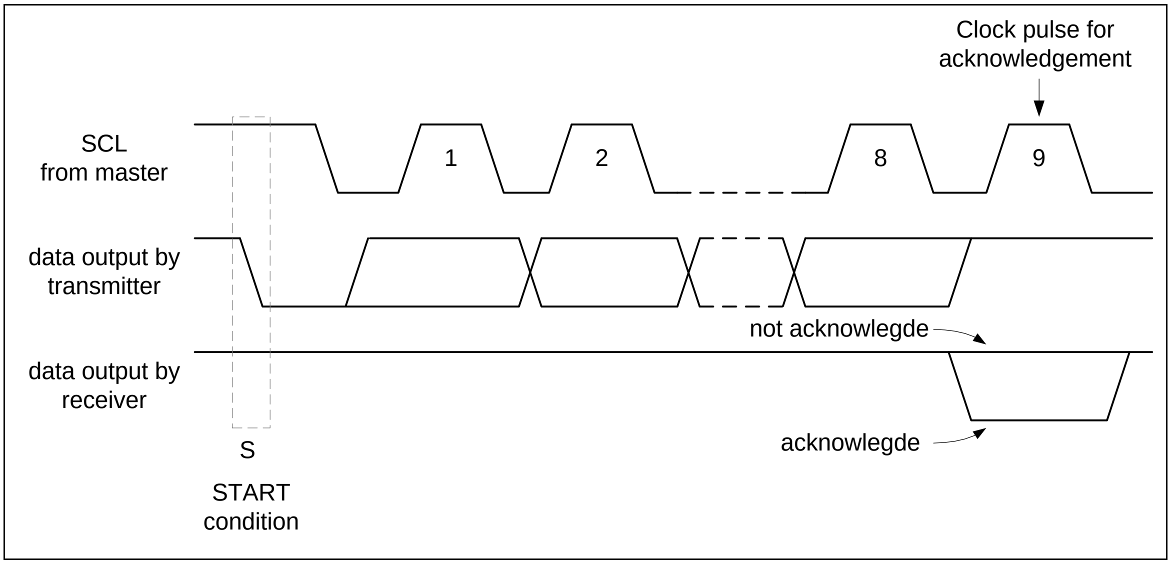


Figure 5-29 Acknowledge on the I²C Bus

5.8.4 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 I²C 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 I²C 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.

5.8.4.1 *Address Registers (I2CADDR)*

I²C port is equipped with four slave address registers I2CADDRn (n=0~3). The contents of the register are irrelevant when I²C 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 I²C hardware will react if the contents of I2CADDRn are matched with the received slave address.

The I²C ports support the "General Call" function. If the GC bit (I2CADDRn [0]) is set the I²C port hardware will respond to General Call address (00H). Clear GC bit to disable general call function.

When GC bit is set and the I²C is in Slave mode, it can receive the general call address by 00H after Master sends general call address to I²C bus, and then it will follow status of GC mode.

I²C 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.

5.8.4.2 *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 I²C 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 I²C 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.

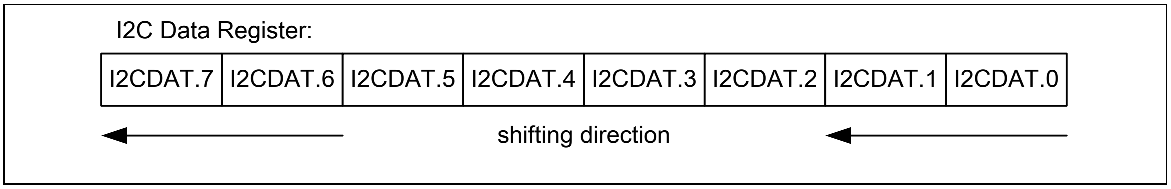


Figure 5-30 I^{2}C Data Shifting Direction

5.8.4.3 *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.

5.8.4.4 *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 cannot recognize stop condition during this action when bus error occurs.

5.8.4.5 *I^{2}C Clock Baud Rate Bits (I2CLK)*

The data baud rate of I^{2}C is determines by I2CLK [7:0] register when I^{2}C is in Master mode. It is not important when I^{2}C is in a Slave mode. In the Slave modes, I^{2}C will automatically synchronize with any clock frequency from master I^{2}C device.

The data baud rate of I^{2}C setting is Data Baud Rate of I^{2}C = (system clock) / (4x (I2CLK [7:0] +1)). If system clock = 16 MHz, the I2CLK [7:0] = 40 (28H), so data baud rate of I^{2}C = 16 MHz/ (4x (40 +1)) = 97.5 Kbits/sec.

5.8.4.6 *The I^{2}C Time-out Counter Register (I2CTOC)*

There is a 14-bit time-out counter which can be used to deal with the I^{2}C bus hang-up. If the time-out counter is enabled, the counter starts counting up until it overflows (TIF=1) and generates I^{2}C 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 counting up after SI is cleared. If I^{2}C 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^{2}C interrupt. Refer to the Figure 5-31 for the 14-bit time-out counter. User may write 1 to clear TIF to zero.

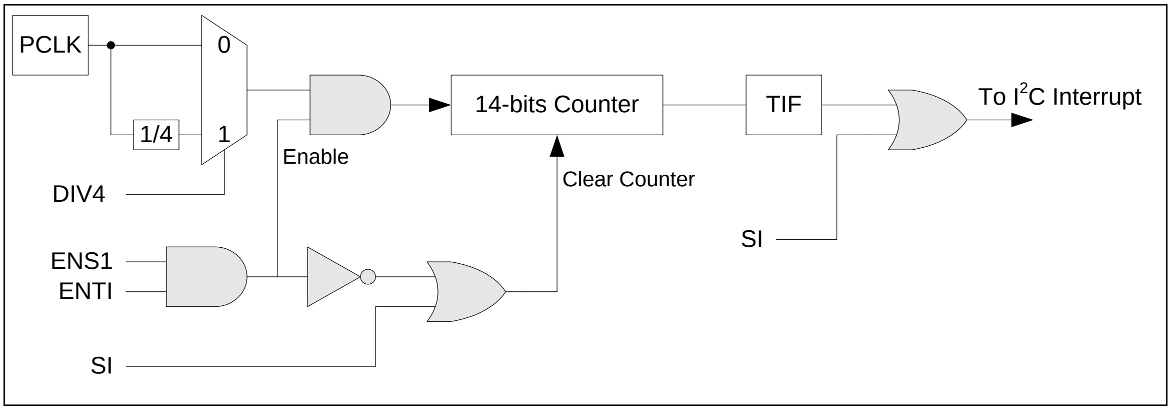


Figure 5-31 I^{2}C Time-out Count Block Diagram

5.8.4.7 *The I^{2}C wake-up control Register (**I2CWKUPCON**)*

When entering sleep mode, other I^{2}C master can wake up our chip by addressing our I^{2}C device, user must configure the related setting before entering sleep mode.

WKUPEN enables I^{2}C wake-up function

5.8.4.8 *The I^{2}C wake-up status Register (**I2CWKUPSTS**)*

When system is woken up by other I^{2}C master device, WKUPIF is set to indicate this event

WKUPIF Wake-up interrupt flag

5.8.5 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
I2CWKUPCON	I2Cx_BA+0x3C	R/W	I^{2}C Wake-up Control Register	0x0000_0000
I2CWKUPSTS	I2Cx_BA+0x40	R	I^{2}C Wake-up Status Register	0x0000_0000

5.8.6 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	Description	
[31:8]	**Reserved**	Reserved
[7]	**EI**	**Enable Interrupt** 1 = I²C interrupt Enabled. 0 = I²C interrupt Disabled.
[6]	**ENS1**	**I²C Controller Enable Bit** 1 = Enabled. 0 = Disabled. 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

I²C Data Register (I2CDAT)

Register	Offset	R/W	Description	Reset Value
I2CDAT	I2C_BA+0x08	R/W	I²C Data Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
I2CDAT[7:0]							

Bits	Description	
[31:8]	**Reserved**	Reserved
[7:0]	**I2CDAT**	**I²C Data Register** Bit [7:0] is located with the 8-bit transferred data of I²C serial port.

I^2C Status Register (I2CSTATUS)

Register	Offset	R/W	Description	Reset Value
I2CSTATUS	I2C_BA+0x0C	R/W	I^2C 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	Description	
[31:8]	**Reserved**	Reserved
[7:0]	**I2CSTATUS**	**I^2C Status Register** The status register of I^2C: 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^2C 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^{2}C Clock Divided Register (I2CLK)

Register	Offset	R/W	Description	Reset Value
I2CLK	I2C_BA+0x10	R/W	I^{2}C 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	Description	
[31:8]	**Reserved**	Reserved
[7:0]	**I2CLK**	**I^{2}C clock divided Register** The I^{2}C clock rate bits: Data Baud Rate of I^{2}C = (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	Description	
[31:3]	**Reserved**	Reserved
[2]	**ENTI**	**Time-out Counter Enable/Disable** 1 = Enabled. 0 = Disabled. When Enabled, the 14-bit time-out counter will start counting when SI is cleared. Setting the flag SI to high will reset counter and re-start counting up after SI is cleared.
[1]	**DIV4**	**Time-out Counter Input Clock Divided by 4** 1 = Enabled. 0 = Disabled. When Enabled, the time-out period is extend 4 times.
[0]	**TIF**	**Time-Out Flag** 1 = Time-out flag is set by H/W. It can interrupt CPU. 0 = Software can clear the flag.

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	Description	
[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 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 addresses is matched.
[0]	**GC**	**General Call Function** 0 = General Call function Disabled. 1 = General Call function Enabled.

I²C Slave Address Mask Register (I2CADMx)

Register	Offset	R/W	Description	Reset Value
I2CADM0	I2C_BA+0x24	R/W	I²C Slave Address Mask Register0	0x0000_0000
I2CADM1	I2C_BA+0x28	R/W	I²C Slave Address Mask Register1	0x0000_0000
I2CADM2	I2C_BA+0x2C	R/W	I²C Slave Address Mask Register2	0x0000_0000
I2CADM3	I2C_BA+0x30	R/W	I²C Slave Address Mask Register3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
I2CADM[7:1]							**Reserved**

Bits	Description	
[31:8]	**Reserved**	Reserved
[7:1]	**I2CADM**	**I²C Address Mask Register** 1 = Mask Enabled (the received corresponding address bit is don't care.) 0 = Mask Disabled (the received corresponding register bit should be exactly the same as address register.) I²C bus controllers support multiple address recognition with four address mask register. When the bit in the address mask register is set to one, it means the received corresponding address bit is don't-care. If the bit is set to zero, it means the received corresponding register bit should be exact the same as address register.
[0]	**Reserved**	Reserved

I^{2}C Wake-up Control Register (I2WKUPCON)

Register	Offset	R/W	Description	Reset Value
I2CWKUPCON	I2C_BA+0x3C	R/W	I^{2}C Wake-up Control Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved
[0]	**WKUPEN**	**I^{2}C Wake-up Function Enable** 1 = I^{2}C wake-up function Enabled. 0 = I^{2}C wake-up function Disabled.

I^{2}C Wake-up StatusRegister (I2WKUPSTS)

Register	Offset	R/W	Description	Reset Value
I2CWKUPSTS	I2C_BA+0x40	R	I^{2}C 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							**WKUPIF**

Bits	Description	
[31:1]	**Reserved**	Reserved
[0]	**WKUPIF**	**Wake-up Interrupt Flag** 1 = Wake-up flag active. 0 = Wake-up flag inactive. Software can write one to clear this flag

5.8.7 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 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 entering Master mode so that a possible slave action didn't be interrupted. If bus arbitration is lost in Master mode, I^{2}C port switches to the Slave mode immediately and can detect its own slave address in the same serial transfer.

5.8.7.1 *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.

5.8.7.2 *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.

5.8.7.3 *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.

5.8.7.4 *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.

5.8.8 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^{2}C 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 I^{2}C 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 5-32 to Figure 5-37.

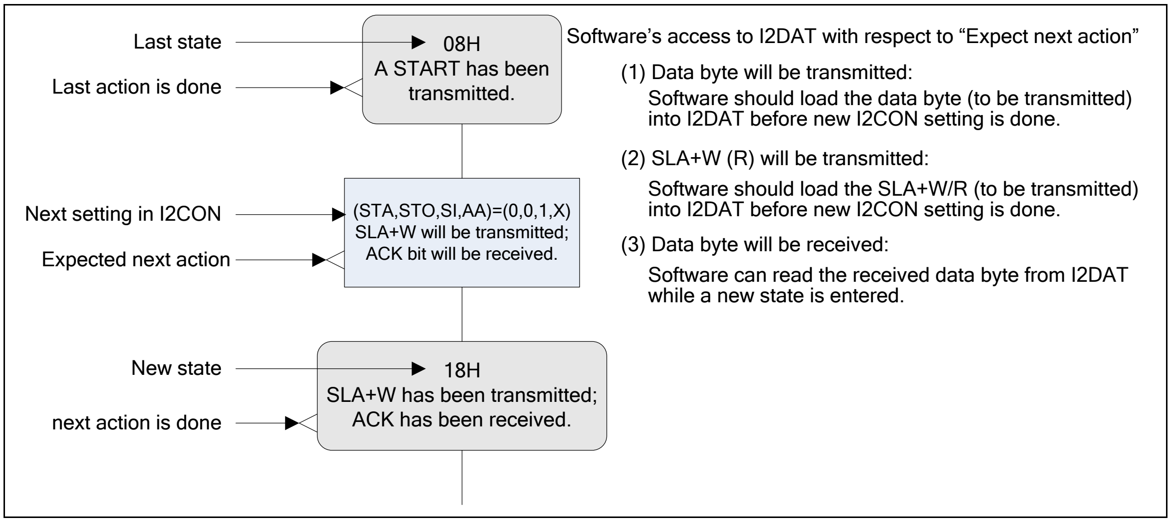

Last state
Last action is done
08H
A START has been transmitted.
Next setting in I2CON
Expected next action
(STA,STO,SI,AA)=(0,0,1,X)
SLA+W will be transmitted;
ACK bit will be received.
New state
next action is done
18H
SLA+W has been transmitted;
ACK has been received.
Software's access to I2DAT with respect to "Expect next action"
(1) Data byte will be transmitted:
Software should load the data byte (to be transmitted) into I2DAT before new I2CON setting is done.
(2) SLA+W (R) will be transmitted:
Software should load the SLA+W/R (to be transmitted) into I2DAT before new I2CON setting is done.
(3) Data byte will be received:
Software can read the received data byte from I2DAT while a new state is entered.

Figure 5-32 Legend for the Following Five Figures

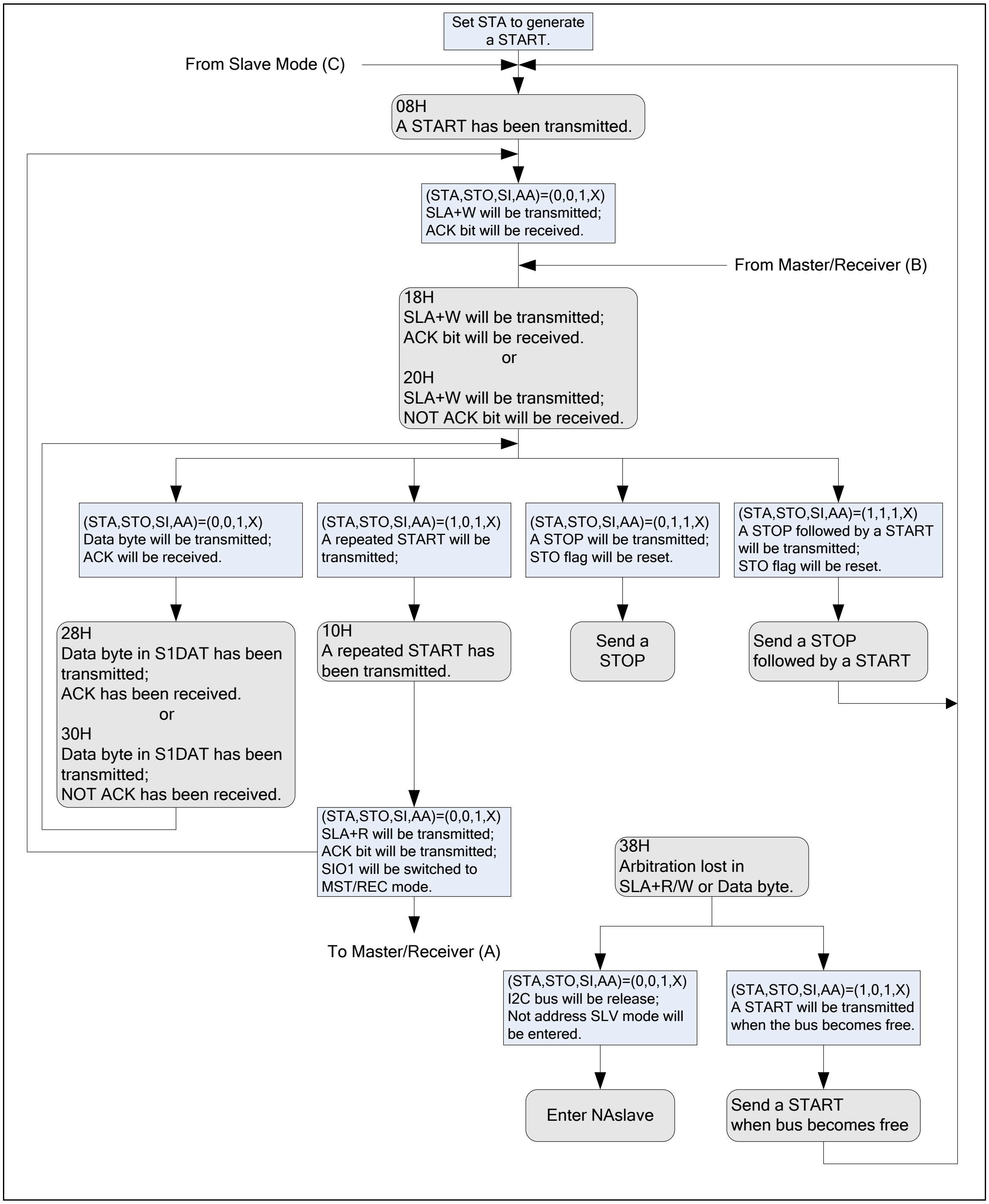

Set STA to generate a START.
From Slave Mode (C)
08H
A START has been transmitted.
(STA,STO,SI,AA)=(0,0,1,X)
SLA+W will be transmitted;
ACK bit will be received.
From Master/Receiver (B)
18H
SLA+W will be transmitted;
ACK bit will be received.
or
20H
SLA+W will be transmitted;
NOT ACK bit will be received.
(STA,STO,SI,AA)=(0,0,1,X)
Data byte will be transmitted;
ACK will be received.
(STA,STO,SI,AA)=(1,0,1,X)
A repeated START will be transmitted;
(STA,STO,SI,AA)=(0,1,1,X)
A STOP will be transmitted;
STO flag will be reset.
(STA,STO,SI,AA)=(1,1,1,X)
A STOP followed by a START will be transmitted;
STO flag will be reset.
28H
Data byte in S1DAT has been transmitted;
ACK has been received.
or
30H
Data byte in S1DAT has been transmitted;
NOT ACK has been received.
10H
A repeated START has been transmitted.
Send a STOP
Send a STOP followed by a START
(STA,STO,SI,AA)=(0,0,1,X)
SLA+R will be transmitted;
ACK bit will be transmitted;
SIO1 will be switched to MST/REC mode.
To Master/Receiver (A)
38H
Arbitration lost in SLA+R/W or Data byte.
(STA,STO,SI,AA)=(0,0,1,X)
I2C bus will be release;
Not address SLV mode will be entered.
(STA,STO,SI,AA)=(1,0,1,X)
A START will be transmitted when the bus becomes free.
Enter NAslave
Send a START when bus becomes free

Figure 5-33 Master Transmitter Mode

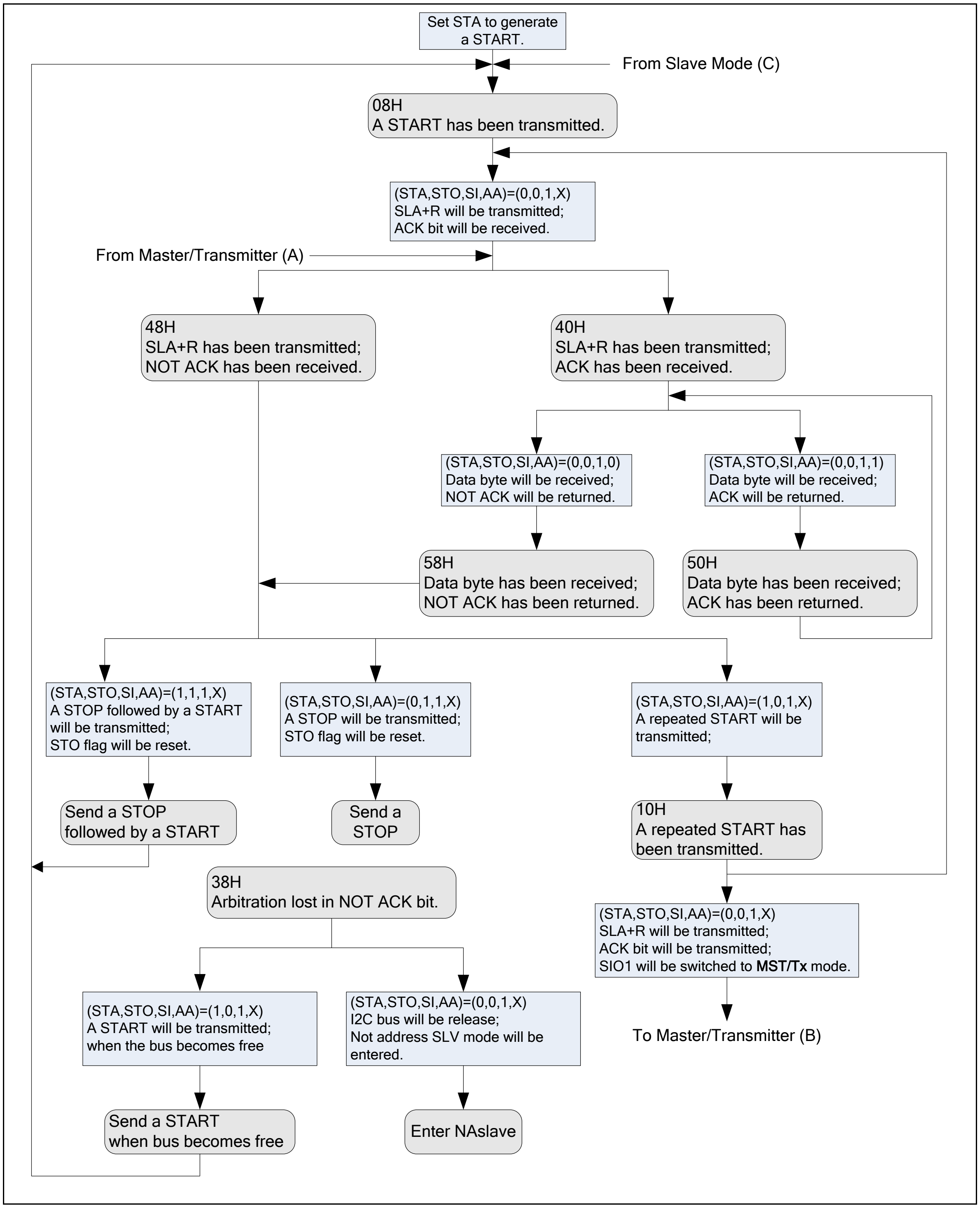

Set STA to generate a START.
From Slave Mode (C)
08H
A START has been transmitted.
(STA,STO,SI,AA)=(0,0,1,X)
SLA+R will be transmitted;
ACK bit will be received.
From Master/Transmitter (A)
48H
SLA+R has been transmitted;
NOT ACK has been received.
40H
SLA+R has been transmitted;
ACK has been received.
(STA,STO,SI,AA)=(0,0,1,0)
Data byte will be received;
NOT ACK will be returned.
(STA,STO,SI,AA)=(0,0,1,1)
Data byte will be received;
ACK will be returned.
58H
Data byte has been received;
NOT ACK has been returned.
50H
Data byte has been received;
ACK has been returned.
(STA,STO,SI,AA)=(1,1,1,X)
A STOP followed by a START will be transmitted;
STO flag will be reset.
(STA,STO,SI,AA)=(0,1,1,X)
A STOP will be transmitted;
STO flag will be reset.
(STA,STO,SI,AA)=(1,0,1,X)
A repeated START will be transmitted;
Send a STOP followed by a START
Send a STOP
10H
A repeated START has been transmitted.
38H
Arbitration lost in NOT ACK bit.
(STA,STO,SI,AA)=(0,0,1,X)
SLA+R will be transmitted;
ACK bit will be transmitted;
SIO1 will be switched to MST/Tx mode.
(STA,STO,SI,AA)=(1,0,1,X)
A START will be transmitted;
when the bus becomes free
(STA,STO,SI,AA)=(0,0,1,X)
I2C bus will be release;
Not address SLV mode will be entered.
To Master/Transmitter (B)
Send a START when bus becomes free
Enter NAslave

Figure 5-34 Master Receiver Mode

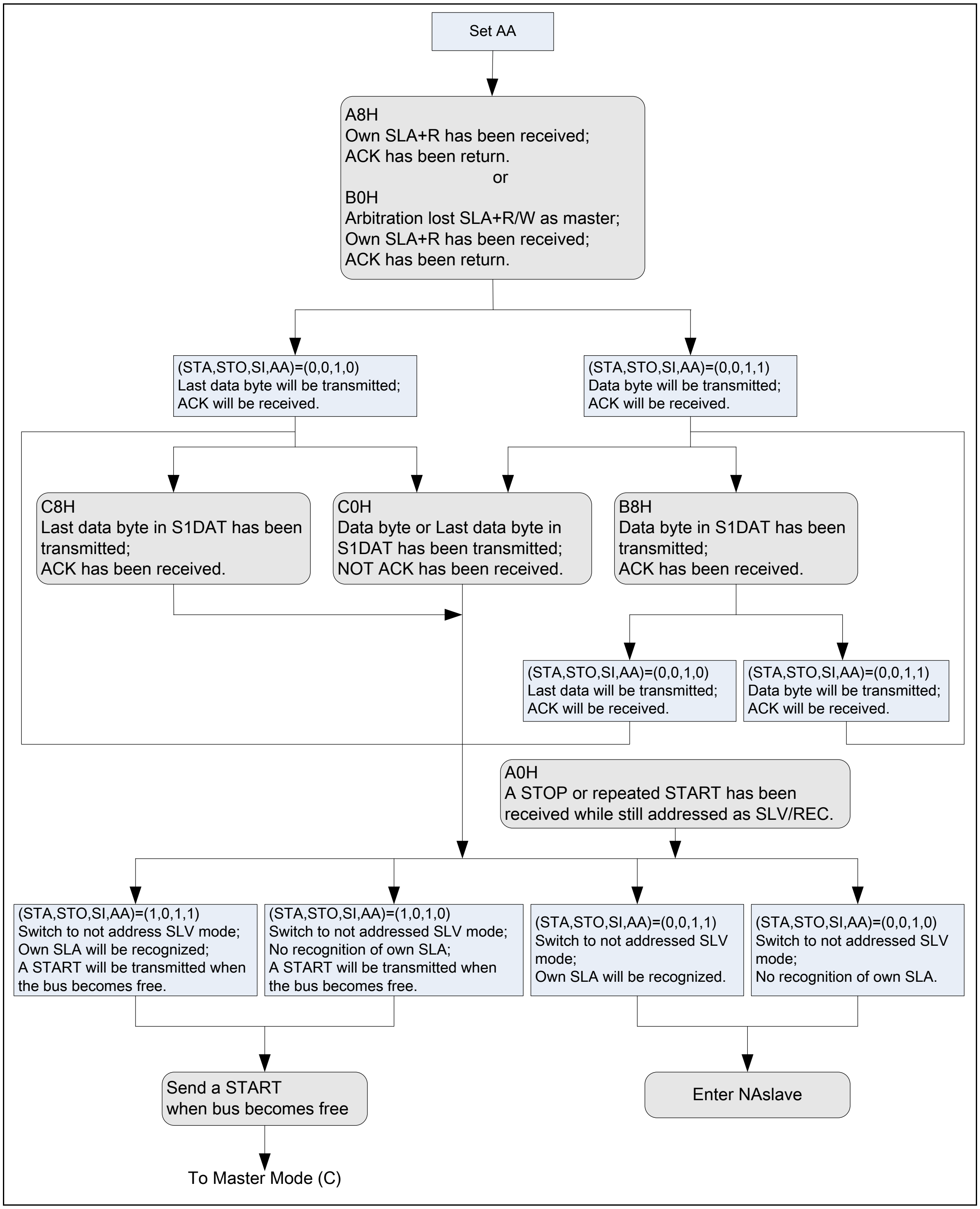

Set AA
A8H
Own SLA+R has been received;
ACK has been return.
or
B0H
Arbitration lost SLA+R/W as master;
Own SLA+R has been received;
ACK has been return.
(STA,STO,SI,AA)=(0,0,1,0)
Last data byte will be transmitted;
ACK will be received.
(STA,STO,SI,AA)=(0,0,1,1)
Data byte will be transmitted;
ACK will be received.
C8H
Last data byte in S1DAT has been transmitted;
ACK has been received.
C0H
Data byte or Last data byte in S1DAT has been transmitted;
NOT ACK has been received.
B8H
Data byte in S1DAT has been transmitted;
ACK has been received.
(STA,STO,SI,AA)=(0,0,1,0)
Last data will be transmitted;
ACK will be received.
(STA,STO,SI,AA)=(0,0,1,1)
Data byte will be transmitted;
ACK will be received.
A0H
A STOP or repeated START has been received while still addressed as SLV/REC.
(STA,STO,SI,AA)=(1,0,1,1)
Switch to not address SLV mode;
Own SLA will be recognized;
A START will be transmitted when the bus becomes free.
(STA,STO,SI,AA)=(1,0,1,0)
Switch to not addressed SLV mode;
No recognition of own SLA;
A START will be transmitted when the bus becomes free.
(STA,STO,SI,AA)=(0,0,1,1)
Switch to not addressed SLV mode;
Own SLA will be recognized.
(STA,STO,SI,AA)=(0,0,1,0)
Switch to not addressed SLV mode;
No recognition of own SLA.
Send a START when bus becomes free
Enter NAslave
To Master Mode (C)

Figure 5-35 Slave Transmitter Mode

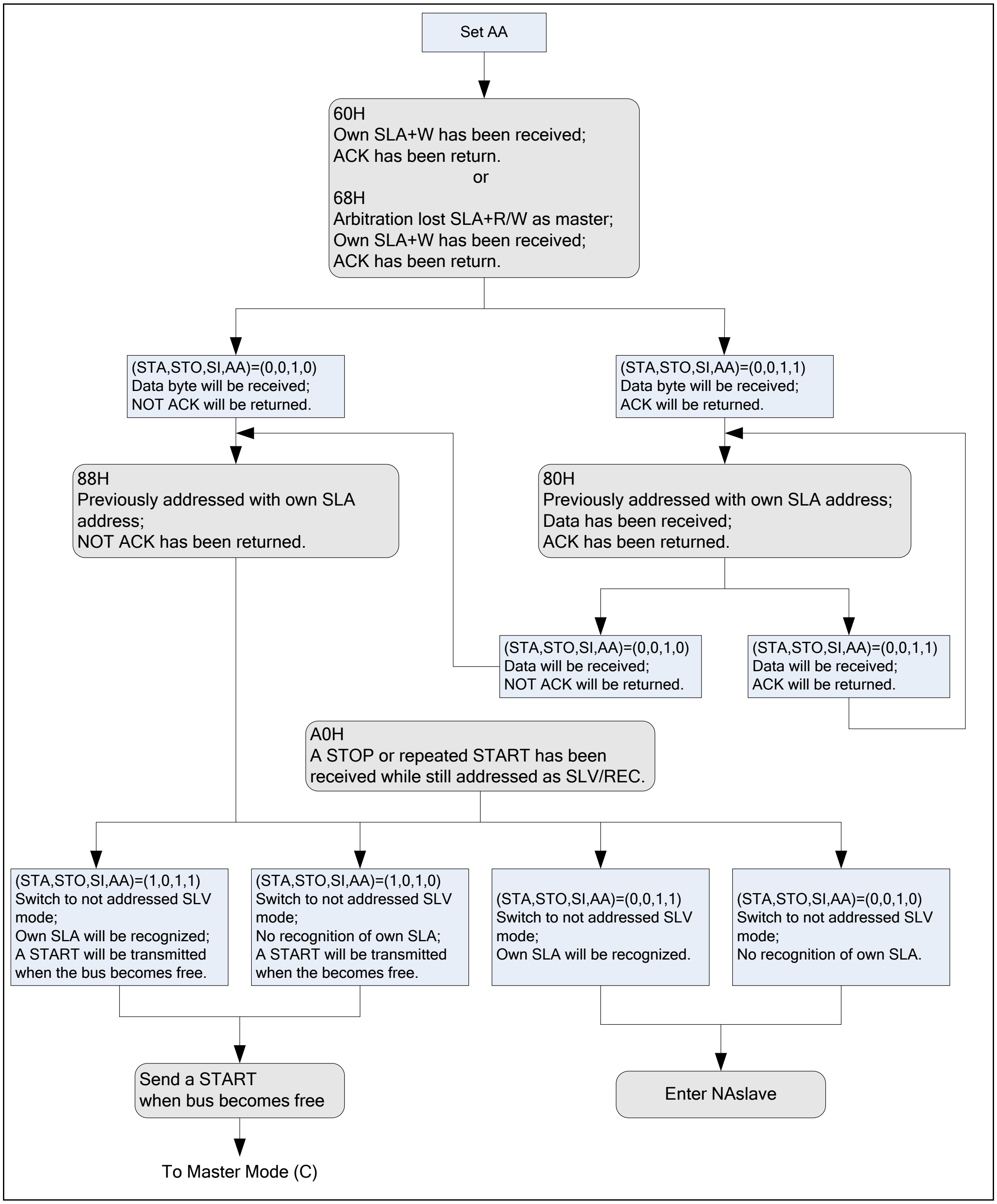

Set AA
60H
Own SLA+W has been received;
ACK has been return.
or
68H
Arbitration lost SLA+R/W as master;
Own SLA+W has been received;
ACK has been return.
(STA,STO,SI,AA)=(0,0,1,0)
Data byte will be received;
NOT ACK will be returned.
(STA,STO,SI,AA)=(0,0,1,1)
Data byte will be received;
ACK will be returned.
88H
Previously addressed with own SLA address;
NOT ACK has been returned.
80H
Previously addressed with own SLA address;
Data has been received;
ACK has been returned.
(STA,STO,SI,AA)=(0,0,1,0)
Data will be received;
NOT ACK will be returned.
(STA,STO,SI,AA)=(0,0,1,1)
Data will be received;
ACK will be returned.
A0H
A STOP or repeated START has been received while still addressed as SLV/REC.
(STA,STO,SI,AA)=(1,0,1,1)
Switch to not addressed SLV mode;
Own SLA will be recognized;
A START will be transmitted when the bus becomes free.
(STA,STO,SI,AA)=(1,0,1,0)
Switch to not addressed SLV mode;
No recognition of own SLA;
A START will be transmitted when the bus becomes free.
(STA,STO,SI,AA)=(0,0,1,1)
Switch to not addressed SLV mode;
Own SLA will be recognized.
(STA,STO,SI,AA)=(0,0,1,0)
Switch to not addressed SLV mode;
No recognition of own SLA.
Send a START when bus becomes free
Enter NAslave
To Master Mode (C)

Figure 5-36 Slave Receiver Mode

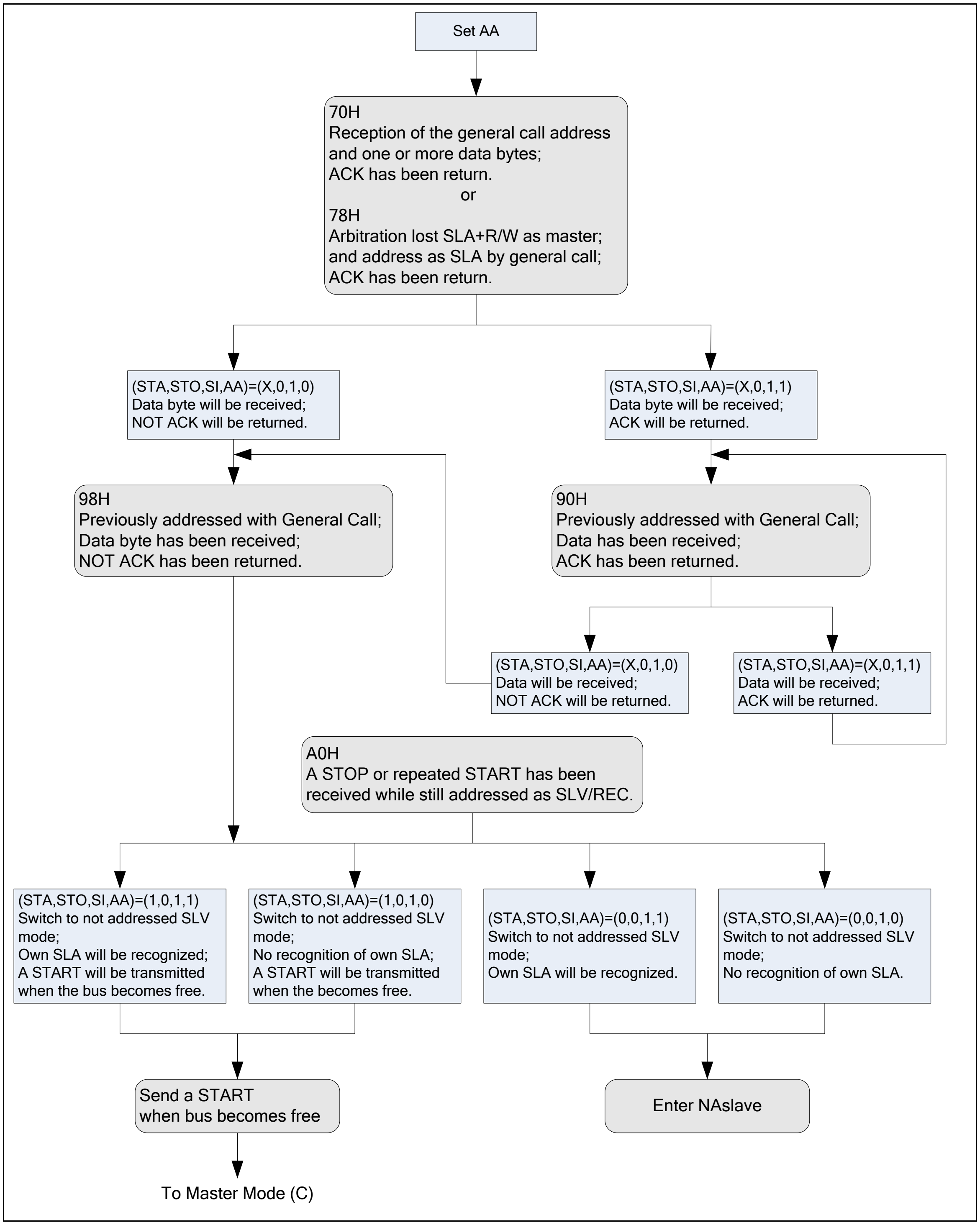

Set AA
70H
Reception of the general call address and one or more data bytes;
ACK has been return.
or
78H
Arbitration lost SLA+R/W as master; and address as SLA by general call;
ACK has been return.
(STA,STO,SI,AA)=(X,0,1,0)
Data byte will be received;
NOT ACK will be returned.
(STA,STO,SI,AA)=(X,0,1,1)
Data byte will be received;
ACK will be returned.
98H
Previously addressed with General Call;
Data byte has been received;
NOT ACK has been returned.
90H
Previously addressed with General Call;
Data has been received;
ACK has been returned.
(STA,STO,SI,AA)=(X,0,1,0)
Data will be received;
NOT ACK will be returned.
(STA,STO,SI,AA)=(X,0,1,1)
Data will be received;
ACK will be returned.
A0H
A STOP or repeated START has been received while still addressed as SLV/REC.
(STA,STO,SI,AA)=(1,0,1,1)
Switch to not addressed SLV mode;
Own SLA will be recognized;
A START will be transmitted when the bus becomes free.
(STA,STO,SI,AA)=(1,0,1,0)
Switch to not addressed SLV mode;
No recognition of own SLA;
A START will be transmitted when the becomes free.
(STA,STO,SI,AA)=(0,0,1,1)
Switch to not addressed SLV mode;
Own SLA will be recognized.
(STA,STO,SI,AA)=(0,0,1,0)
Switch to not addressed SLV mode;
No recognition of own SLA.
Send a START when bus becomes free
Enter NAslave
To Master Mode (C)

Figure 5-37 GC Mode

5.9 PWM Generator and Capture Timer (PWM)

5.9.1 Overview

This chip has 1 set of PWM group supporting 1 set of PWM generators which can be configured as 4 independent PWM outputs, PWM0~PWM3, or as 2 complementary PWM pairs, (PWM0, PWM1), (PWM2, PWM3) with two 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 PWM generators provide four 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) , timers and dead-zone generator 2, respectively. Refer to Figure 5-38 to Figure 5-41 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 perform at least three steps including: 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 takes time T0 to finish, the capture signal must not 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

Therefore, the maximum capture frequency will be 1/900ns ≈ 1000 kHz

5.9.2 Features

5.9.2.1 *PWM function:*

- Two PWM generators, each supporting 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 1 PWM group (PWMA) to support 4 PWM channels or 2 PWM paired channels

5.9.2.2 *Capture function:*

- Timing control logic shared with PWM Generators
- Supports 4 Capture input channels shared with 4 PWM output channels
- Each channel supports one rising latch register (CRLR), one falling latch register (CFLR) and Capture interrupt flag (CAPIFx)
- Supports PDMA transfer function for each channel

5.9.3 Block Diagram

Figure 5-38 to Figure 5-41 illustrate the architecture of PWM in pair (i.e. PWM-Timer 0/1 are in one pair and PWM-Timer 2/3 are in the other one).

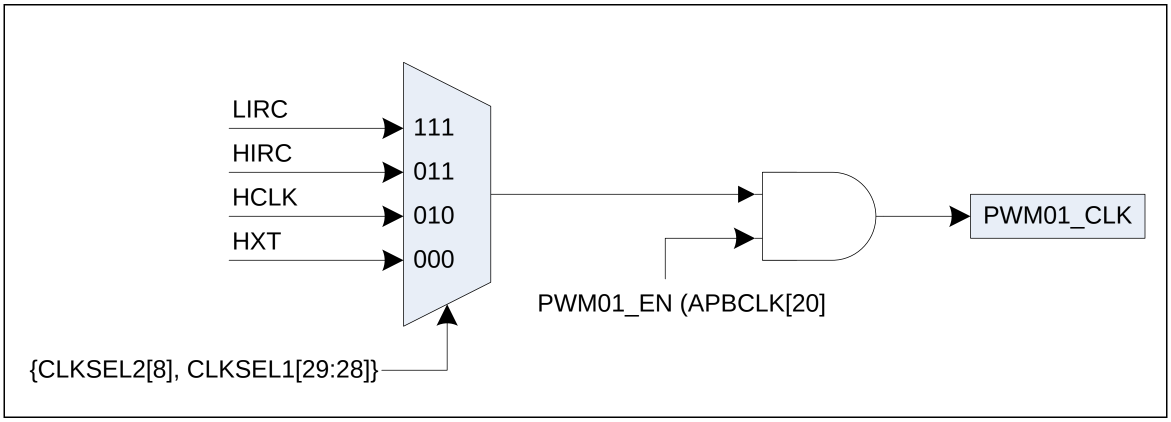


Figure 5-38 PWM Generator 0 Clock Source Control

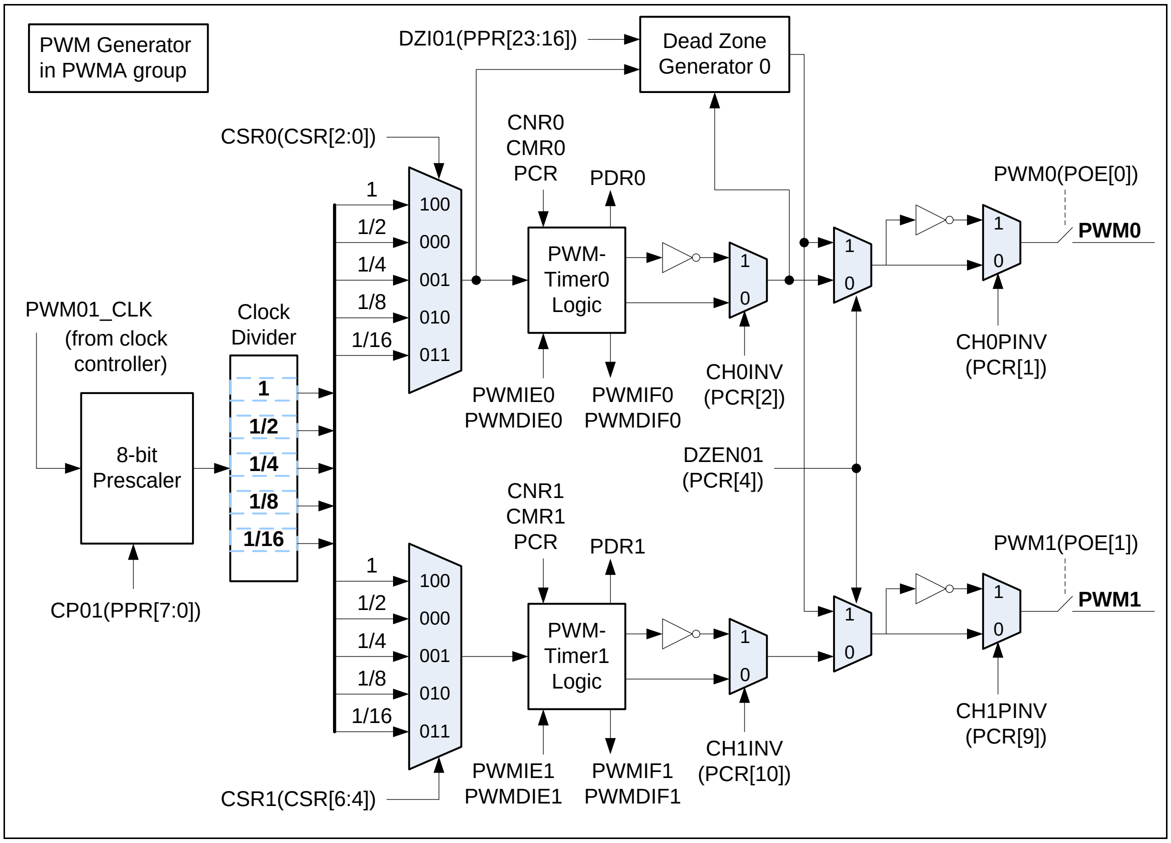


Figure 5-39 PWM Generator 0 Architecture Diagram

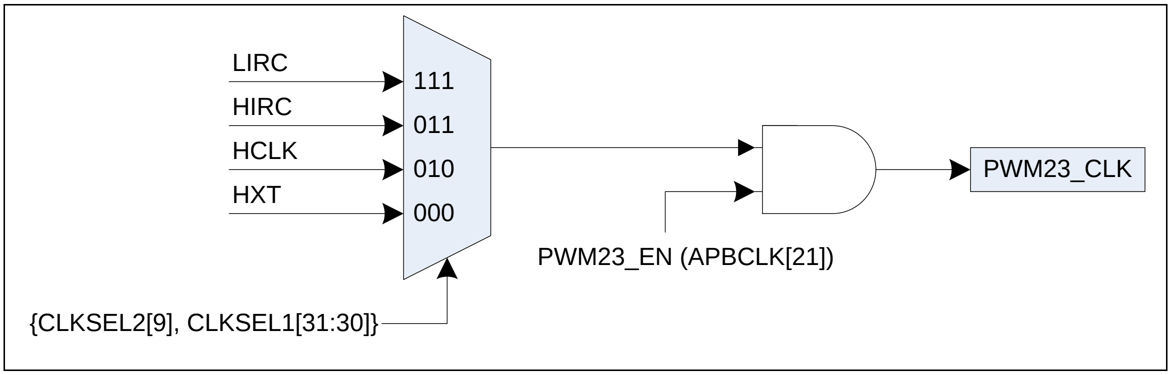


Figure 5-40 PWM Generator 2 Clock Source Control

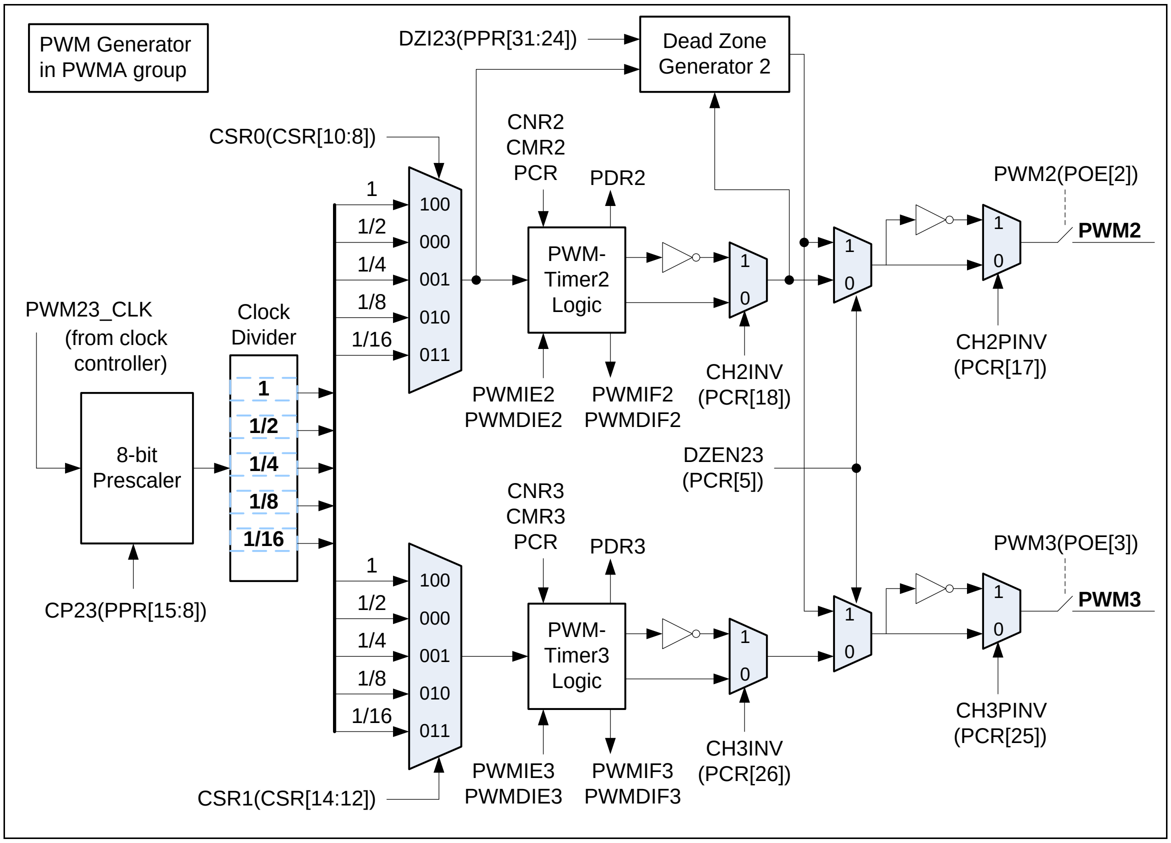


Figure 5-41 PWM Generator 2 Architecture Diagram

5.9.4 Functional Description

5.9.4.1 *PWM-Timer Operation*

The PWM controller supports two operation modes: Edge-aligned and Center-aligned mode.

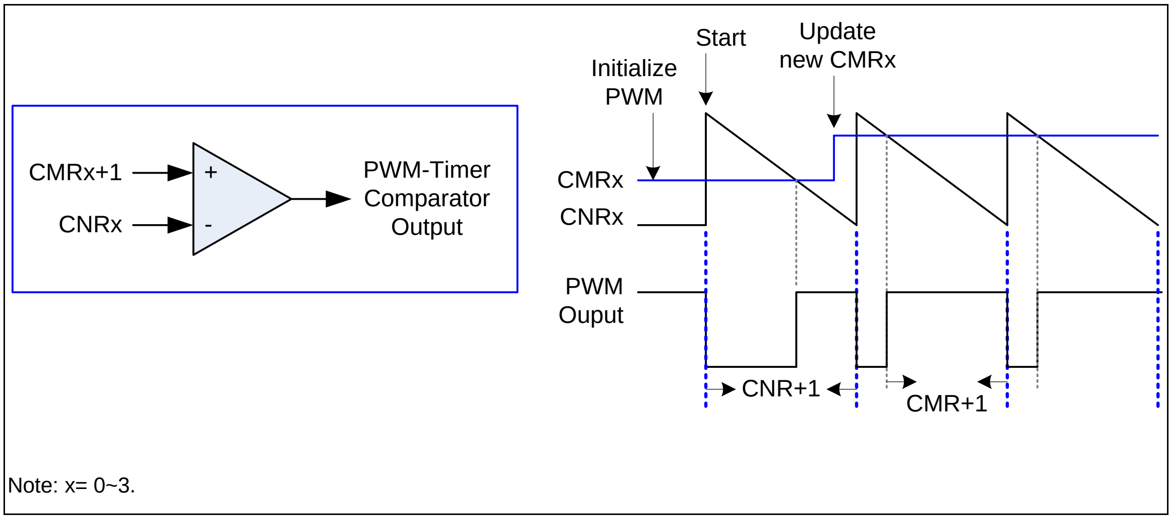
5.9.4.1.1 Edge-aligned PWM (Down-counter)

In Edge-aligned PWM Output mode, the 16 bits PWM counter will starts down-counting from CNRn to match with the value of the duty cycle CMRn (old), when this happens it will toggle the PWMn generator output to low. The counter will continue counting down to zero, at this moment, it toggles the PWMn generator output to high and CMRn (new) and CNRn (new) are updated with CHnMODE=1 and request the PWM interrupt if PWM interrupt is enabled (PIER.n=1).

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 5-43. The pulse width modulation follows the formula below and the legend of PWM-Timer Comparator is shown as Figure 5-42. Note that the corresponding GPIO pins must be configured as PWM function (enabling POE and disabling 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 depending on the 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.



Note: x= 0~3.

Figure 5-42 Legend of Internal Comparator Output of PWM-Timer

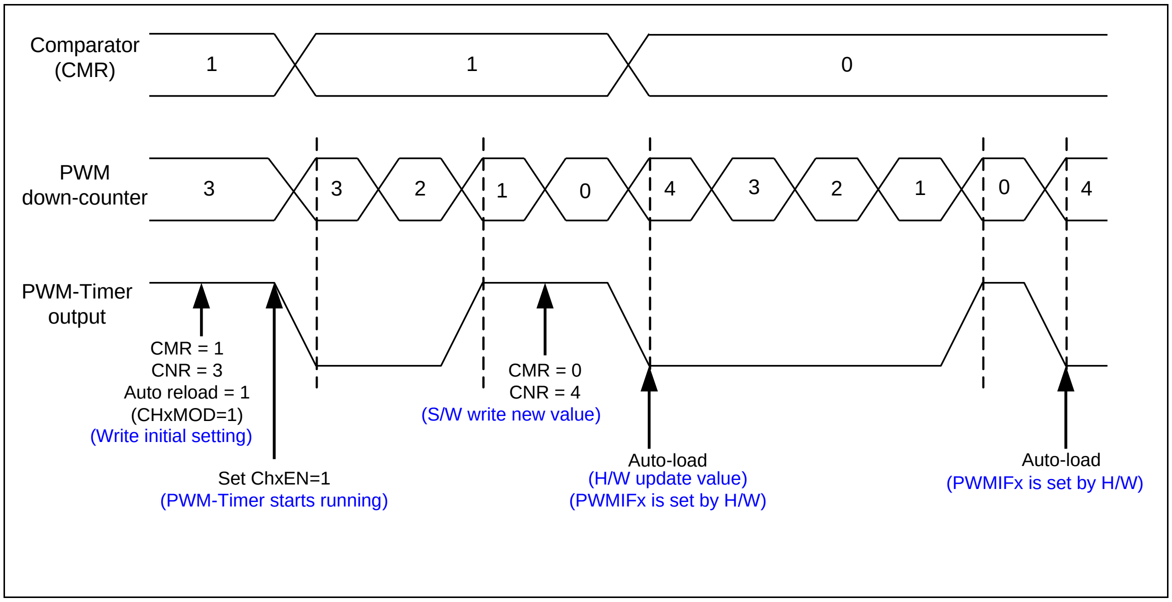


Figure 5-43 PWM-Timer Operation Timing

5.9.4.1.2 Center-aligned PWM (up/down-counter)

The Center-aligned PWM signals are produced by the module when the PWM time base is configured in Up/Down Counting mode. The PWM counter will start counting-up from 0 to match the value of CMRn (old); this will cause the toggling of the PWMn generator output to low. The counter will continue counting to match with the CNRn (old). Upon reaching this state counter is configured automatically to count down, when PWM counter matches the CMRn (old) value again the PWMn generator outputs toggles to high. Once the PWM counter underflows it will update the PWM period register CNRn (new) and duty cycle register CMRn (new) with CHnMODE = 1.

In Center-aligned mode, the PWM period interrupt is requested at down-counter underflow if INTTYPExx (PIER[17:16]) =0, i.e. at start (end) of each PWM cycle or at up-counter matching with CNRn if INTTYPExx (PIER[17:16]) =1, i.e. at center point of PWM cycle.

- PWM frequency = PWMxy_CLK/[(prescale+1)*(clock divider)*(CNR+1)]; where xy could be 01, 23, 45 or 67 depending on selected PWM channel.
- Duty ratio = [(2 x CMR) + 1]/[2 x (CNR+1)]
- CMR > CNR: PWM output is always high
- CMR <= CNR: PWM low width= 2 x (CNR-CMR) + 1 unit[1]; PWM high width = (2 x CMR) + 1 unit
- CMR = 0: PWM low width = 2 x CNR + 1 unit; PWM high width = 1 unit

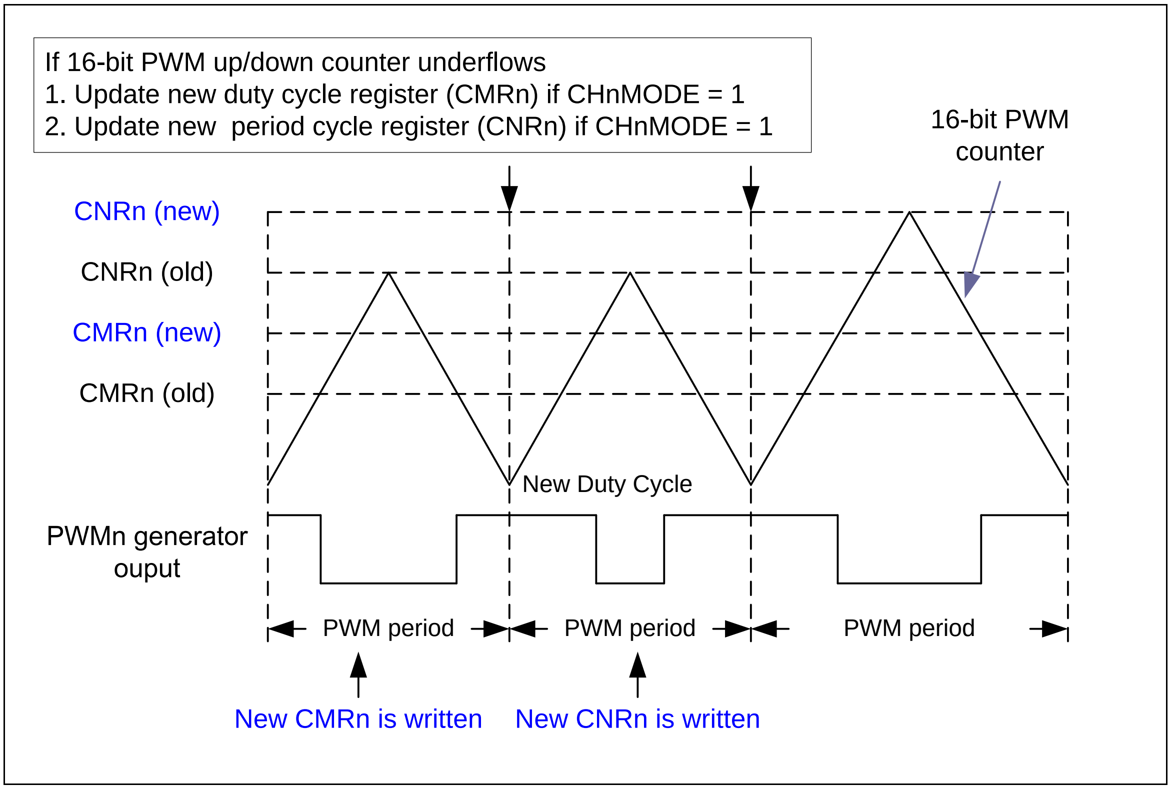


Figure 5-44 Center-Aligned Mode Output Waveform

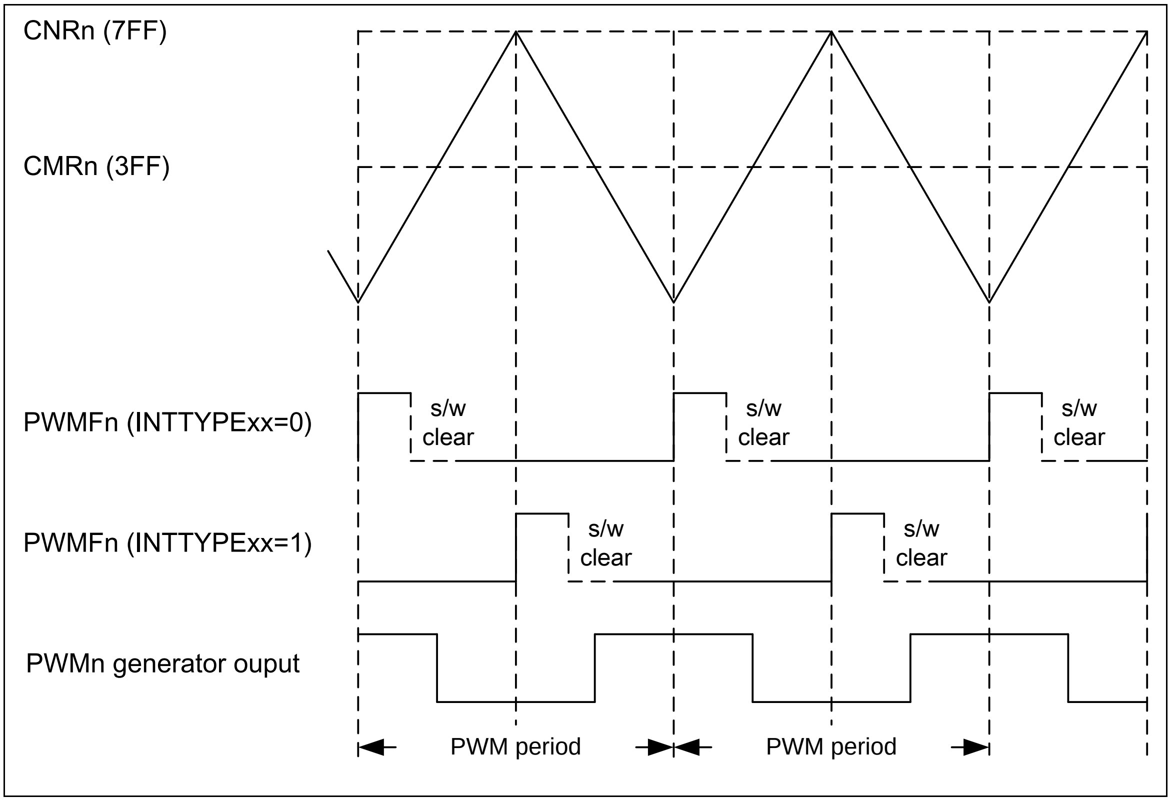


Figure 5-45 PWM Center Aligned Interrupt Generate Timing Waveform

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

PWM Timers have double buffering function by which 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 the current PWM counter value can be read from PDRx.

The bit CH0MOD in PWM Control Register (PCR) defines PWM0 operated in Auto-reload or One-shot mode. If CH0MOD is set as one, the auto-reload operation loads CNR0 to PWM counter when PWM counter reaches zero. If CNR0 is set as zero, PWM counter will be halted 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.

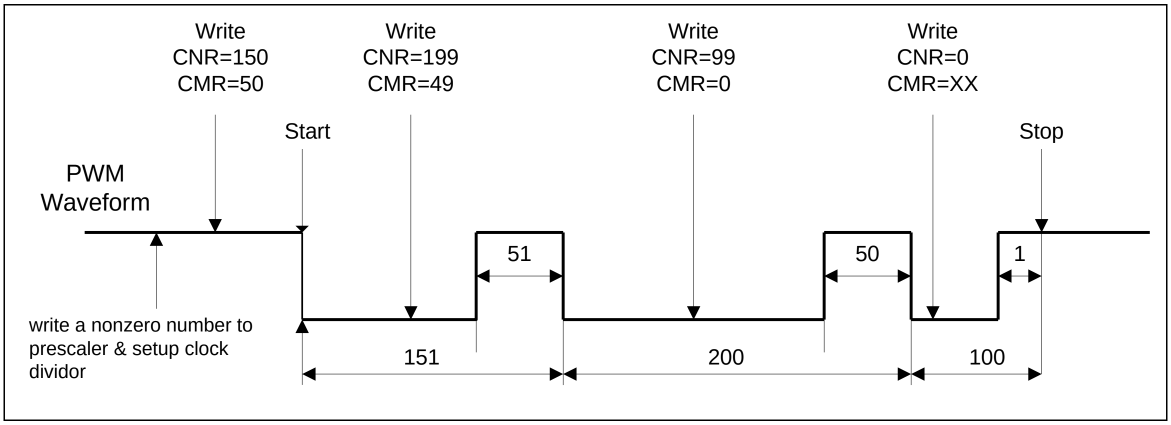


Figure 5-46 PWM Double Buffering Illustration

5.9.4.3 *Modulate Duty Ratio*

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

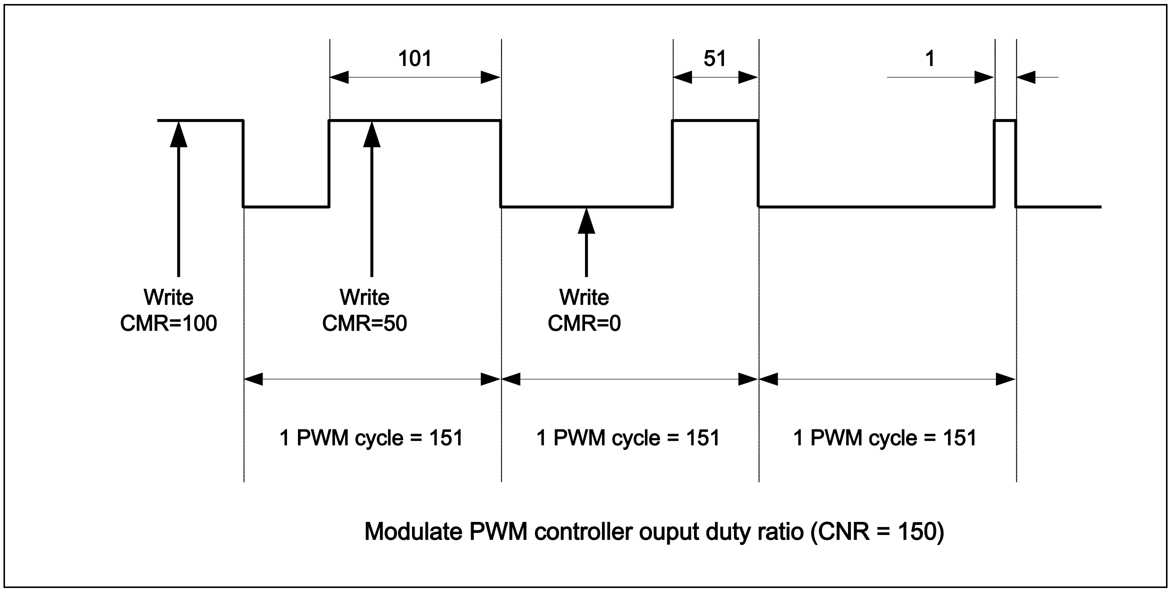


Figure 5-47 PWM Controller Output Duty Ratio

5.9.4.4 *Dead-zone Generator*

PWM controller is implemented with a 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.

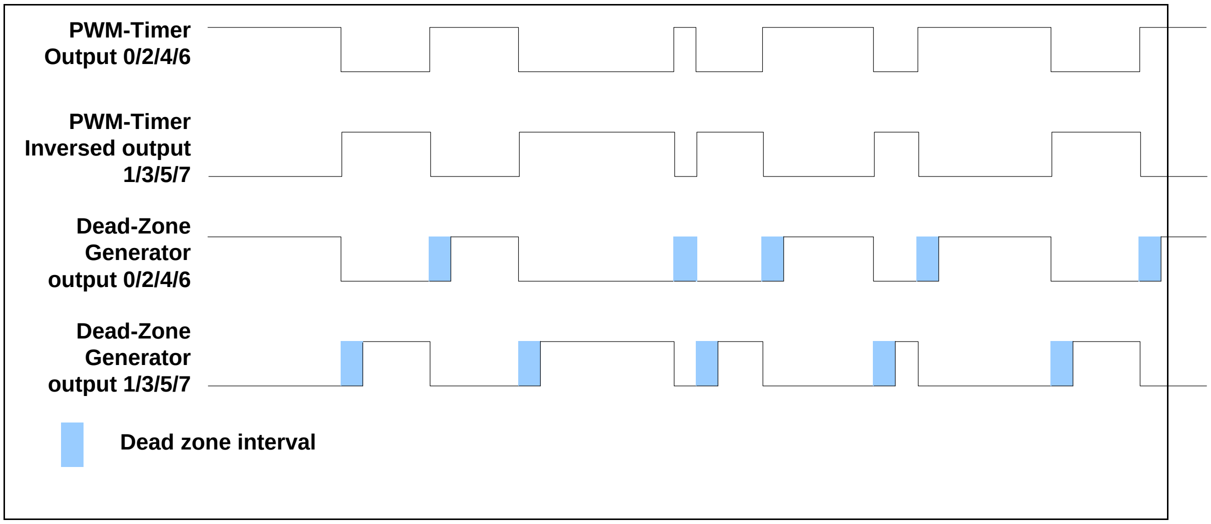

PWM-Timer Output 0/2/4/6
PWM-Timer Inversed output 1/3/5/7
Dead-Zone Generator output 0/2/4/6
Dead-Zone Generator output 1/3/5/7
Dead zone interval

Figure 5-48 Paired-PWM Output with Dead-zone Generation Operation

5.9.4.5 *Capture Operation*

The Capture 0 and PWM 0 share one timer included in PWM 0, and the Capture 1 and PWM 1 share another timer. 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 (disabling POE and enabling CAPENR) for the corresponding capture channel.

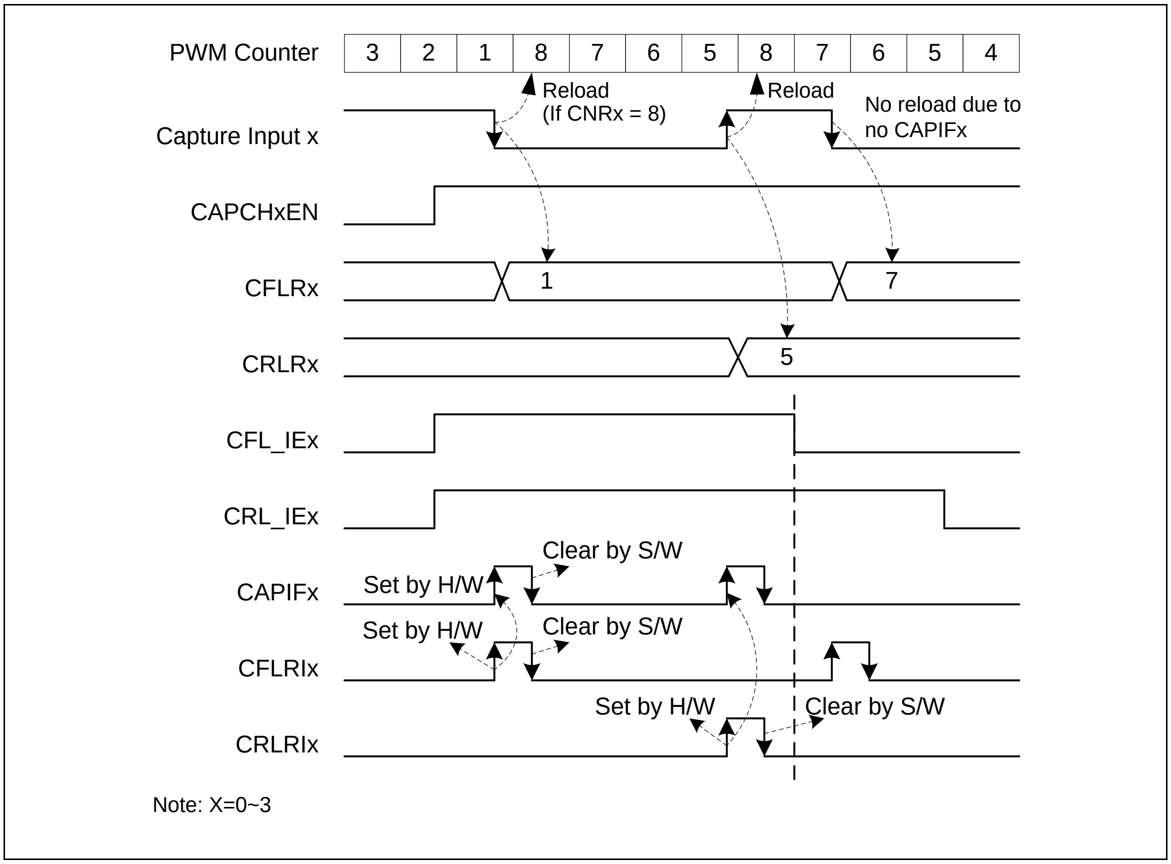


Figure 5-49 Capture Operation Timing

In this case, 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).

5.9.4.6 *PWM PDMA function*

PWM supports PDMA transfer function for each channel when operating in Capturing mode. Take channel 0 for example, when the corresponding PDMA enable bit (defined in CAPCTL register) is set, the capturing module will issue a request to PDMA controller when the preceding capture event happened. The PDMA controller will issue ack to capture module and read back CAP0RFPDMA register to memory. By setting CAP0PDMAMOD, PDMA can transfer rising latched data or falling latched data or both of them to memory. When using PDMA to transfer both falling and rising data, remember to set CAP0RFORDER in PWM_CAPCTL to decide the order of transferring data (whether the falling edge latched is first or rising edged latched is first).

5.9.4.7 *PWM-Timer Interrupt Architecture*

There are four PWM interrupts, PWM0_INT~PWM3_INT, which are grouped into PWMA_INT for Advanced Interrupt Controller (AIC). That is, PWM 0 and Capture 0 share one interrupt, and PWM1 and Capture 1 share the same interrupt. Therefore, PWM function and Capture function in the same channel cannot be used at the same time. Figure 5-50 demonstrates the architecture of PWM-Timer interrupts.

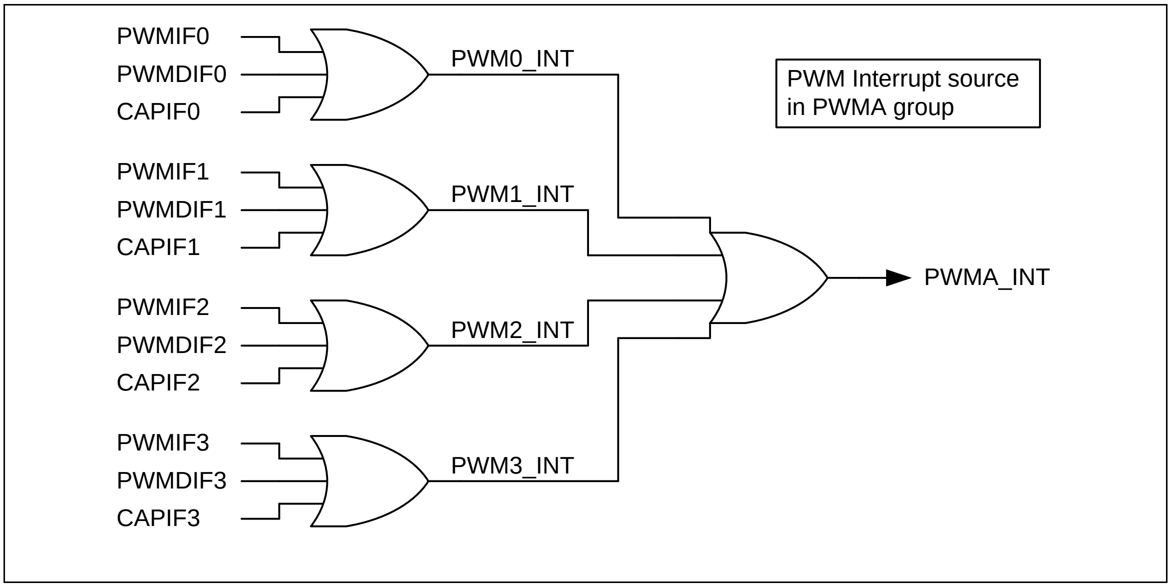


Figure 5-50 PWM Group A PWM-Timer Interrupt Architecture Diagram

PWM-Timer Start Procedure

The following procedure is recommended for starting a PWM drive:

1. Set clock selector (CSR)
2. Set prescaler (PPR)
3. Set inverter on/off, dead-zone generator on/off, auto-reload/one-shot mode and stop PWM-timer (PCR)
4. Set comparator register (CMR) for setting PWM duty.
5. Set PWM down-counter register (CNR) for setting PWM period.
6. Set interrupt enable register (PIER)
7. Set the 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)

5.9.4.8 *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 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 disabling 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.

5.9.4.9 *Capture Start Procedure*

1. Set clock selector (CSR)
2. Set prescaler (PPR)
3. Set channel enabled, rising/falling interrupt enable and input signal inverter on/off (CCR0, CCR2)
4. Set PWM down-counter (CNR)
5. Set 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)

5.9.5 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)				
PPR	PWMA_BA+0x00	R/W	PWM Group A Prescaler Register	0x0000_0000
CSR	PWMA_BA+0x04	R/W	PWM Group A Clock Select Register	0x0000_0000
PCR	PWMA_BA+0x08	R/W	PWM Group A Control Register	0x0000_0000
CNR0	PWMA_BA+0x0C	R/W	PWM Group A Counter Register 0	0x0000_0000
CMR0	PWMA_BA+0x10	R/W	PWM Group A Comparator Register 0	0x0000_0000
PDR0	PWMA_BA+0x14	R	PWM Group A Data Register 0	0x0000_0000
CNR1	PWMA_BA+0x18	R/W	PWM Group A Counter Register 1	0x0000_0000
CMR1	PWMA_BA+0x1C	R/W	PWM Group A Comparator Register 1	0x0000_0000
PDR1	PWMA_BA+0x20	R	PWM Group A Data Register 1	0x0000_0000
CNR2	PWMA_BA+0x24	R/W	PWM Group A Counter Register 2	0x0000_0000
CMR2	PWMA_BA+0x28	R/W	PWM Group A Comparator Register 2	0x0000_0000
PDR2	PWMA_BA+0x2C	R	PWM Group A Data Register 2	0x0000_0000
CNR3	PWMA_BA+0x30	R/W	PWM Group A Counter Register 3	0x0000_0000
CMR3	PWMA_BA+0x34	R/W	PWM Group A Comparator Register 3	0x0000_0000
PDR3	PWMA_BA+0x38	R	PWM Group A Data Register 3	0x0000_0000
PBCR	PWMA_BA+0x3C	R/W	PWM Group A backward compatible Register	0x0000_0000
PIER	PWMA_BA+0x40	R/W	PWM Group A Interrupt Enable Register	0x0000_0000
PIIR	PWMA_BA+0x44	R/W	PWM Group A Interrupt Indication Register	0x0000_0000
CCR0	PWMA_BA+0x50	R/W	PWM Group A Capture Control Register 0	0x0000_0000
CCR2	PWMA_BA+0x54	R/W	PWM Group A Capture Control Register 2	0x0000_0000
CRLR0	PWMA_BA+0x58	R	PWM Group A Capture Rising Latch Register (Channel 0)	0x0000_0000
CFLR0	PWMA_BA+0x5C	R	PWM Group A Capture Falling Latch Register (Channel 0)	0x0000_0000
CRLR1	PWMA_BA+0x60	R	PWM Group A Capture Rising Latch Register (Channel 1)	0x0000_0000
CFLR1	PWMA_BA+0x64	R	PWM Group A Capture Falling Latch Register (Channel 1)	0x0000_0000
CRLR2	PWMA_BA+0x68	R	PWM Group A Capture Rising Latch Register (Channel 2)	0x0000_0000

CFLR2	PWMA_BA+0x6C	R	PWM Group A Capture Falling Latch Register (Channel 2)	0x0000_0000
CRLR3	PWMA_BA+0x70	R	PWM Group A Capture Rising Latch Register (Channel 3)	0x0000_0000
CFLR3	PWMA_BA+0x74	R	PWM Group A 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
POE	PWMA_BA+0x7C	R/W	PWM Group A Output Enable for channel 0~3	0x0000_0000
TCON	PWMA_BA+0x80	R/W	PWM Group A Trigger Control Register	0x0000_0000
TSTATUS	PWMA_BA+0x84	R/W	PWM Group A Trigger Status Register	0x0000_0000
SYNCBUSY0	PWMA_BA+0x88	R	PWM Group A Synchronous Busy Status Register	0x0000_0000
SYNCBUSY1	PWMA_BA+0x8C	R	PWM Group A Synchronous Busy Status Register	0x0000_0000
SYNCBUSY2	PWMA_BA+0x90	R	PWM Group A Synchronous Busy Status Register	0x0000_0000
SYNCBUSY3	PWMA_BA+0x94	R	PWM Group A Synchronous Busy Status Register	0x0000_0000
CAPPDMACTL	PWMA_BA+0xC0	R/W	PWM Group A PDMA control register	0x0000_0000
CAP0PDMA	PWMA_BA+0xC4	R	PWM Group A channel0 data register	0x0000_0000
CAP1PDMA	PWMA_BA+0xC8	R	PWM Group A channel1 data register	0x0000_0000
CAP2PDMA	PWMA_BA+0xCC	R	PWM Group A channel2 data register	0x0000_0000
CAP3PDMA	PWMA_BA+0xD0	R	PWM Group A channel3 data register	0x0000_0000

5.9.6 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

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	Description	
[31:24]	**DZI23**	**Dead-zone Interval for Pair of Channel2 and Channel3 (PWM2 and PWM3 Pair for PWM Group A)** These 8 bits 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)** These 8 bits 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)** Clock input is divided by (CP23 + 1) before it is fed to the corresponding PWM-timer. If CP23 = 0, the clock prescaler 2 output clock will be stopped. Thus the corresponding PWM timer will also be stopped.
[7:0]	**CP01**	**Clock Prescaler 0 (PWM-timer 0 / 1 for Group A)** Clock input is divided by (CP01 + 1) before it is fed to the corresponding PWM-timer. If CP01 = 0, the clock prescaler 0 output clock will be stopped. Thus the corresponding PWM timer will also be stopped.

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

31	30	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">Description</th></tr>
<tr><td>[31:15]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[14:12]</td><td><b>CSR3</b></td><td><b>PWM Timer 3 Clock Source Selection</b> (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><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[10:8]</td><td><b>CSR2</b></td><td><b>PWM Timer 2 Clock Source Selection (PWM Timer 2 for Group A and PWM Timer 6 for Group B)</b>
Select clock input for PWM timer.
(Table is the same as CSR3.)</td></tr>
<tr><td>[7]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[6:4]</td><td><b>CSR1</b></td><td><b>PWM Timer 1 Clock Source Selection (PWM Timer 1 for Group A and PWM Timer 5 for Group B)</b>
Select clock input for PWM timer.
(Table is the same as CSR3.)</td></tr>
<tr><td>[3]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
</table>

[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

31	**30**	**29**	**28**	**27**	**26**	**25**	**24**
PWMTYPE23	**PWMTYPE01**	**Reserved**		**CH3MOD**	**CH3INV**	**CH3PINV**	**CH3EN**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved				**CH2MOD**	**CH2INV**	**CH2PINV**	**CH2EN**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**CH1MOD**	**CH1INV**	**CH1PINV**	**CH1EN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**DZEN23**	**DZEN01**	**CH0MOD**	**CH0INV**	**CH0PINV**	**CH0EN**

Bits	Description	
[31]	**PWMTYPE23**	**PWM23 Aligned Type Selection Bit (PWM2 and PWM3 Pair for PWM Group A)** 0 = Edge-aligned type. 1 = Center-aligned type.
[30]	**PWMTYPE01**	**PWM01 Aligned Type Selection Bit (PWM0 and PWM1 Pair for PWM Group A)** 0 = Edge-aligned type. 1 = Center-aligned type.
[30:28]	**Reserved**	Reserved
[27]	**CH3MOD**	**PWM-Timer 3 Auto-reload/One-Shot Mode (PWM Timer 3 for Group A)** 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)** 1 = Inverter Enabled. 0 = Inverter Disabled.
[25]	**CH3PINV**	**PWM-Timer 3 Output Polar Inverse Enable (PWM Timer 3 for Group A)** 1 = PWM3 output polar inverse Enabled. 0 = PWM3 output polar inverse Disabled.
[24]	**CH3EN**	**PWM-Timer 3 Enable (PWM Timer 3 for Group A)** 1 = Corresponding PWM-Timer Start Run Enabled. 0 = Corresponding PWM-Timer Running Stopped.
[23:20]	**Reserved**	Reserved

[19]	**CH2MOD**	**PWM-Timer 2 Auto-reload/One-Shot Mode (PWM Timer 2 for Group A)** 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)** 1 = Inverter Enabled. 0 = Inverter Disabled.
[17]	**CH2PINV**	**PWM-Timer 2 Output Polar Inverse Enable (PWM Timer 2 for Group A)** 1 = PWM2 output polar inverse Enabled. 0 = PWM2 output polar inverse Disabled.
[16]	**CH2EN**	**PWM-Timer 2 Enable** (**PWM Timer 2 for Group A)** 1 = Corresponding PWM-Timer Start Run Enabled. 0 = Corresponding PWM-Timer Running Stopped.
[15:12]	**Reserved**	Reserved
[11]	**CH1MOD**	**PWM-Timer 1 Auto-reload/One-Shot Mode (PWM Timer 1 for Group A)** 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 Enabled. 0 = Inverter Disabled.
[9]	**CH1PINV**	**PWM-Timer 1 Output Polar Inverse Enable (PWM Timer 1 for Group A)** 1 = PWM1 output polar inverse Enabled. 0 = PWM1 output polar inverse Disabled.
[8]	**CH1EN**	**PWM-Timer 1 Enable (PWM Timer 1 for Group A and PWM Timer 5 for Group B)** 1 = Corresponding PWM-Timer Start Run Enabled. 0 = Corresponding PWM-Timer Running Stopped.
[7:6]	**Reserved**	Reserved
[5]	**DZEN23**	**Dead-Zone 2 Generator Enable (PWM2 and PWM3 Pair for PWM Group A)** 1 = Enabled. 0 = Disabled. **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)** 1 = Enabled. 0 = Disabled. **Note:** When dead-zone generator is enabled, the pair of PWM0 and PWM1 becomes a complementary pair for PWM group A.
[3]	**CH0MOD**	**PWM-Timer 0 Auto-reload/One-Shot Mode (PWM Timer 0 for Group A)** 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)** 1 = Inverter Enabled. 0 = Inverter Disabled.
[1]	**CH0PINV**	**PWM-Timer 0 Output Polar Inverse Enable (PWM Timer 0 for Group A)** 1 = PWM0 output polar inverse Enabled. 0 = PWM0 output polar inverse Disabled.
[0]	**CH0EN**	**PWM-Timer 0 Enable (PWM timer 0 for group A)** 1 = Corresponding PWM-Timer Start Run Enabled. 0 = Corresponding PWM-Timer Running Stopped.

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
CNR1	PWMA_BA+0x18	R/W	PWM Group A Counter Register 1	0x0000_0000
CNR2	PWMA_BA+0x24	R/W	PWM Group A Counter Register 2	0x0000_0000
CNR3	PWMA_BA+0x30	R/W	PWM Group A 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	Description	
[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, depending on the selected PWM channel. For Edge-aligned mode: ● 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 For Center-aligned mode: ● Duty ratio = [(2 x CMR) + 1]/[2 x (CNR+1)]. ● CMR > CNR: PWM output is always high. ● CMR <= CNR: PWM low width = 2 x (CNR-CMR) + 1 unit; PWM high width = (2 x CMR) + 1 unit. ● CMR = 0: PWM low width = 2 x CNR + 1 unit; PWM high width = 1 unit (Unit = one PWM clock cycle) **Note:** Any write to CNR will take effect in the next PWM cycle.

		Note: When PWM operating at center-aligned type, CNR value should be set between 0x0000 to 0xFFFE. If CNR equal to 0xFFFF, the PWM will work unpredictable. **Note:** When CNR value is set to 0, PWM output is always high.

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
CMR1	PWMA_BA+0x1C	R/W	PWM Group A Comparator Register 1	0x0000_0000
CMR2	PWMA_BA+0x28	R/W	PWM Group A Comparator Register 2	0x0000_0000
CMR3	PWMA_BA+0x34	R/W	PWM Group A 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	Description	
[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, depending on the selected PWM channel. For Edge-aligned mode: ● 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 For Center-aligned mode: ● Duty ratio = [(2 x CMR) + 1]/[2 x (CNR+1)]. ● CMR > CNR: PWM output is always high. ● CMR <= CNR: PWM low width = 2 x (CNR-CMR) + 1 unit; PWM high width = (2 x CMR) + 1 unit. ● CMR = 0: PWM low width = 2 x CNR + 1 unit; PWM high width = 1 unit (Unit = one PWM clock cycle) **Note:** Any write to CNR will take effect in the 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
PDR1	PWMA_BA0+0x20	R	PWM Group A Data Register 1	0x0000_0000
PDR2	PWMA_BA0+0x2C	R	PWM Group A Data Register 2	0x0000_0000
PDR3	PWMA_BA0+0x38	R	PWM Group A 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	Description	
[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 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

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved						INTTYPE23	INTTYPE01
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				PWMDIE3	PWMDIE2	PWMDIE1	PWMDIE0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				PWMIE3	PWMIE2	PWMIE1	PWMIE0

Bits	Description	
[31:18]	**Reserved**	Reserved
[17]	**INTTYPE23**	**PWM12 Interrupt Type Selection Bit (PWM2 and PWM3 Pair for PWM Group A, PWM6 and PWM7 Pair for PWM Group B)** 1 = PWMIFn will be set if PWM counter matches CNRn register. 0 = PWMIFn will be set if PWM counter underflow. **Note:** This bit is effective when PWM in central align mode only.
[16]	**INTTYPE01**	**PWM01 Interrupt Type Selection Bit (PWM0 and PWM1 Pair for PWM Group A)** 1 = PWMIFn will be set if PWM counter matches CNRn register. 0 = PWMIFn will be set if PWM counter underflow. **Note:** This bit is effective when PWM in central align mode only.
[11]	**PWMDIE3**	**PWM Channel 3 Duty Interrupt Enable** 1 = Enabled. 0 = Disabled.
[10]	**PWMDIE2**	**PWM Channel 2 Duty Interrupt Enable** 1 = Enabled. 0 = Disabled.
[9]	**PWMDIE1**	**PWM Channel 1 Duty Interrupt Enable** 1 = Enabled. 0 = Disabled.

[8]	**PWMDIE0**	**PWM Channel 0 Duty Interrupt Enable** 1 = Enabled. 0 = Disabled.
[7:4]	**Reserved**	**Reserved**
[3]	**PWMIE3**	**PWM Channel 3 Interrupt Enable** 1 = Enabled. 0 = Disabled.
[2]	**PWMIE2**	**PWM Channel 2 Interrupt Enable** 1 = Enabled. 0 = Disabled.
[1]	**PWMIE1**	**PWM Channel 1 Interrupt Enable** 1 = Enabled. 0 = Disabled.
[0]	**PWMIE0**	**PWM Channel 0 Interrupt Enable** 1 = Enabled. 0 = Disabled.

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

31	30	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				PWMDIF3	PWMDIF2	PWMDIF1	PWMDIF0
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				PWMIF3	PWMIF2	PWMIF1	PWMIF0

Bits	Description	
[31:12]	**Reserved**	Reserved
[11]	**PWMDIF3**	**PWM channel 3 Duty Interrupt Flag** Flag is set by hardware when channel 3 PWM counter down count and reaches CMR3, software can clear this bit by writing a one to it. **Note:** If CMR is equal to CNR, this flag is not working
[10]	**PWMDIF2**	**PWM channel 2 Duty Interrupt Flag** Flag is set by hardware when channel 2 PWM counter down count and reaches CMR2, software can clear this bit by writing a one to it. **Note:** If CMR is equal to CNR, this flag is not working
[9]	**PWMDIF1**	**PWM channel 1 Duty Interrupt Flag** Flag is set by hardware when channel 1 PWM counter down count and reaches CMR1, software can clear this bit by writing a one to it. **Note:** If CMR is equal to CNR, this flag is not working
[8]	**PWMDIF0**	**PWM channel 0 Duty Interrupt Flag** Flag is set by hardware when channel 0 PWM counter down count and reaches CMR0, software can clear this bit by writing a one to it. **Note:** If CMR is equal to CNR, this flag is not working
[7:4]	**Reserved**	Reserved
[3]	**PWMIF3**	**PWM channel 3 Interrupt Status** This bit is set by hardware when PWM3 counter reaches the requirement of interrupt (depending on INTTYPE23 bit of PIER register) 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 counter reaches the requirement of interrupt (depending on INTTYPE23 bit of PIER register) if PWM2 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 counter reaches the requirement of interrupt (depending on INTTYPE01 bit of PIER register) if PWM1 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 counter reaches the requirement of interrupt (depending on INTTYPE01 bit of PIER register) if PWM0 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 the 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

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	Description	
[31:24]	**Reserved**	Reserved
[23]	**CFLRI1**	**CFLR1 Latched Indicator Bit** When PWM group input channel 1 has a falling transition, CFLR1 is 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 is 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 = Capture function on PWM group channel 1 Enabled. 0 = Capture function on PWM group channel 1 Disabled. When Enabled, Capture latched the PWM-counter and saved to CRLR (Rising latch) and CFLR (Falling latch). When Disabled, Capture does not update CRLR and CFLR, and disable PWM group channel 1 Interrupt.

[18]	**CFL_IE1**	**Channel 1 Falling Latch Interrupt Enable** 1 = Falling latch interrupt Enabled. 0 = Falling latch interrupt Disabled. When Enabled, if Capture detects PWM group channel 1 has falling transition, capture issues an Interrupt.
[17]	**CRL_IE1**	**Channel 1 Rising Latch Interrupt Enable** 1 = Rising latch interrupt Enabled. 0 = Rising latch interrupt Disabled. When Enable, if Capture detects PWM group channel 1 has rising transition, capture issues an Interrupt.
[16]	**INV1**	**Channel 1 Inverter Enable** 1 = Inverter Enabled. Reverse the input signal from GPIO before fed to Capture timer. 0 = Inverter Disabled.
[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 = Capture function on PWM group channel 0 Enabled. 0 = Capture function on PWM group channel 0 Disabled. When Enabled, Capture latched the PWM-counter value and saved to CRLR (Rising latch) and CFLR (Falling latch). When Disabled, capture does not update CRLR and CFLR, and disable PWM group channel 0 Interrupt.

[2]	**CFL_IE0**	**Channel 0 Falling Latch Interrupt Enable** 1 = Falling latch interrupt Enabled. 0 = Falling latch interrupt Disabled. When Enabled, if capture detects PWM group channel 0 has falling transition, capture issues an Interrupt.
[1]	**CRL_IE0**	**Channel 0 Rising Latch Interrupt Enable** 1 = Rising latch interrupt Enabled. 0 = Rising latch interrupt Disabled. When Enabled, if capture detects PWM group channel 0 has rising transition, capture issues an Interrupt.
[0]	**INV0**	**Channel 0 Inverter Enable** 1 = Inverter Enabled. Reverse the input signal from GPIO before fed to Capture timer 0 = Inverter Disabled

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

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	Description	
[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 = Capture function on PWM group channel 3 Enabled. 0 = Capture function on PWM group channel 3 Disabled. When Enabled, capture latched the PWM-counter and saved to CRLR (Rising latch) and CFLR (Falling latch). When Disabled, capture does not update CRLR and CFLR, and disable PWM group channel 3 Interrupt.
[18]	**CFL_IE3**	**Channel 3 Falling Latch Interrupt Enable** 1 = Falling latch interrupt Enabled. 0 = Falling latch interrupt Disabled. When Enabled, if capture detects PWM group channel 3 has falling transition, capture issues an Interrupt.
[17]	**CRL_IE3**	**Channel 3 Rising Latch Interrupt Enable** 1 = Rising latch interrupt Enabled. 0 = Rising latch interrupt Disabled. When Enabled, if Capture detects PWM group channel 3 has rising transition, capture issues an Interrupt.
[16]	**INV3**	**Channel 3 Inverter Enable** 1 = Inverter Enabled. Reverse the input signal from GPIO before fed to Capture timer 0 = Inverter Disabled.
[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 = Capture function on PWM group channel 2 Enabled. 0 = Capture function on PWM group channel 2 Disabled. When Enabled, capture latched the PWM-counter value and saved to CRLR (Rising latch) and CFLR (Falling latch). When Disabled, capture does not update CRLR and CFLR, and disable PWM group channel 2 Interrupt.
[2]	**CFL_IE2**	**Channel 2 Falling Latch Interrupt Enable** 1 = Falling latch interrupt Enabled. 0 = Falling latch interrupt Disabled. When Enabled, if capture detects PWM group channel 2 has falling transition, capture issues an Interrupt.
[1]	**CRL_IE2**	**Channel 2 Rising Latch Interrupt Enable** 1 = Rising latch interrupt Enabled. 0 = Rising latch interrupt Disabled. When Enabled, if Capture detects PWM group channel 2 has rising transition, Capture issues an Interrupt.
[0]	**INV2**	**Channel 2 Inverter Enable** 1 = Inverter Enabled. Reverse the input signal from GPIO before fed to Capture timer 0 = Inverter Disabled.

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
CRLR1	PWMA_BA+0x60	R	PWM Group A Capture Rising Latch Register (channel 1)	0x0000_0000
CRLR2	PWMA_BA+0x68	R	PWM Group A Capture Rising Latch Register (channel 2)	0x0000_0000
CRLR3	PWMA_BA+0x70	R	PWM Group A 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	Description	
[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
CFLR1	PWMA_BA+0x64	R	PWM Group A Capture Falling Latch Register (channel 1)	0x0000_0000
CFLR2	PWMA_BA+0x6C	R	PWM Group A Capture Falling Latch Register (channel 2)	0x0000_0000
CFLR3	PWMA_BA+0x74	R	PWM Group A 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	Description	
[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

31	30	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	Description	
[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 = Disabled (PWMx multi-function pin input does not affect input capture function.) 1 = Enabled (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

31	30	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	Description	
[3]	**PWM3**	**Channel 3 Output Enable Register** 1 = PWM channel 3 output to pin Enabled. 0 = PWM channel 3 output to pin Disabled. **Note:** The corresponding GPIO pin must also be switched to PWM function.
[2]	**PWM2**	**Channel 2 Output Enable Register** 1 = PWM channel 2 output to pin Enabled. 0 = PWM channel 2 output to pin Disabled. **Note:** The corresponding GPIO pin must also be switched to PWM function.
[1]	**PWM1**	**Channel 1 Output Enable Register** 1 = PWM channel 1 output to pin Enabled. 0 = PWM channel 1 output to pin Disabled. **Note:** The corresponding GPIO pin must also be switched to PWM function.
[0]	**PWM0**	**Channel 0 Output Enable Register** 1 = PWM channel 0 output to pin Enabled. 0 = PWM channel 0 output to pin Disabled. **Note:** The corresponding GPIO pin must also be switched to PWM function.

PWM Trigger Control Register (TCON)

Register	Offset	R/W	Description	Reset Value
TCON	PWMA_BA+0x80	R/W	PWM Group A Trigger Control 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				**PWM3TEN**	**PWM2TEN**	**PWM1TEN**	**PWM0TEN**

Bits	Description	
[3]	**PWM3TEN**	**Channel 3 Center-Aligned Trigger Enable Register** 1 = PWM channel 3 trigger ADC function Enabled. 0 = PWM channel 3 trigger ADC function Disabled. PWM can trigger ADC to start conversion when PWM counter counts up to CNR if this bit is set to 1. **Note:** This function is only supported when PWM operating in Center-aligned mode.
[2]	**PWM2TEN**	**Channel 2 Center-Aligned Trigger Enable Register** 1 = PWM channel 2 trigger ADC function Enabled. 0 = PWM channel 2 trigger ADC function Disabled. PWM can trigger ADC to start conversion when PWM counter counts up to CNR if this bit is set to 1. **Note:** This function is only supported when PWM operating in Center-aligned mode.
[1]	**PWM1TEN**	**Channel 1 Center-Aligned Trigger Enable Register** 1 = PWM channel 1 trigger ADC function Enabled. 0 = PWM channel 1 trigger ADC function Disabled. PWM can trigger ADC to start conversion when PWM counter counts up to CNR if this bit is set to 1. **Note:** This function is only supported when PWM operating in Center-aligned mode
[0]	**PWM0TEN**	**Channel 0 Center-Aligned Trigger Enable Register** 1 = PWM channel 0 trigger ADC function Enabled. 0 = PWM channel 0 trigger ADC function Disabled. PWM can trigger ADC to start conversion when PWM counter counts up to CNR

		if this bit is set to 1. **Note:** This function is only supported when PWM operating in Center-aligned mode.

PWM Trigger Status Register (TSTATUS)

Register	Offset	R/W	Description	Reset Value
TSTATUS	PWMA_BA+0x84	R/W	PWM Group A Trigger 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				PWM3TF	PWM2TF	PWM1TF	PWM0TF

Bits	Description	
[3]	**PWM3TF**	**Channel 3 Center-aligned Trigger Flag** For Center-aligned operating mode, this bit is set to 1 by hardware when PWM counter up count to CNR if PWM3TEN bit is set to 1. After this bit is set to 1, ADC will start conversion if ADC triggered source is selected by PWM. Software can write 1 to clear this bit.
[2]	**PWM2TF**	**Channel 2 Center-aligned Trigger Flag** For Center-aligned operating mode, this bit is set to 1 by hardware when PWM counter up count to CNR if PWM2TEN bit is set to 1. After this bit is set to 1, ADC will start conversion if ADC triggered source is selected by PWM. Software can write 1 to clear this bit.
[1]	**PWM1TF**	**Channel 1 Center-aligned Trigger Flag** For Center-aligned operating mode, this bit is set to 1 by hardware when PWM counter up count to CNR if PWM1TEN bit is set to 1. After this bit is set to 1, ADC will start conversion if ADC triggered source is selected by PWM. Software can write 1 to clear this bit.
[0]	**PWM0TF**	**Channel 0 Center-aligned Trigger Flag** For Center-aligned operating mode, this bit is set to 1 by hardware when PWM counter up count to CNR if PWM0TEN bit is set to 1. After this bit is set to 1, ADC will start conversion if ADC triggered source is selected by PWM. Software can write 1 to clear this bit.

PWM0 Synchronous Busy Status Register (SYNCBUSY0)

Register	Offset	R/W	Description	Reset Value
SYNCBUSY0	PWMA_BA+0x88	R	PWM0 Group A Synchronous Busy 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							S_BUSY

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**S_BUSY**	**PWM Synchronous Busy** When software writes CNR0/CMR0/PPR or switch PWM0 operation mode (PCR[3]), PWM will have a busy time to update these values completely because PWM clock may be different from system clock domain. Software needs to check this busy status before writes CNR0/CMR0/PPR or switch PWM0 operation mode (PCR[3]) to make sure previous setting has been update completely. This bit will be set when software write CNR0/CMR0/PPR or switch PWM0 operation mode (PCR[3]) and will be cleared by hardware automatically when PWM update these value completely.

PWM1 Synchronous Busy Status Register (SYNCBUSY1)

Register	Offset	R/W	Description	Reset Value
SYNCBUSY1	PWMA_BA+0x8C	R	PWM1 Group A Synchronous Busy 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							**S_BUSY**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**S_BUSY**	**PWM Synchronous Busy** When software writes CNR1/CMR1/PPR or switches PWM1 operation mode (PCR[11]), PWM will have a busy time to update these values completely because PWM clock may be different from system clock domain. Software needs to check this busy status before writing CNR1/CMR1/PPR or switching PWM1 operation mode (PCR[11]) to make sure the previous setting has been updated completely. This bit will be set when software write CNR1/CMR1/PPR or switch PWM1 operation mode (PCR[11]) and will be cleared by hardware automatically when PWM updates these value completely.

PWM2 Synchronous Busy Status Register (SYNCBUSY2)

Register	Offset	R/W	Description	Reset Value
SYNCBUSY2	PWMA_BA+0x90	R	PWM2 Group A Synchronous Busy 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							**S_BUSY**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**S_BUSY**	**PWM Synchronous Busy** When software writes CNR2/CMR2/PPR or switch PWM2 operation mode (PCR[19]), PWM will have a busy time to update these values completely because PWM clock may be different from system clock domain. Software needs to check this busy status before writing CNR2/CMR2/PPR or switching PWM2 operation mode (PCR[19]) to make sure the previous setting has been updated completely. This bit will be set when software write CNR2/CMR2/PPR or switch PWM2 operation mode (PCR[19]) and will be cleared by hardware automatically when PWM updates these value completely.

PWM3 Synchronous Busy Status Register (SYNCBUSY3)

Register	Offset	R/W	Description	Reset Value
SYNCBUSY3	PWMA_BA+0x94	R	PWM3 Group A Synchronous Busy 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							**S_BUSY**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**S_BUSY**	**PWM Synchronous Busy** When software writes CNR3/CMR3/PPR or switch PWM3 operation mode (PCR[27]), PWM will have a busy time to update these values completely because PWM clock may be different from system clock domain. Software needs to check this busy status before writing CNR3/CMR3/PPR or switching PWM3 operation mode (PCR[27]) to make sure the previous setting has been updated completely. This bit will be set when software write CNR3/CMR3/PPR or switch PWM3 operation mode (PCR[27]) and will be cleared by hardware automatically when PWM updates these value completely.

PWM PDMA control Register (CAPPDMACTL)

Register	Offset	R/W	Description	Reset Value
CAPPDMACTL	PWMA_BA+0xC0	R/W	PWM Group A Trigger Status Register	0x0000_0000

31	30	29	28	27	26	25	24
				CAP3RFORDER	**CAP3PDMAMOD**		**CAP3PDMAEN**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
				CAP2RFORDER	**CAP2PDMAMOD**		**CAP2PDMAEN**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
				CAP1RFORDER	**CAP1PDMAMOD**		**CAP1PDMAEN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
				CAP0RFORDER	**CAP0PDMAMOD**		**CAP0PDMAEN**

Bits	Description	
[27]	**CAP3RFORDER**	**Capture Channel 3 Rising/Falling Order** **Set this bit to determine whether the CRLR3 or CFLR3 is the first captured data transferred to memory through PDMA when CAP3PDMAMOD =11** 1 = CRLR3 is the first captured data to memory. 0 = CFLR3 is the first captured data to memory.
[26:25]	**CAP3PDMAMOD**	**Select CRLR3 or CFLR3 to do PDMA Transfer** 00 = Reserved 01 = CRLR3 10 = CFLR3 11 = Both CRLR3 and CFLR3
[24]	**CAP3PDMAEN**	**Channel 3 PDMA enable** 1 = Channel 3 PDMA function Enabled for the channel 3 captured data and transfer to memory. 0 = Channel 3 PDMA function Disabled.
[19]	**CAP2RFORDER**	**Capture channel 2 Rising/Falling Order** **Set this bit to determine whether the CRLR2 or CFLR2 is the first captured data transferred to memory through PDMA when CAP2PDMAMOD =11** 1 = CRLR2 is the first captured data to memory. 0 = CFLR2 is the first captured data to memory.

[18:17]	**CAP2PDMAMOD**	**Select CRLR2 or CFLR2 to do PDMA Transfer** 00 = Reserved 01 = CRLR2 10 = CFLR2 11 = Both CRLR2 and CFLR2
[16]	**CAP2PDMAEN**	**Channel 2 PDMA Enable** 1 = Channel 2 PDMA function Enabled for the channel 2 captured data and transfer to memory. 0 = Channel 2 PDMA function Disabled.
[11]	**CAP1RFORDER**	**Capture channel 1 Rising/Falling Order** **Set this bit to determine whether the CRLR1 or CFLR1 is the first captured data transferred to memory through PDMA when CAP1PDMAMOD = 11** 1 = CRLR1 is the first captured data to memory 0 = CFLR1 is the first captured data to memory
[10:9]	**CAP1PDMAMOD**	**Select CRLR1 or CFLR1 to Transfer PDMA** 00 = Reserved 01 = CRLR1 10 = CFLR1 11 = both CRLR1 and CFLR1
[8]	**CAP1PDMAEN**	**Channel 1 PDMA Enable** 1 = Channel 1 PDMA function Enabled for the channel 1 captured data and transfer to memory. 0 = Channel 1 PDMA function Disabled.
[3]	**CAP0RFORDER**	**Set this bit to determine whether the CRLR0 or CFLR0 is the first captured data transferred to memory through PDMA when CAP0PDMAMOD =11** 1 = CRLR0 is the first captured data to memory. 0 = CFLR0 is the first captured data to memory.
[2:1]	**CAP0PDMAMOD**	**Select CRLR0 or CFLR0 to Transfer PDMA** 00 = Reserved 01 = CRLR0 10 = CFLR0 11 = Both CRLR0 and CFLR0
[0]	**CAP0PDMAEN**	**Channel 0 PDMA Enable** 1 = Channel 0 PDMA function Enabled for the channel 0 captured data and transfer to memory. 0 = Channel 0 PDMA function Disabled.

PWM PDMA DATA Register0 (CAP0PDMA)

Register	Offset	R/W	Description	Reset Value
CAP0PDMA	PWMA_BA+0xC4	R	PWM Group A PDMA channel 0 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**
CAP0RFPDMA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CAP0RFPDMA							

Bits	Description	
[15:0]	**CAP0RFPDMA[15:0]**	**PDMA data register for channel 0** it is the capturing value(CFLR0/CRLR0) for channel 0

PWM PDMA DATA Register1 (CAP1PDMA)

Register	Offset	R/W	Description	Reset Value
CAP1PDMA	PWMA_BA+0xC8	R	PWM Group A PDMA channel 1 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**
CAP1RFPDMA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CAP1RFPDMA							

Bits	Description	
[15:0]	**CAP1RFPDMA[15:0]**	**PDMA data register for channel 1** it is the capturing value(CFLR1/CRLR1) for channel 1

PWM PDMA DATA Register2 (CAP2PDMA)

Register	Offset	R/W	Description	Reset Value
CAP2PDMA	PWMA_BA+0xCC	R	PWM Group A PDMA channel 2 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**
CAP2RFPDMA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CAP2RFPDMA							

Bits	Description	
[15:0]	**CAP2RFPDMA[15:0]**	**PDMA data register for channel 2** it is the capturing value(CFLR2/CRLR2) for channel 2

PWM PDMA DATA Register2 (CAP2PDMA)

Register	Offset	R/W	Description	Reset Value
CAP3PDMA	PWMA_BA+0xD0	R	PWM Group A PDMA channel 3 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**
CAP3RFPDMA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CAP3RFPDMA							

Bits	Description	
[15:0]	**CAP3RFPDMA[15:0]**	**PDMA data register for channel 3** it is the capturing value(CFLR3/CRLR3) for channel 3

5.10 Serial Peripheral Interface (SPI)

5.10.1 Overview

The Serial Peripheral Interface (SPI) is a synchronous serial data communication protocol that operates in full duplex mode. Devices communicate in Master/Slave mode with the 4-wire bi-direction interface. This chip contains up to three sets of SPI controllers performing a serial-to-parallel conversion on data received from a peripheral device, and a parallel-to-serial conversion on data transmitted to a peripheral device. Each set of SPI controller can be configured as a master or a slave device.

The SPI controller supports the variable serial clock function for special applications and 2-bit Transfer mode to connect 2 off-chip slave devices at the same time. This controller also supports the PDMA function to access the data buffer and also supports Dual I/O Transfer mode.

5.10.2 Features

- Up to three sets of SPI controller
- Supports Master or Slave mode operation
- Supports 2-bit Transfer mode
- Supports Dual I/O Transfer mode
- Configurable bit length of a transfer word from 8 to 32-bit
- Provides separate 8-layer depth transmit and receive FIFO buffers
- Supports MSB first or LSB first transfer sequence
- Two slave select lines in Master mode
- Supports the byte reorder function
- Supports Byte or Word Suspend mode
- Variable output serial clock frequency in Master mode
- Supports PDMA transfer
- Supports 3-wire, no slave select signal, bi-direction interface

5.10.3 Block Diagram

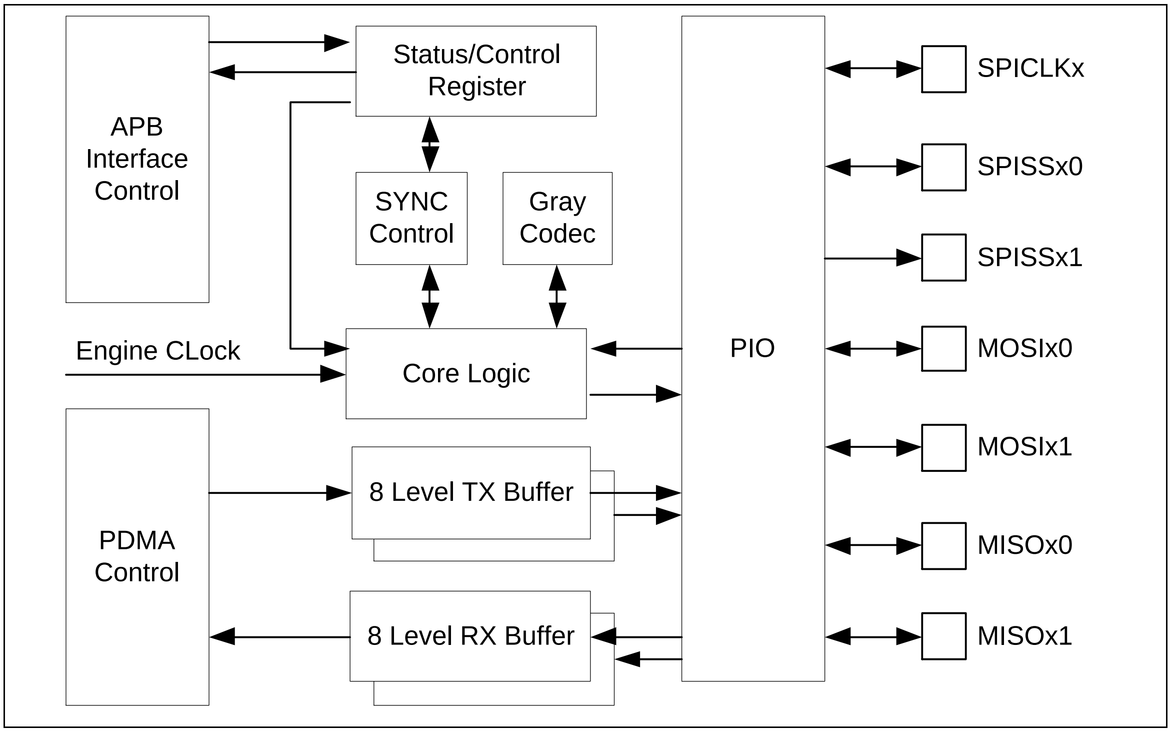


Figure 5-51 SPI Block Diagram

5.10.4 Functional Description

SPI Engine Clock and Serial Clock

The SPI controller needs the SPI engine clock to drive the SPI logic unit to perform the data transfer. The SPI engine clock rate is determined by the settings of clock source, BCn option and clock divisor. The SPIx_S bit of CLKSEL1 register determines the clock source of the SPI engine clock. The clock source can be HCLK or PLL output clock. Set the BCn bit of SPI_CNTRL2 register to 0 for the compatible SPI clock rate calculation of previous products. The DIVIDER setting of SPI_DIVIDER register determines the divisor of the clock rate calculation.

In Master mode, if the variable clock function is disabled, the output frequency of the serial clock output pin is equal to the SPI engine clock rate. In Slave mode, the SPI serial clock is provided by an off-chip master device. The SPI engine clock rate of slave device must be faster than the serial clock rate of the master device connected together. The frequency of SPI engine clock cannot be faster than the APB clock rate regardless of Master or Slave mode.

Master/Slave Mode

The 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 below.

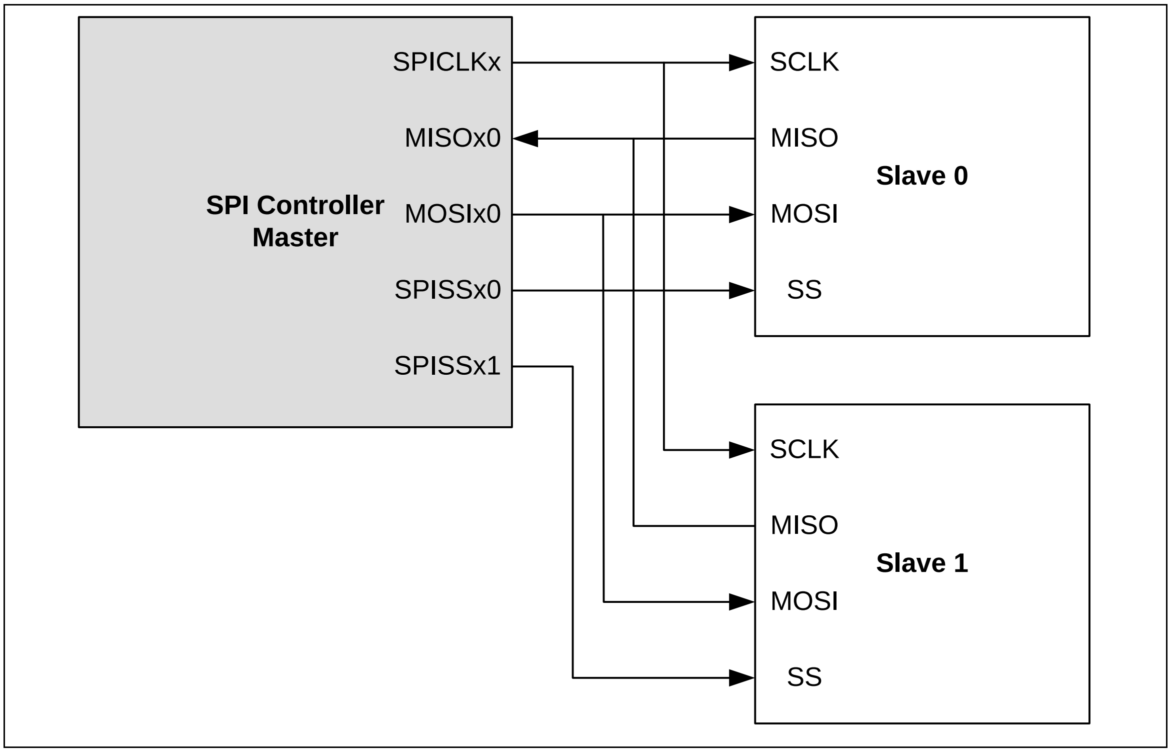


Figure 5-52 SPI Master Mode Application Block Diagram

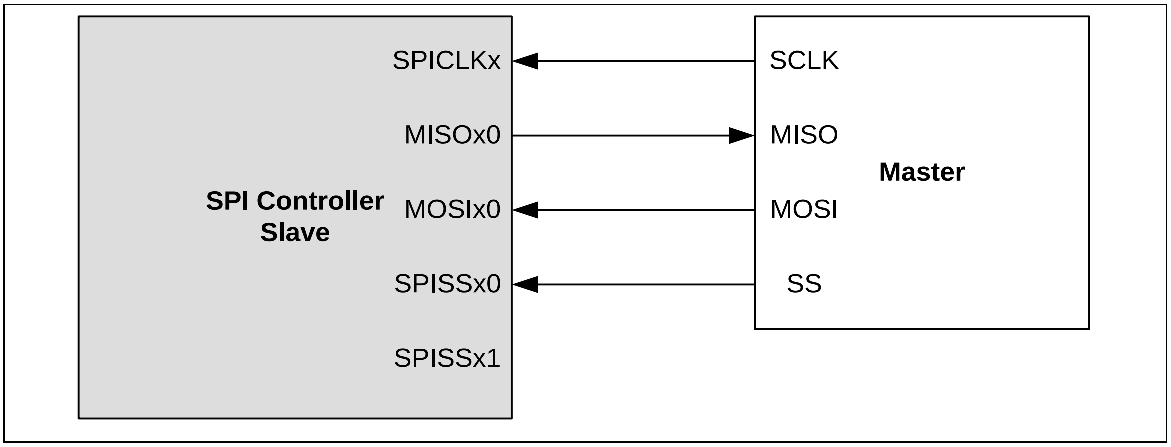


Figure 5-53 SPI Slave Mode Application Block Diagram

Slave Selection

In Master mode, the 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 the SPI controller. In Master/Slave mode, the active state of slave select signal can be programmed to low 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-triggered or edge-triggered. The selection of trigger conditions 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 the received bits among one transaction meets the requirement defined in TX_BIT_LEN.

Level-trigger/Edge-trigger

In Slave mode, the slave select signal can be configured as level-trigger or edge-trigger. For edge-trigger, the data transfer starts from an active edge and ends on an inactive edge. If the master does not send an inactive edge to slave, the transfer procedure will not be completed and the unit transfer interrupt flag of slave will not be set. For level-trigger, the following two conditions will terminate the transfer procedure and the unit transfer interrupt flag of slave will be set. The first condition is that if the number of transferred bits matches the settings of TX_BIT_LEN, the unit transfer interrupt flag of slave will be set. As to the second condition, if the master sets 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 unit transfer interrupt flag will be set. User can read the status of LTRIG_FLAG bit to see if the data has been completely transferred.

Automatic Slave Selection

In Master mode, if the bit AUTOSS (SPI_SSR[3]) is set, the slave select signals will be generated automatically and output to the SPISSx0 and SPISSx1 pins according to whether SSR[0] (SPI_SSR[0]) and SSR[1] (SPI_SSR[1]) are enabled or not. This means that the slave select signals, which are selected in SSR[1:0], will be asserted by the SPI controller when the SPI data transfer 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]).

In Master mode, if the value of SP_CYCLE[3:0] is less than 3 and the AUTOSS is set as 1, the slave select signal will be kept in active state between two successive transactions.

In Slave mode, to recognize the inactive state of the slave select signal, the inactive period of the slave select signal must be larger than or equal to 6 engine clock periods between two successive transactions.

Variable Clock Function

In Master mode, if the VARCLK_EN bit (SPI_CTL[23]) is set to 1, the output of serial clock can be programmed as variable frequency pattern. The serial clock period of each cycle depends on the setting of the SPI_VARCLK register. When the variable clock function is enabled, the TX_BIT_LEN setting must be set as 0x10 to configure the data transfer as 16-bit transfer mode. The VARCLK[31] determines the clock period of the first clock cycle. If VARCLK[31] is 0, the first clock cycle depends on the DIVIDER setting; if it is 1, the first clock cycle depends on the DIVIDER2 setting. Two successive bits in VARCLK[30:1] defines one clock cycle. The bit field VARCLK[30:29] defines the second clock cycle of SPI serial clock of a transaction, and the bit field VARCLK[28:27] defines the third clock cycle and so on. The VARCLK[0] has no meaning. The following figure shows the timing relationship among the serial clock (SPICLK), the VARCLK, the DIVIDER and the DIVIDER2 registers.

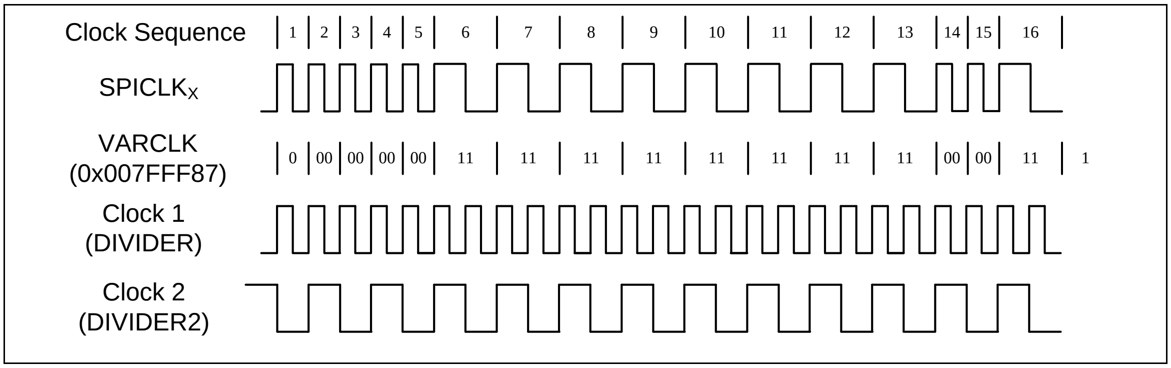


Figure 5-54 Variable Serial Clock Frequency

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. If CLKP = 0, it is idle at low state.

Transmit/Receive Bit Length

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

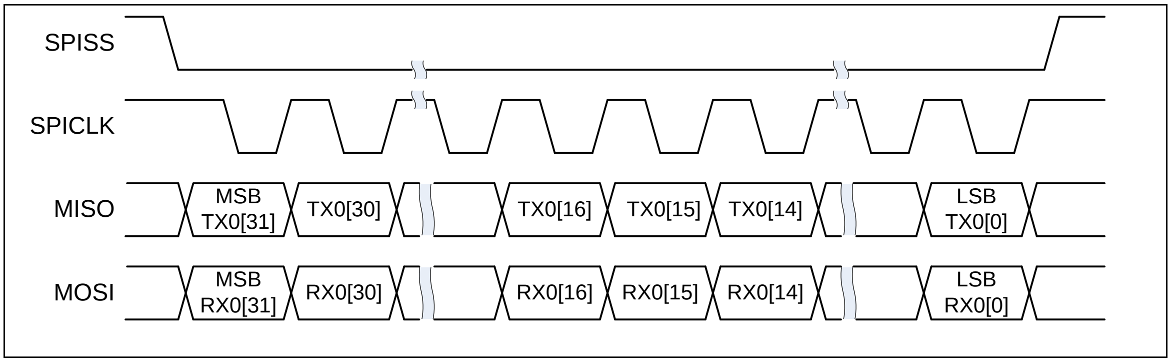


Figure 5-55 32-Bit in One Transaction

LSB/MSB First

The LSB bit (SPI_CNTRL[10]) defines the bit transfer sequence in a transaction. If the LSB bit is set to 1, the transfer sequence is LSB first. The bit 0 will be transferred firstly. If the LSB bit is cleared to 0, the transfer sequence is MSB first.

Transmit Edge

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

Receive Edge

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

Note: The settings of TX_NEG and RX_NEG are mutual exclusive. In other words, do not transmit and receive data at the same clock edge.

Word Suspend

The four bit fields of SP_CYCLE (SPI_CNTRL[15:12]) provide a configurable suspend interval, 0.5 ~ 15.5 serial clock periods, between two successive transaction words in Master mode. The definition of the suspend interval is the interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value of SP_CYCLE is 0x3 (3.5 serial clock cycles). This SP_CYCLE setting will not take effect to the word suspend interval if FIFO mode is disabled by software.

If both the VARCLK_EN, SPI_CNTRL[23], and the FIFO bit, SPI_CNTRL[21], are set as 1, the minimum word suspend period is (6.5 + SP_CYCLE)*SPI serial clock period.

Byte Reorder

When the transfer is set as MSB first (LSB = 0) and the REORDER bit is set to 1, the data stored in the TX buffer and RX buffer will be rearranged in the order as [BYTE0, BYTE1, BYTE2, BYTE3] in 32-bit Transfer mode (TX_BIT_LEN = 0). The sequence of transmitted/received data will be BYTE0, BYTE1, BYTE2, and then BYTE3. If the TX_BIT_LEN is set as 24-bit transfer mode, the data in TX buffer and RX buffer will be rearranged as [unknown byte, BYTE0, BYTE1, BYTE2]. The SPI controller will transmit/receive data with the sequence of BYTE0, BYTE1 and then BYTE2. Each byte will be transmitted/received with MSB first. The rule of 16-bit mode is the same as above. Byte reorder function is only available when TX_BIT_LEN is configured as 16, 24, and 32 bits.

Note: The byte reorder function is not supported when the variable serial clock function is enabled.

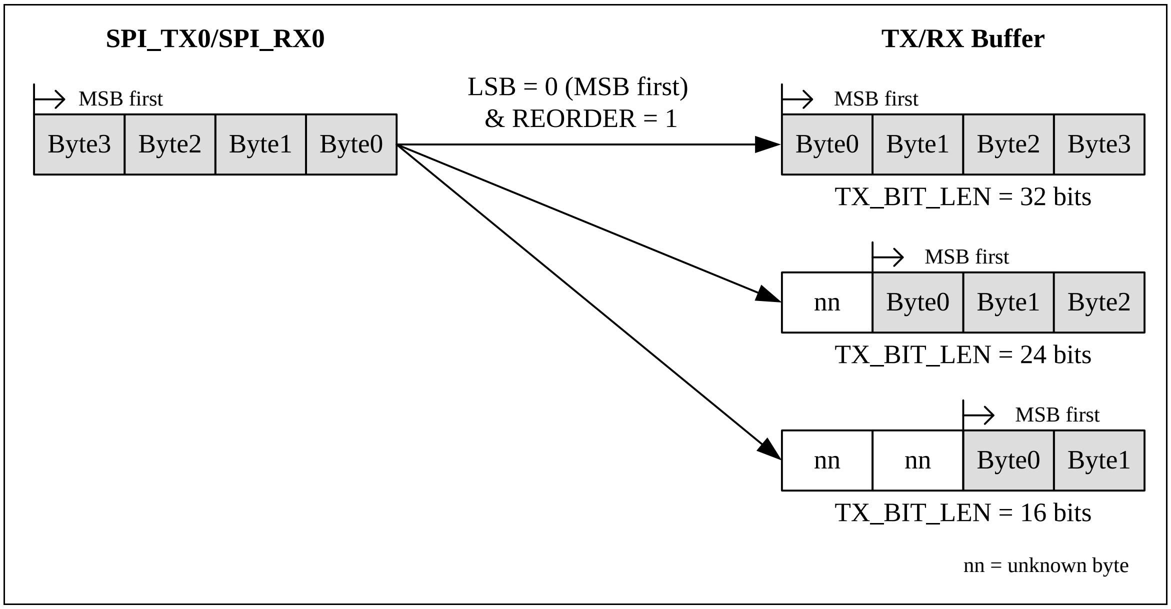


Figure 5-56 Byte Reorder Function

Byte Suspend

In Master mode, if SPI_CNTRL[19] is set to 1, a suspend interval of 0.5 ~ 15.5 serial clock periods will be inserted by hardware between two successive bytes in a transaction word. Both settings of byte suspend interval and word suspend interval are configured in SP_CYCLE.

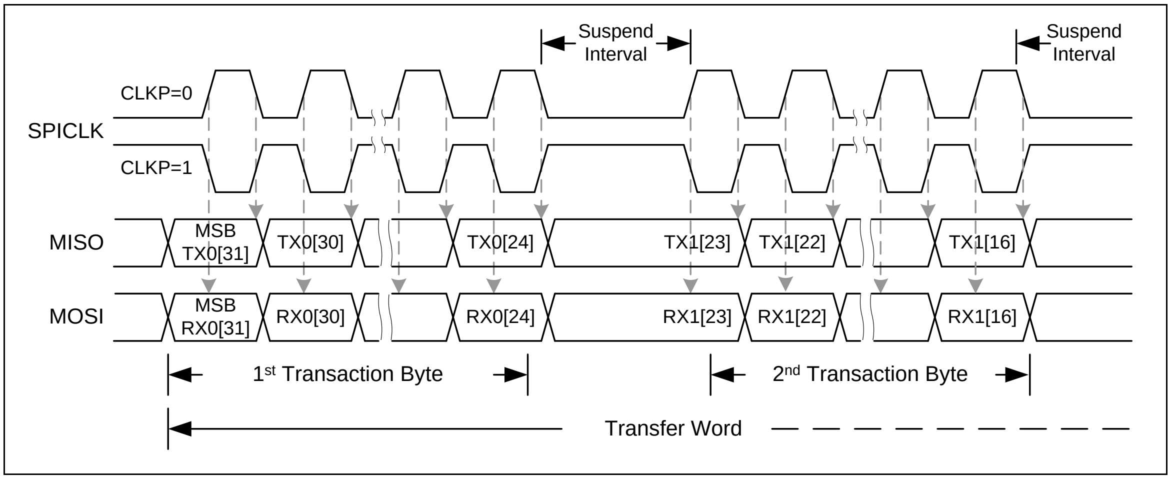


Figure 5-57 Timing Waveform for Byte Suspend

Slave 3-Wire Mode

When the NOSLVSEL bit is set by software to enable the Slave 3-wire mode, the SPI controller can work with no slave select signal in Slave mode. The NOSLVSEL bit only takes effect in Slave mode. Only three pins, SPICLK, SPI_MISO, and SPI_MOSI, are required to communicate with a SPI master. The SPISS pin can be configured as a GPIO. When the NOSLVSEL bit is set to 1, the SPI slave will be ready to transmit/receive data after the GO_BUSY bit is set to 1. In Slave 3-wire mode, the SS_LTRIG, SPI_SSR[4], should be set as 1.

2-Bit Mode

The SPI controller also supports 2-bit Transfer mode when setting the TWOB bit (SPI_CNTRL[22]) to 1. In 2-bit mode, the SPI controller performs full duplex data transfer. In other words, the 2-bit serial data can be transmitted and received simultaneously.

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

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

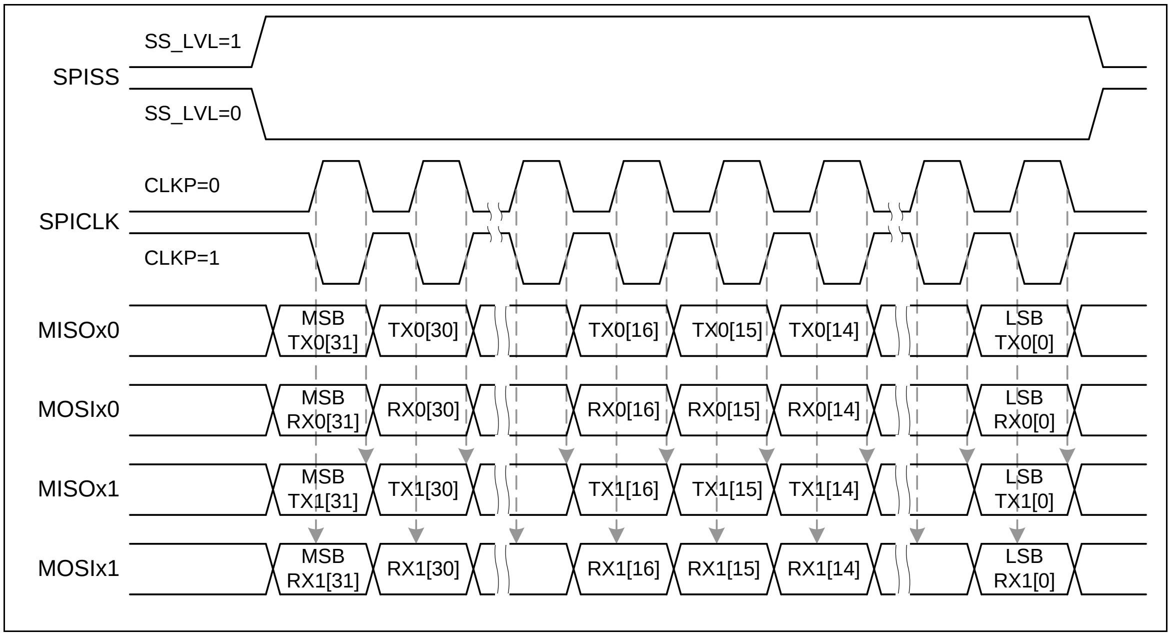


Figure 5-58 2-Bit Mode (Slave Mode)

Dual I/O Mode

The SPI controller also supports dual I/O transfer when setting the DUAL_IO_EN bit (SPI_CNTRL2[13]) to 1. Many general SPI flashes support dual I/O transfer. The DUAL_IO_DIR bit (SPI_CNTRL2[12]) is used to define the direction of the transfer data. When the DUAL_IO_DIR bit is set to 1, the controller will send the data to external device. When the DUAL_IO_DIR bit is set to 0, the controller will read the data from the external device. This function supports 8, 16, 24, and 32-bits of bit length.

The dual I/O mode is not supported when the Slave 3-wire mode or the byte reorder function is enabled.

If both the DUAL_IO_EN and DUAL_IO_DIR bits are set as 1, the MOSI0 is the even bit data output and the MISO0 will be set as the odd bit data output. If the DUAL_IO_EN is set as 1 and DUAL_IO_DIR is set as 0, both the MISO0 and MOSI0 will be set as data input ports.

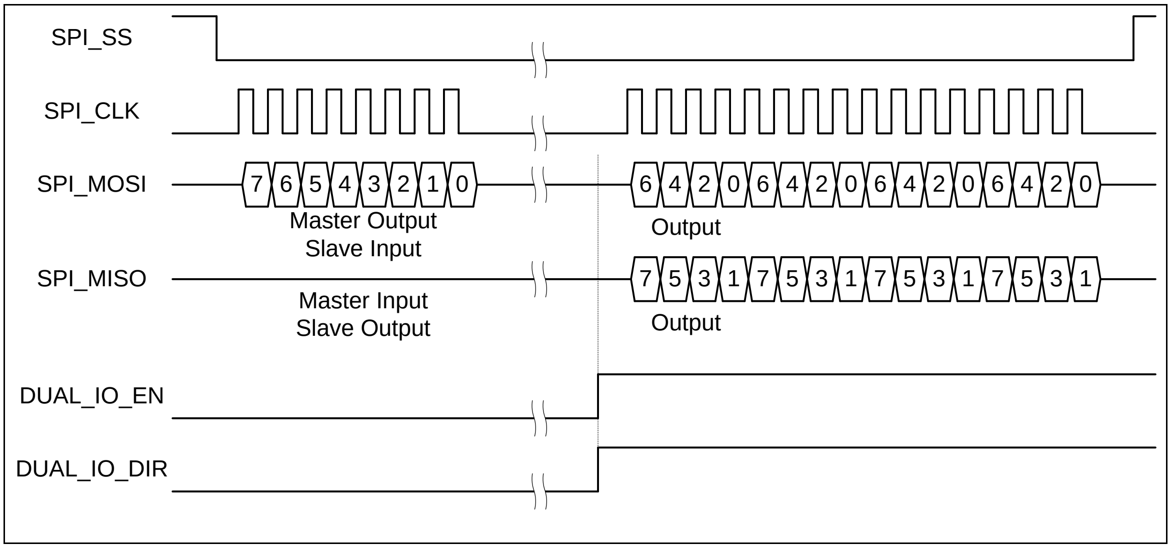


Figure 5-59 Bit Sequence of Dual Output Mode

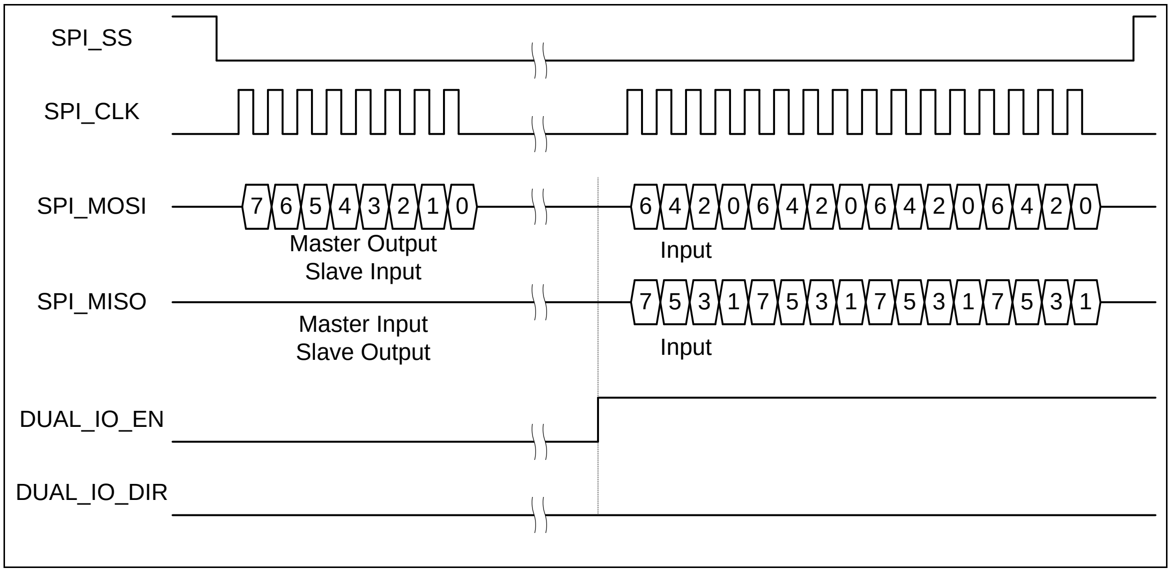


Figure 5-60 Bit Sequence of Dual Input Mode

FIFO Mode

The SPI controller supports FIFO mode when the FIFO bit in SPI_CNTRL[21] is set as 1. The SPI controllers equip with eight 32-bit wide transmit and receive FIFO buffers.

The transmit FIFO buffer is an 8-layer depth, 32-bit wide, first-in, first-out register buffer. Data can be written to the transmit FIFO buffer through software by writing the SPI_TX0 register. The data stored in the transmit FIFO buffer will be read and sent out by the transmission control logic. If the 8-layer transmit FIFO buffer is full, the TX_FULL bit will be set to 1. When the SPI transmission logic unit draws out the last datum of the transmit FIFO buffer, so that the 8-layer transmit FIFO buffer is empty, the TX_EMPTY bit will be set to 1. Notice that the TX_EMPTY flag is set to 1 while the last transaction is still in progress. In Master mode, both the GO_BUSY bit and TX_EMPTY bit should be checked by software to make sure whether the SPI is in idle or not.

The received FIFO buffer is also an 8-layer depth, 32-bit wide, first-in, first-out register buffer. The receive control logic will store the received data to this buffer. The FIFO buffer data can be read from SPI_RX0 register by software. There are FIFO related status bits, like RX_EMPTY and RX_FULL, to indicate the current status of FIFO buffer.

In FIFO mode, the transmitting and receiving threshold can be set through software by setting the TX_THRESHOLD and RX_THRESHOLD settings. When the count of valid data stored in transmit FIFO buffer is less than or equal to TX_THRESHOLD setting, the TX_INTSTS bit will be set to 1. When the count of valid data stored in receive FIFO buffer is larger than RX_THRESHOLD setting, the RX_INTSTS bit will be set to 1.

In FIFO mode, 8 data can be written to the SPI transmit FIFO buffer by software in advance. When the SPI controller operates with FIFO mode, the GO_BUSY bit of SPI_CNTRL register will be controlled by hardware, and the content of SPI_CNTRL register should not be modified by software unless the FIFO bit is cleared to disable FIFO mode.

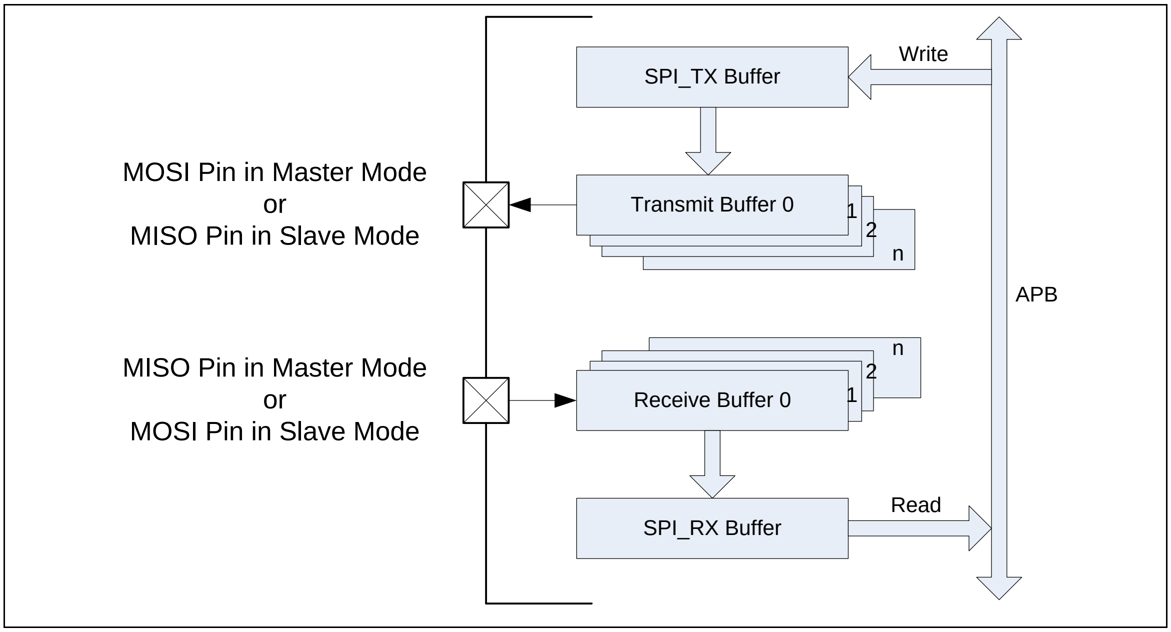


Figure 5-61 FIFO Mode Block Diagram

In Master mode, during transmission operation, the TX_EMPTY flag will be cleared to 0 when the FIFO bit is set to 1 and the first datum will be written to the SPI_TX0 register by software. The transmission immediately starts as long as the transmit FIFO buffer is not empty. User can write the next data into SPI_TX0 register immediately. 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]). User can write data into SPI_TX0 register as long as the TX_FULL flag is 0.

The subsequent transactions will be triggered automatically if the transmitted data are updated in time. If the SPI_TX0 register does not be updated after all data transfer are done, the transfer will stop.

In Master mode during receiving operation, the serial data are received from MISOx pin and stored to receive FIFO buffer. The RX_EMPTY flag will be cleared to 0 while the receive FIFO buffer contain unread data. The received data can be read by software from SPI_RX0 register as long as the RX_EMPTY flag is 0. If the receive FIFO buffer contains 8 unread data, the RX_FULL flag will be set to 1. The SPI controller will stop receiving data until the SPI_RX0 register is read by software.

In Slave mode, when the FIFO bit is set as 1, the GO_BUSY bit will be set as 1 by hardware automatically.

In Slave mode, during transmission operation, when data is written to the SPI_TX0 register by software, the data will be loaded into transmit FIFO buffer and the TX_EMPTY flag will be set to 0. The transmission will start when the slave device receives clock signal from master. Data can be written to SPI_TX0 register as long as the TX_FULL flag is 0. After all data have been drawn out by the SPI transmission logic unit and the SPI_TX0 register is not updated by software, the TX_EMPTY flag will be set to 1.

In Slave mode, during receiving operation, the serial data is received from MOSIx pin and stored to SPI_RX0 register. The reception mechanism is similar to Master mode reception operation.

Interrupt

- SPI unit transfer interrupt

As the SPI controller finishes a unit transfer, the unit transfer interrupt flag IF (SPI_CNTRL[16]) will be set to 1. The unit transfer interrupt event will generate an interrupt to CPU if the unit transfer interrupt enable bit IE (SPI_CNTRL[17]) is set. The unit transfer interrupt flag can be cleared only by writing 1 to it.

- SPI Slave 3-wire mode start interrupt

In 3-wire mode, the Slave 3-wire mode start interrupt flag, SLV_START_INTSTS, will be set to 1 when the slave senses the SPI clock signal. The SPI controller will issue an interrupt if the SSTA_INTEN is set to 1. If the count of the received bits is less than the setting of TX_BIT_LEN and there is no more serial clock input over the expected time period which is defined by the user, the user can set the SLV_ABORT bit to abort the current transfer. The unit transfer interrupt flag, IF, will be set to 1 if the SLV_ABORT bit is set by software.

- Receive FIFO time-out interrupt

In FIFO mode, there is time-out function to inform user. If there is a received data in the FIFO and it does not be read by software over 64 SPI engine clock periods in Master mode or over 576 SPI engine clock periods in Slave mode, it will send a time-out interrupt to the system if the time-out interrupt enable bit, FIFO_CTL[21], is set to 1.

- Transmit FIFO interrupt

In FIFO mode, if the valid data count of the transmit FIFO buffer is less than or equal to the setting value of TX_THRESHOLD, the transmit FIFO interrupt flag will be set to 1. The SPI controller will generate a transmit FIFO interrupt to the system if the transmit FIFO interrupt enable bit, SPI_FIFO_CTL[3], is set to 1.

- Receive FIFO interrupt

In FIFO mode, if the valid data count of the receive FIFO buffer is larger than the setting value of RX_THRESHOLD, the receive FIFO interrupt flag will be set to 1. The SPI controller will generate a receive FIFO interrupt to the system if the receive FIFO interrupt enable bit, SPI_FIFO_CTL[2], is set to 1.

5.10.5 Timing Diagram

The active state of slave select signal can be defined by setting the SS_LVL bit (SPI_SSR[2]) and SS_LTRIG bit (SPI_SSR[4]). The serial clock (SPICLK) idle state can be configured as high 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]), and transmit/receive data from MSB or LSB first in LSB bit (SPI_CNTRL[10]). User can also select which edge of serial clock to transmit/receive data in TX_NEG/RX_NEG (SPI_CNTRL[2:1]). Four SPI timing diagrams for master/slave operations and the related settings are shown below.

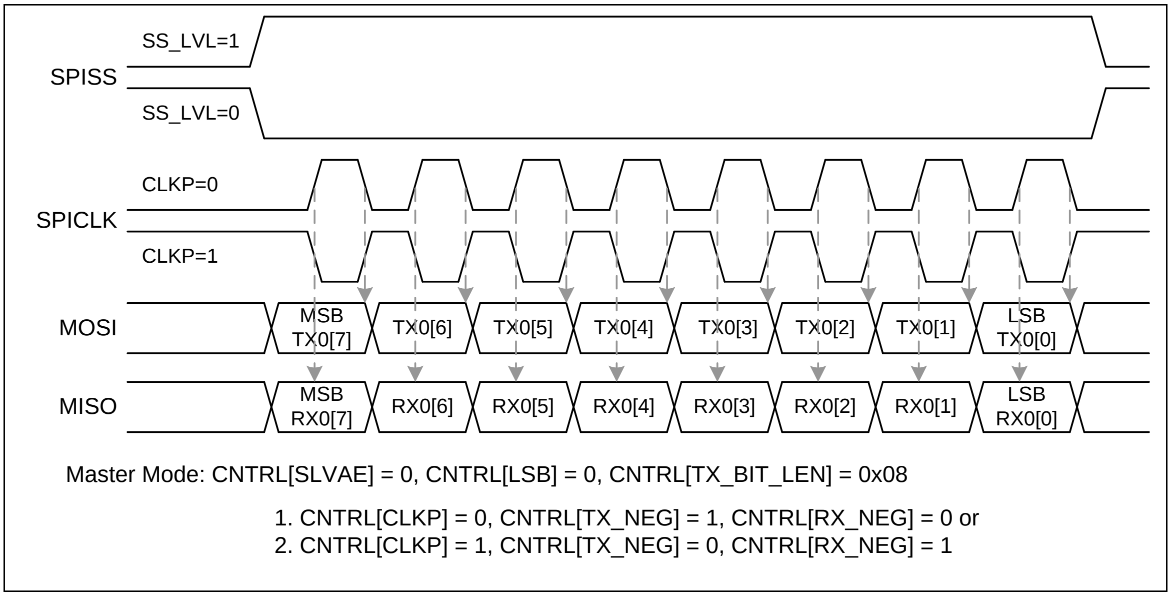
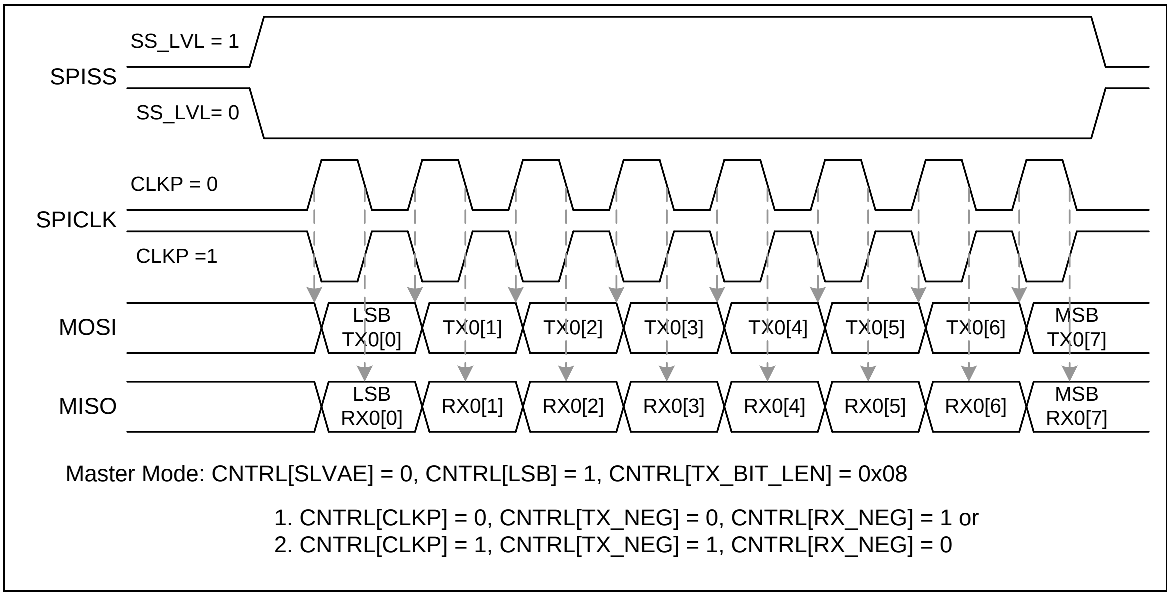

SPISS
SS_LVL=1
SS_LVL=0
SPICLK
CLKP=0
CLKP=1
MOSI
MSB TX0[7]
TX0[6]
TX0[5]
TX0[4]
TX0[3]
TX0[2]
TX0[1]
LSB TX0[0]
MISO
MSB RX0[7]
RX0[6]
RX0[5]
RX0[4]
RX0[3]
RX0[2]
RX0[1]
LSB RX0[0]
Master Mode: CNTRL[SLVAE] = 0, CNTRL[LSB] = 0, CNTRL[TX_BIT_LEN] = 0x08
1. CNTRL[CLKP] = 0, CNTRL[TX_NEG] = 1, CNTRL[RX_NEG] = 0 or
2. CNTRL[CLKP] = 1, CNTRL[TX_NEG] = 0, CNTRL[RX_NEG] = 1

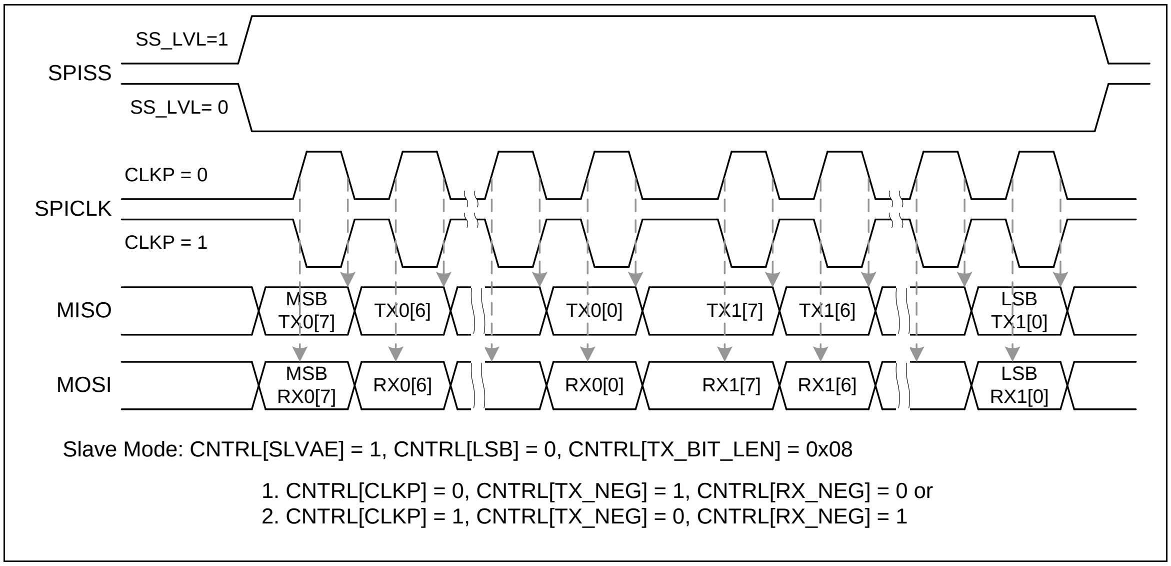
Figure 5-62 SPI Timing in Master Mode



Master Mode: CNTRL[SLVAE] = 0, CNTRL[LSB] = 1, 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 5-63 SPI Timing in Master Mode (Alternate Phase of SPICLK)



Slave Mode: CNTRL[SLVAE] = 1, CNTRL[LSB] = 0, 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 5-64 SPI Timing in Slave Mode

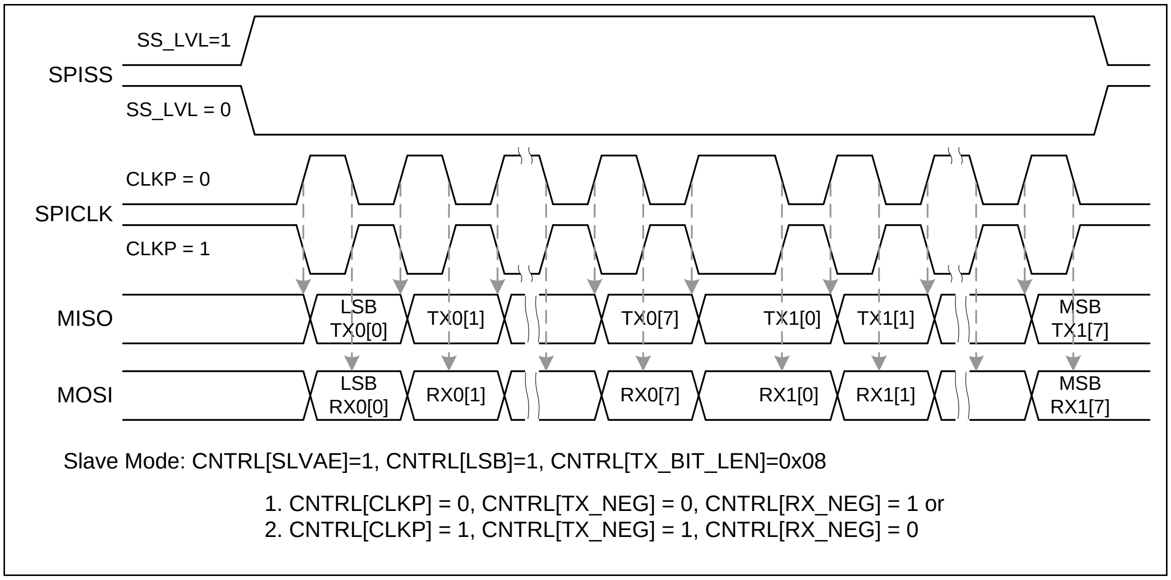

SS_LVL=1
SPISS
SS_LVL = 0
CLKP = 0
SPICLK
CLKP = 1
MISO
LSB
TX0[0]
TX0[1]
TX0[7]
TX1[0]
TX1[1]
MSB
TX1[7]
MOSI
LSB
RX0[0]
RX0[1]
RX0[7]
RX1[0]
RX1[1]
MSB
RX1[7]
Slave Mode: CNTRL[SLVAE]=1, CNTRL[LSB]=1, CNTRL[TX_BIT_LEN]=0x08
1. CNTRL[CLKP] = 0, CNTRL[TX_NEG] = 0, CNTRL[RX_NEG] = 1 or
2. CNTRL[CLKP] = 1, CNTRL[TX_NEG] = 1, CNTRL[RX_NEG] = 0

Figure 5-65 SPI Timing in Slave Mode (Alternate Phase of SPICLK)

5.10.6 Programming Examples

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

- Data bit is latched on positive edge of 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.
- Uses the first SPI slave select pin to connect with an off-chip slave device. The slave select signal is active low.

The operation flow is as follows.

1) Set the DIVIDER (SPI_DIVIDER [7:0]) 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).

    2. Select low level trigger output of slave select signal in the Slave Select Active Level bit SS_LVL (SPI_SSR[2] = 0) and Slave Select Level Trigger bit SS_LTRIG (SPI_SSR[4] = 1).

    3. Select slave select signal to be output active at the I/O pin by setting the Slave Select Register bits SSR[0] (SPI_SSR[0]) to active the off-chip slave device.

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

1. Set the 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 MSB transfer first in MSB bit (SPI_CNTRL[10] = 0).

4) If this SPI master attempts to transmit (write) one byte data to the off-chip slave device, write the byte data that will be transmitted into the SPI_TX0 register.

5) If this SPI master just only attempts to receive (read) one byte data from the off-chip slave device and does not care what data will be transmitted, the SPI_TX0 register does not need to be updated by software.

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

7) Waiting for SPI interrupt (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 SPI_RX0[7:0].

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

Example 2: The SPI controller is set as a slave device and connects with an off-chip master device. The off-chip master device communicates with the on-chip SPI slave controller through the SPI interface with the following specifications:

- Data bit is latched on positive edge of 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 the 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 LSB transfer first in LSB bit (SPI_CNTRL[10] = 1).

3) If this SPI slave attempts to transmit (be read) one byte data to the off-chip master device, write the byte data that will be transmitted into the SPI_TX0 register.

4) If this SPI slave just only attempts to receive (be written) one byte data from the off-chip master device and does not care what data will be transmitted, the SPI_TX0 register does not need to be updated by software.

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 (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 SPI_RX0[7:0].

8) Go to 3) to continue another data transfer or stop data transfer.

5.10.7 SPI Register Map

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

Register	Offset	R/W	Description	Reset Value
SPI Base Address: **SPI0_BA = 0x4003_0000** **SPI1_BA = 0x4003_4000** **SPI2_BA = 0x4013_0000**				
SPI_CNTRL	SPIx_BA+0x00	R/W	Control and Status Register	0x0500_3004
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_1000
SPI_FIFO_CTL	SPIx_BA+0x40	R/W	FIFO Control Register	0x4400_0000
SPI_STATUS	SPIx_BA+0x44	R/W	SPI Status Register	0x0500_0000

5.10.8 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_3004

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	Reserved	REORDER	SLAVE	IE	IF
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
SP_CYCLE				CLKP	LSB	Reserved	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TX_BIT_LEN					TX_NEG	RX_NEG	GO_BUSY

Bits	Description	
[31:28]	**Reserved**	Reserved
[27]	**TX_FULL**	**Transmit FIFO Buffer Full Indicator (Read Only)** It is a mutual mirror bit of SPI_STATUS[27]. 1 = Transmit FIFO buffer is full. 0 = Transmit FIFO buffer is not full.
[26]	**TX_EMPTY**	**Transmit FIFO Buffer Empty Indicator (Read Only)** It is a mutual mirror bit of SPI_STAUTS[26]. 1 = Transmit FIFO buffer is empty. 0 = Transmit FIFO buffer is not empty.
[25]	**RX_FULL**	**Receive FIFO Buffer Full Indicator (Read Only)** It is a mutual mirror bit of SPI_STATUS[25]. 1 = Receive FIFO buffer is full. 0 = Receive FIOF buffer is not full.
[24]	**RX_EMPTY**	**Receive FIFO Buffer Empty Indicator (Read Only)** It is a mutual mirror bit of SPI_CNTRL[24]. 1 = Receive FIFO buffer is empty. 0 = Receive FIFO buffer is not empty.
[23]	**VARCLK_EN**	**Variable Clock Enable (Master Only)** 1 = Serial clock output frequency is variable. The output frequency is decided by the value of VARCLK, DIVIDER, and DIVIDER2.

		0 = Serial clock output frequency is fixed and decided only by the value of DIVIDER. **Note:** When this VARCLK_EN bit is set to 1, the setting of TX_BIT_LEN must be programmed as 0x10 (16-bit mode).
[22]	**TWOB**	**2-Bit Mode Enable** 1 = 2-bit mode Enabled. 0 = 2-bit mode Disabled. **Note:** When 2-bit mode is enabled, the serial transmitted 2-bit data are from SPI_TX1/0, and the received 2-bit data input are put in SPI_RX1/0.
[21]	**FIFO**	**FIFO Mode Enable** 1 = FIFO mode Enabled. 0 = FIFO mode Disabled. **Note:** 1. Before enabling FIFO mode, the other related settings should be set in advance. 2. In Master mode, if the FIFO mode is enabled, the GO_BUSY bit will be set to 1 automatically after data is written into the FIFO buffer by software; the GO_BUSY bit will be cleared to 0 automatically when the SPI controller is in idle. If all data stored at transmit FIFO buffer are sent out, the TX_EMPTY bit will be set to 1 and the GO_BUSY bit will be cleared to 0.
[19]	**REORDER**	**Byte Reorder Function Enable** 1 = Byte reorder function Enabled. A byte suspend interval will be inserted among each byte. The period of the byte suspend interval depends on the setting of SP_CYCLE. 0 = Byte reorder function Disabled. **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, the slave select pin must be kept at active state during the byte suspend interval. 3. The byte reorder function is not supported when the variable serial clock function or Dual I/O mode is enabled.
[18]	**SLAVE**	**Slave Mode Enable** 1 = Slave mode. 0 = Master mode.
[17]	**IE**	**Unit Transfer Interrupt Enable** 1 = SPI unit transfer interrupt Enabled. 0 = SPI unit transfer interrupt Disabled.
[16]	**IF**	**Unit Transfer Interrupt Flag** 1 = SPI controller has finished one unit transfer. 0 = No transaction has been finished since this bit was cleared to 0. **Note:** This bit will be cleared by writing 1 to itself.
[15:12]	**SP_CYCLE**	**Suspend Interval (Master Only)** The four bits provide configurable suspend interval between two successive transmit/receive transaction in a transfer. The definition of the suspend interval is the interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value is 0x3. The period of the

		suspend interval is obtained according to the following equation. (SP_CYCLE[3:0] + 0.5) * period of SPICLK clock cycle Example: SP_CYCLE = 0x0 … 0.5 SPICLK clock cycle SP_CYCLE = 0x1 … 1.5 SPICLK clock cycle …… SP_CYCLE = 0xE … 14.5 SPICLK clock cycle SP_CYCLE = 0xF … 15.5 SPICLK clock cycle If the variable clock function is enabled and the transmit FIFO buffer is not empty, the minimum period of suspend interval between the successive transactions is (6.5 + SP_CYCLE) * SPICLK clock cycle.
[11]	**CLKP**	**Clock Polarity** 1 = SPICLK is idle high. 0 = SPICLK is idle low.
[10]	**LSB**	**Send LSB First** 1 = The LSB, bit 0 of the SPI TX0/1 register, is sent first to the SPI data output pin, and the first bit received from the SPI data input pin will be put in the LSB position of the RX register (bit 0 of SPI_RX0/1). 0 = The MSB, which bit of transmit/receive register depends on the setting of TX_BIT_LEN, is transmitted/received first.
[9:8]	**Reserved**	Reserved

[7:3]	**TX_BIT_LEN**	**Transmit Bit Length**
		This field specifies how many bits can be transmitted / received in one transaction. The minimum bit length is 8 bits and can up to 32 bits.
		TX_BIT_LEN = 0x08 … 8 bits
		TX_BIT_LEN = 0x09 … 9 bits
		……
		TX_BIT_LEN = 0x1F … 31 bits
		TX_BIT_LEN = 0x00 … 32 bits

[2]	**TX_NEG**	**Transmit on Negative Edge** 1 = Transmitted data output signal is changed on the falling edge of SPICLK. 0 = Transmitted data output signal is changed on the rising edge of SPICLK.
[1]	**RX_NEG**	**Receive on Negative Edge** 1 = Received data input signal is latched on the falling edge of SPICLK. 0 = Received data input signal is latched on the rising edge of SPICLK.
[0]	**GO_BUSY**	**SPI Transfer Control Bit 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 = Data transfer stopped. If FIFO mode is disabled, during the data transfer, this bit keeps the value of 1. As the transfer is finished, this bit will be cleared automatically. Software can read this bit to check if the SPI is in busy status.

		In FIFO mode, this bit will be controlled by hardware. Software should not modify this bit. In Slave mode, this bit always returns 1 when this register is read by software. In Master mode, this bit reflects the busy or idle status of SPI. **Note:** 1. When FIFO mode is disabled, all configurations should be set before writing 1 to this GO_BUSY bit. 2. When FIFO mode is disabled and the software uses TX or RX PDMA function to transfer data, this bit will be cleared after the PDMA finishes the data transfer.

SPI Divider Register (SPI_DIVIDER)

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

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

Bits	Description	
[31:24]	**Reserved**	Reserved
[23:16]	**DIVIDER2**	**Clock Divider 2 Register (Master Only)** The value in this field is the 2nd frequency divider for generating the second clock of the variable clock function. The frequency is obtained according to the following equation: $f_{clock2} = \frac{f_{spi\_eclk}}{(DIVIDER2+1)*2}$ If the VARCLK_EN bit is cleared to 0, this setting is unmeaning.
[15:8]	**Reserved**	Reserved
[7:0]	**DIVIDER**	**Clock Divider 1 Register** The value in this field is the frequency divider for generating the SPI engine clock, f_{spi_eclk}, and the SPI serial clock of SPI master. The frequency is obtained according to the following equation. If the bit of BCn, SPI_CNTRL2[31], is set to 0, $f_{spi\_eclk} = \frac{f_{system\_clock}}{(DIVIDER+1)*2}$ else if BCn is set to 1, $f_{spi\_eclk} = \frac{f_{spi\_clock\_src}}{(DIVIDER+1)}$ where $f_{spi\_clock\_src}$ is the SPI engine clock source, which is defined in the

		CLKSEL1 register.

SPI Slave Select Register (SPI_SSR)

Register	Offset	R/W	Description	Reset Value
SPI_SSR	SPIx_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	Description	
[31:6]	**Reserved**	Reserved
[5]	**LTRIG_FLAG**	**Level Trigger Accomplish Flag** In Slave mode, this bit indicates whether the received bit number meets the requirement or not after the current transaction done. 1 = Transferred bit length meets the specified requirement which defined in TX_BIT_LEN. 0 = Transferred bit length of one transaction does not meet the specified requirement. **Note:** This bit is READ only. As the GO_BUSY bit is set to 1 by software, the LTRIG_FLAG will be cleared to 0 after 4 SPI engine clock periods plus 1 system clock period. In FIFO mode, this bit has no meaning.
[4]	**SS_LTRIG**	**Slave Select Level Trigger Enable (Slave Only)** 1 = Slave select signal is level-trigger. The SS_LVL bit decides the signal is active low or active high. 0 = Slave select signal is edge-trigger. This is the default value. The SS_LVL bit decides the signal is active after a falling-edge or rising-edge.
[3]	**AUTOSS**	**Automatic Slave Select Function Enable (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 SPI_SSR[1:0], will be asserted by the SPI controller when transmit/receive is started, 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 the corresponding bits of SPI_SSR[1:0].
[2]	**SS_LVL**	**Slave Select Active Level** This bit defines the active status of slave select signal (SPISSx0/1). 1 = The slave select signal SPISSx0/1 is active on high-level/rising-edge.

		0 = The slave select signal SPISSx0/1 is active on low-level/falling-edge.
[1:0]	**SSR**	**Slave Select Control Bits (Master Only)** If AUTOSS bit is cleared, writing 1 to any bit of this field sets the proper SPISSx0/1 line to an active state and writing 0 sets the line back to inactive state. If the 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 appropriate 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 defined as the 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	Description	
[31:0]	**RX**	**Data Receive Register** The data receive register holds the datum received from SPI data input pin. If FIFO mode is disabled, the last received data can be accessed through software by reading this register. If the FIFO bit is set as 1 and the RX_EMPTY bit, SPI_CNTRL[24] or SPI_STATUS[24], is not set to 1, the receive FIFO buffer can be accessed through software by reading this register. This is a read-only register.

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	Description	
[31:0]	**TX**	**Data Transmit Register** The data transmit registers hold the data to be transmitted in the next transfer. The number of valid bits depends on the setting of transmit bit length field of the SPI_CNTRL register. For example, if TX_BIT_LEN is set to 0x08, the bits TX[7:0] will be transmitted in next transfer. If TX_BIT_LEN is set to 0x00, the SPI controller will perform a 32-bit transfer. **Note:** When the SPI controller is configured as a slave device and FIFO mode is disabled, if the SPI controller attempts to transmit data to a master, the transmit data register should be updated by software before setting the GO_BUSY bit to 1

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	Description	
[31:0]	**VARCLK**	**Variable Clock Pattern** This register defines the clock pattern of the SPI transfer. If the variable clock function is disabled, this setting is unmeaning. Refer to the “Variable Clock Function” paragraph for more detail description.

SPI_DMA Control Register (SPI_DMA)

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

Bits	Description	
[31:3]	**Reserved**	Reserved
[2]	**PDMA_RST**	**PDMA Reset** 1 = Reset the PDMA control logic of the SPI controller. This bit will be cleared to 0 automatically. 0 = No effect.
[1]	**RX_DMA_GO**	**Receive DMA Start** Setting this bit to 1 will start the receive PDMA process. The SPI controller will issue request to PDMA controller automatically when the SPI receive buffer is not empty. This bit will be cleared to 0 by hardware automatically after PDMA transfer is done. If the software uses the receive PDMA function to access the received data of SPI and does not use the transmit PDMA function, the GO_BUSY bit should be set by software. Enabling FIFO mode is recommended if the software uses more than one PDMA channel to transfer data. In Slave mode and when FIFO mode is disabled, if the software only uses one PDMA channel for SPI receive PDMA function and the other PDMA channels are not in use, the minimal suspend interval between two successive transactions must be larger than (9 SPI slave engine clock periods + 4 APB clock periods) for edge-trigger mode or (9.5 SPI slave engine clock periods + 4 APB clock periods) for level-trigger mode.
[0]	**TX_DMA_GO**	**Transmit DMA Start** Setting this bit to 1 will start the transmit PDMA process. SPI controller will issue request to PDMA controller automatically. Hardware will clear this bit to 0 automatically after PDMA transfer done. If the SPI transmit PDMA function is used to transfer data, the GO_BUSY bit should not be set to 1 by software. The PDMA control logic of SPI controller will set it automatically whenever necessary. In Slave mode and when FIFO mode is disabled, the minimal suspend interval

		between two successive transactions must be larger than (8 SPI serial clock periods + 14 APB clock periods) for edge-trigger mode or (9.5 serial clock periods + 14 APB clock periods) for level-trigger mode. If the 2-bit Transfer mode is enabled, additional 18 APB clock periods for the above conditions is required.

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_1000

31	30	29	28	27	26	25	24
BCn	**Reserved**						
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							**SS_INT_ OPT**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**DUAL_IO_ EN**	**DUAL_IO_ DIR**	**SLV_START_ INTSTS**	**SSTA_ INTEN**	**SLV_ABORT**	**NOSLVSEL**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31]	**BCn**	**SPI Engine Clock Backward Compatible Option** 1 = Clock configuration is not backward compatible. 0 = Backward compatible clock configuration. Refer to the description of SPI_DIVIDER register for details.
[30:17]	**Reserved**	Reserved
[16]	**SS_INT_OPT**	**Slave Select Inactive Interrupt Option** This setting is only available if the SPI controller is configured as level trigger slave device. 1 = As the slave select signal goes to inactive level, the IF bit will be set to 1. 0 = As the slave select signal goes to inactive level, the IF bit will NOT be set to 1.
[15:14]	**Reserved**	Reserved
[13]	**DUAL_IO_EN**	**Dual I/O Mode Enable** 1 = Dual I/O mode Enabled. 0 = Dual I/O mode Disabled.
[12]	**DUAL_IO_DIR**	**Dual I/O Mode Direction Control** 1 = Dual Output mode. 0 = Dual Input mode.

[11]	**SLV_START_INTSTS**	**Slave 3-Wire Mode Start Interrupt Status** This bit indicates if a transaction has started in Slave 3-wire mode. It is a mutual mirror bit of SPI_STATUS[11]. 1 = A transaction has started in Slave 3-wire mode. It will be cleared automatically when a transaction is done or by writing 1 to this bit. 0 = Slave has not detected any SPI clock transition since the SSTA_INTEN bit was set to 1.
[10]	**SSTA_INTEN**	**Slave 3-Wire Mode Start Interrupt Enable** Used to enable interrupt when the transfer has started in Slave 3-wire mode. 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 = Transaction start interrupt Enabled. It will be cleared to 0 as the current transfer is done or the SLV_START_INTSTS bit is cleared. 0 = Transaction start interrupt Disabled.
[9]	**SLV_ABORT**	**Slave 3-Wire Mode Abort Control** In normal operation, there is an interrupt event when the received data meet the required bits which defined in TX_BIT_LEN. If the received bits are less than the requirement and there is no more serial clock input over the one transfer time in Slave 3-wire mode, the user can set this bit to force the current transfer done and then the user can get a transfer done interrupt event. **Note:** This bit will be cleared to 0 automatically by hardware after it is set to 1 by software.
[8]	**NOSLVSEL**	**Slave 3-Wire Mode Enable** This is used to ignore the slave select signal in Slave mode. The SPI controller can work with 3-wire interface including SPICLK, SPI_MISO, and SPI_MOSI. 1 = 3-wire bi-direction interface. 0 = 4-wire bi-direction interface. **Note:** In Slave 3-wire mode, the SS_LTRIG, SPI_SSR[4] will be set as 1 automatically.
[7:0]	**Reserved**	Reserved

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	0x4400_0000

31	30	29	28	27	26	25	24
Reserved	TX_THRESHOLD			Reserved	RX_THRESHOLD		
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved		TIMEOUT_INTEN	Reserved				
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	RXOV_INTEN	Reserved		TX_INTEN	RX_INTEN	TX_CLR	RX_CLR

Bits	Description	
[31]	**Reserved**	Reserved
[30:28]	**TX_THRESHOLD**	**Transmit FIFO Threshold** If the valid data count of the transmit FIFO buffer is less than or equal to the TX_THRESHOLD setting, the TX_INTSTS bit will be set to 1, else the TX_INTSTS bit will be cleared to 0.
[27]	**Reserved**	Reserved
[26:24]	**RX_THRESHOLD**	**Receive FIFO Threshold** If the valid data count of the receive FIFO buffer is larger than the RX_THRESHOLD setting, the RX_INTSTS bit will be set to 1, else the RX_INTSTS bit will be cleared to 0.
[23:22]	**Reserved**	Reserved
[21]	**TIMEOUT_INTEN**	**Receive FIFO Time-out Interrupt Enable** 1 = Time-out interrupt Enabled. 0 = Time-out interrupt Disabled.
[20:7]	**Reserved**	Reserved
[6]	**RXOV_INTEN**	**Receive FIFO Overrun Interrupt Enable** 1 = Receive FIFO overrun interrupt Enabled. 0 = Receive FIFO overrun interrupt Disabled.

[5:4]	**Reserved**	Reserved
[3]	**TX_INTEN**	**Transmit Threshold Interrupt Enable** 1 = TX threshold interrupt Enabled. 0 = TX threshold interrupt Disabled.
[2]	**RX_INTEN**	**Receive Threshold Interrupt Enable** 1 = RX threshold interrupt Enabled. 0 = RX threshold interrupt Disabled.
[1]	**TX_CLR**	**Clear Transmit FIFO Buffer** 1 = Clear transmit FIFO buffer. 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 it is set to 1 by software. 0 = No effect.
[0]	**RX_CLR**	**Clear Receive FIFO Buffer** 1 = Clear receive FIFO buffer. 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 it is set to 1 by software. 0 = No effect.

SPI Status Register (SPI_STATUS)

Register	Offset	R/W	Description	Reset Value
SPI_STATUS	SPIx_BA+0x44	R/W	SPI Status Register	0x0500_0000

31	30	29	28	27	26	25	24
TX_FIFO_COUNT				TX_FULL	TX_EMPTY	RX_FULL	RX_EMPTY
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved			TIMEOUT	Reserved			IF
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
RX_FIFO_COUNT				SLV_START _INTSTS	Reserved		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved			TX_INTSTS	Reserved	RX_OVER RUN	Reserved	RX_INTSTS

Bits	Description	
[31:28]	**TX_FIFO_COUNT**	**Transmit FIFO Data Count (Read Only)** This bit field indicates the valid data count of transmit FIFO buffer.
[27]	**TX_FULL**	**Transmit FIFO Buffer Full Indicator (Read Only)** It is a mutual mirror bit of SPI_CNTRL[27]. 1 = Transmit FIFO buffer is full. 0 = Transmit FIFO buffer is not full.
[26]	**TX_EMPTY**	**Transmit FIFO Buffer Empty Indicator (Read Only)** It is a mutual mirror bit of SPI_CNTRL[26]. 1 = Transmit FIFO buffer is empty. 0 = Transmit FIFO buffer is not empty.
[25]	**RX_FULL**	**Receive FIFO Buffer Empty Indicator (Read Only)** It is a mutual mirror bit of SPI_CNTRL[24]. 1 = Receive FIFO buffer is empty. 0 = Receive FIFO buffer is not empty.
[24]	**RX_EMPTY**	**Receive FIFO Buffer Empty Indicator (Read Only)** It is a mutual mirror bit of SPI_CNTRL[24]. 1 = Receive FIFO buffer is empty. 0 = Receive FIFO buffer is not empty.
[23:21]	**Reserved**	Reserved

[20]	**TIMEOUT**	**Time-out Interrupt Flag** 1 = Receive FIFO buffer is not empty and no read operation on receive FIFO buffer over 64 SPI clock period in Master mode or over 576 SPI engine clock period in Slave mode. When the received FIFO buffer is read by software, the time-out status will be cleared automatically. 0 = No receive FIFO time-out event. **Note:** This bit will be cleared by writing 1 to itself.
[19:17]	**Reserved**	Reserved
[16]	**IF**	**SPI Unit Transfer Interrupt Flag** It is a mutual mirror bit of SPI_CNTRL[16]. 1 = SPI controller has finished one unit transfer. 0 = No transaction has been finished since this bit was cleared to 0. **Note:** This bit will be cleared by writing 1 to itself.
[15:12]	**RX_FIFO_COUNT**	**Receive FIFO Data Count (Read Only)** This bit field indicates the valid data count of receive FIFO buffer.
[11]	**SLV_START_INTSTS**	**Slave Start Interrupt Status** It is used to dedicate if a transaction has started in Slave 3-wire mode. It is a mutual mirror bit of SPI_CNTRL2[11]. 1 = A transaction has started in Slave 3-wire mode. It will be cleared as a transaction is done or by writing 1 to this bit. 0 = Slave has not detected any SPI clock transition since the SSTA_INTEN bit was set to 1.
[10:5]	**Reserved**	Reserved
[4]	**TX_INTSTS**	**Transmit FIFO Threshold Interrupt Status (Read Only)** 1 = The valid data count within the transmit FIFO buffer is less than or equal to the setting value of TX_THRESHOLD. 0 = The valid data count within the transmit FIFO buffer is larger than the setting value of TX_THRESHOLD. **Note:** If TX_INTEN = 1 and TX_INTSTS = 1, the SPI controller will generate a SPI interrupt request.
[3]	**Reserved**	Reserved
[2]	**RX_OVERRUN**	**Receive FIFO Overrun Status** When the receive FIFO buffer is full, the follow-up data will be dropped and this bit will be set to 1. **Note:** This bit will be cleared by writing 1 to itself.
[1]	**Reserved**	Reserved
[0]	**RX_INTSTS**	**Receive FIFO Threshold Interrupt Status (Read Only)** 1 = The valid data count within the receive FIFO buffer is larger than the setting

		value of RX_THRESHOLD. 0 = The valid data count within the Rx FIFO buffer is smaller than or equal to the setting value of RX_THRESHOLD. **Note:** If RX_INTEN = 1 and RX_INTSTS = 1, the SPI controller will generate a SPI interrupt request.

5.11 Timer Controller (TMR)

5.11.1 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 time-out, or provide the current value during operation.

5.11.2 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)
- Supports event counting function to count the event from external pin
- Supports input capture function to capture or reset counter value

5.11.3 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 5-66 for the timer controller block diagram. There are four options of clock sources for each channel. Figure 5-67 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.

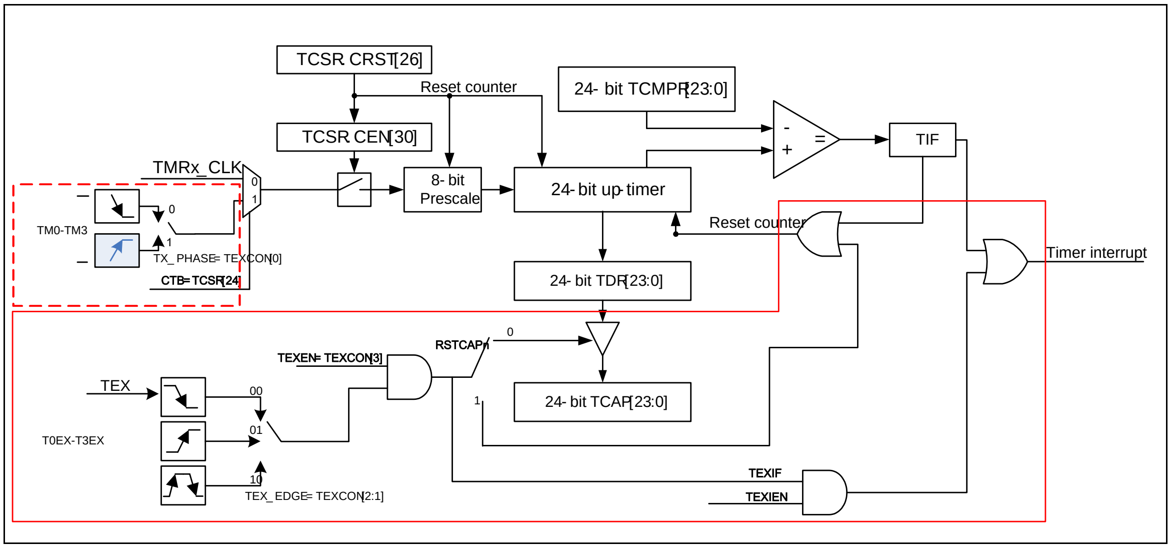


Figure 5-66 Timer Controller Block Diagram

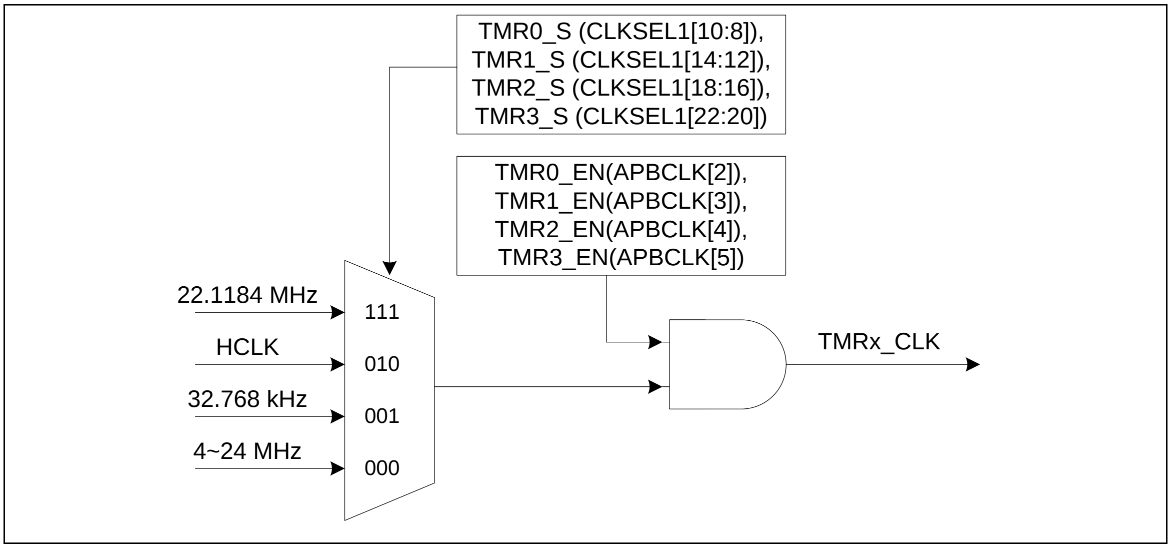


Figure 5-67 Clock Source of Timer Controller

5.11.4 Functional 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 described below.

5.11.4.1 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 counting up. Once the timer counter value reaches timer compare register (TCMPR) value, if IE (TCSR[29] interrupt enable bit) is set to 1, 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.

5.11.4.2 Periodic Mode

If timer is operated at period mode and CEN (TCSR[30] timer enable bit) is set to 1, the timer counter starts counting up. Once the timer counter value reaches timer compare register (TCMPR) value, if IE (TCSR[29] interrupt enable bit) is set to 1, 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 counting up 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, 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.

5.11.4.3 Toggle Mode

If timer is operated in Toggle mode and CEN (TCSR[30] timer enable bit) is set to 1, the timer counter starts counting up. Once the timer counter value reaches timer compare register (TCMPR) value, if IE (TCSR[29] interrupt enable bit) is set to 1, 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 counting up 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, 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.

5.11.4.4 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, 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, 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.

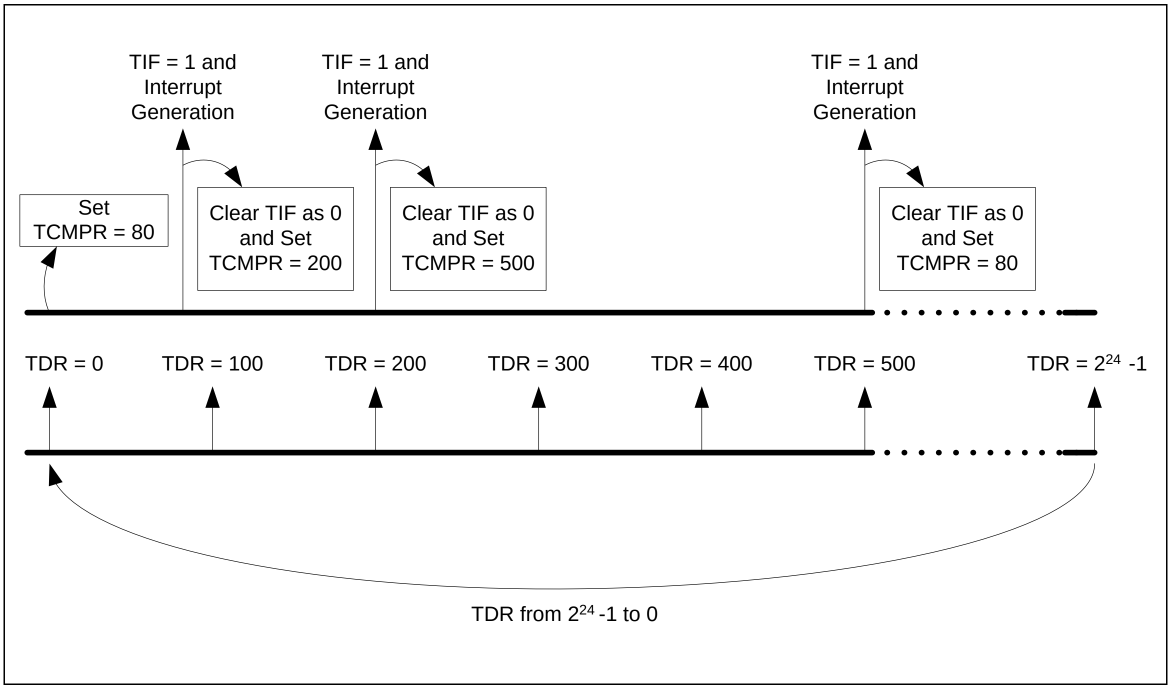

TIF = 1 and Interrupt Generation
TIF = 1 and Interrupt Generation
TIF = 1 and Interrupt Generation
Set TCMPR = 80
Clear TIF as 0 and Set TCMPR = 200
Clear TIF as 0 and Set TCMPR = 500
Clear TIF as 0 and Set TCMPR = 80
TDR = 0
TDR = 100
TDR = 200
TDR = 300
TDR = 400
TDR = 500
TDR = 2^24 -1
TDR from 2^24 -1 to 0

Figure 5-68 Continuous Counting Mode

5.11.4.5 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 5-67 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.

5.11.4.6 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.

5.11.5 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

5.11.6 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**
WAKE_EN	**Reserved**						**TDR_EN**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PRESCALE[7:0]							

Bits	Description	
[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 counting up 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 = Timer Interrupt Enabled 0 = Timer Interrupt Disabled If timer interrupt is enabled, the timer asserts its interrupt signal when the associated

<table>
<tr><td></td><td></td><td>up-timer value is equal to TCMPR.</td></tr>
<tr><td>[28:27]</td><td><b>MODE</b></td><td><b>Timer Operating Mode</b>
<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 5.11.4.4 for detail description about continuous counting mode operation.</td></tr>
</table></td></tr>
<tr><td>[26]</td><td><b>CRST</b></td><td><b>Timer Reset Bit</b>
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</td></tr>
<tr><td>[25]</td><td><b>CACT</b></td><td><b>Timer Active Status Bit (Read only)</b>
This bit indicates the up-timer status.
0 = Timer is not active
1 = Timer is active</td></tr>
<tr><td>[24]</td><td><b>CTB</b></td><td><b>Counter Mode Enable Bit</b>
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 = Counter mode Enabled.
0 = Counter mode Disabled.</td></tr>
<tr><td>[23]</td><td><b>WAKE_EN</b></td><td><b>Wake-up Enable</b>
When WAKE_EN is set and the TIF is set, the timer controller will generator a wake-up trigger event to CPU.
0 = Wake-up trigger event Disabled.
1 = Wake-up trigger event Enabled.</td></tr>
<tr><td>[22:17]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[16]</td><td><b>TDR_EN</b></td><td><b>Data Load Enable</b>
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 Enabled.
0 = Timer Data Register update Disabled.</td></tr>
</table>

[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, 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	Description	
[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 in 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						**TWF**	**TIF**

Bits	Description	
[31:2]	**Reserved**	Reserved
[1]	**TWF**	**Timer Wakeup Flag** If timer causes CPU 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 = Timer does not cause CPU wakeup. 1 = CPU wakes up from sleep or power-down mode by timer time-out.
[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	Description	
[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	Description	
[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	Description	
[31:8]	**Reserved**	Reserved
[7]	**TCDB**	**Timer Counter Pin De-bounce Enable Bit** 1 = De-bounce Enabled. 0 = De-bounce Disabled. 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 = De-bounce Enabled. 0 = De-bounce Disabled. 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 = Timer external interrupt Enabled. 0 = Timer external interrupt Disabled. 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 Selection**

		1 = TEX transition is used as the timer counter reset function. 0 = TEX transition is used as the timer capture function.
[3]	**TEXEN**	**Timer External Pin Enable** This bit enables the reset/capture function on the TEX pin. 1 = Transition detected on the TEX pin will result in capture or reset of timer counter. 0 = TEX pin will be ignored.
[2:1]	**TEX_EDGE**	**Timer External Pin Edge Detect** 00 = 1 to 0 transition on TEX will be detected. 01 = 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

<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="7">Reserved</td><td>TEXIF</td></tr>
</table>

<table>
<tr><th>Bits</th><th colspan="2">Description</th></tr>
<tr><td>[31:1]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[0]</td><td><b>TEXIF</b></td><td><b>Timer External Interrupt Flag</b>
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.</td></tr>
</table>

5.12 Watchdog Timer (WDT)

5.12.1 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.

5.12.2 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.

5.12.3 Block Diagram

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

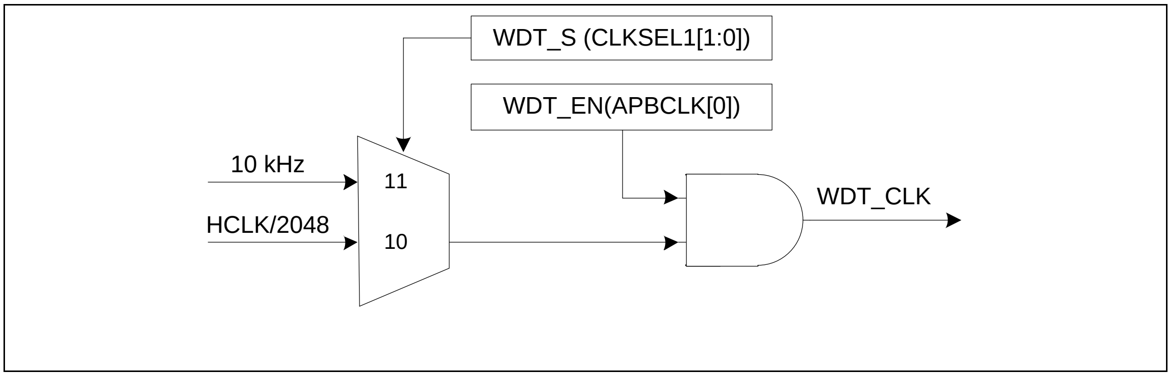
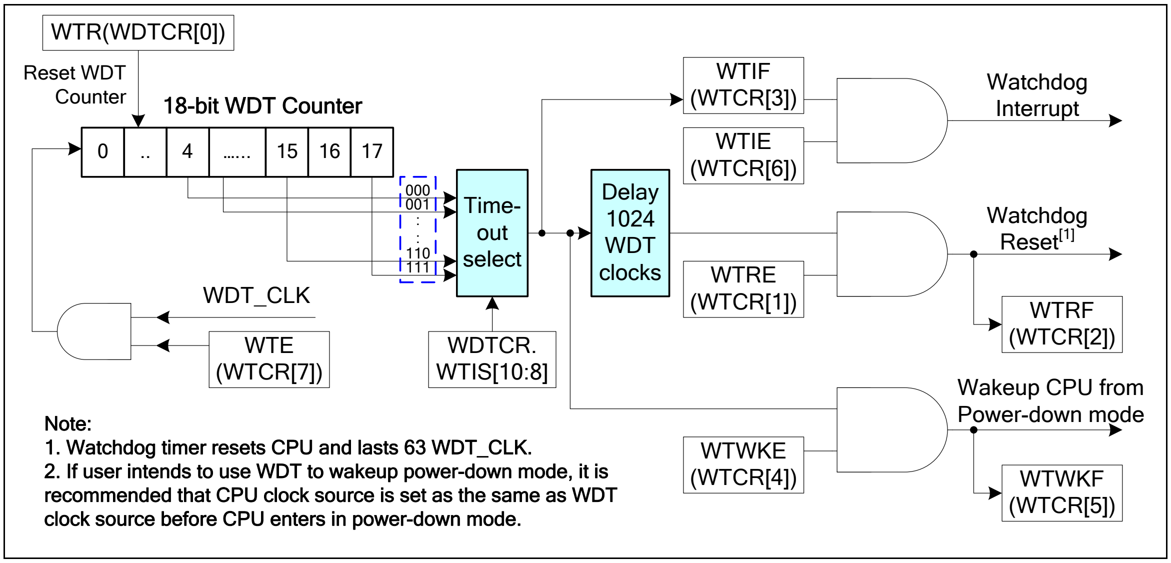


Figure 5-69 Watchdog Timer Clock Control



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 5-70 Watchdog Timer Block Diagram

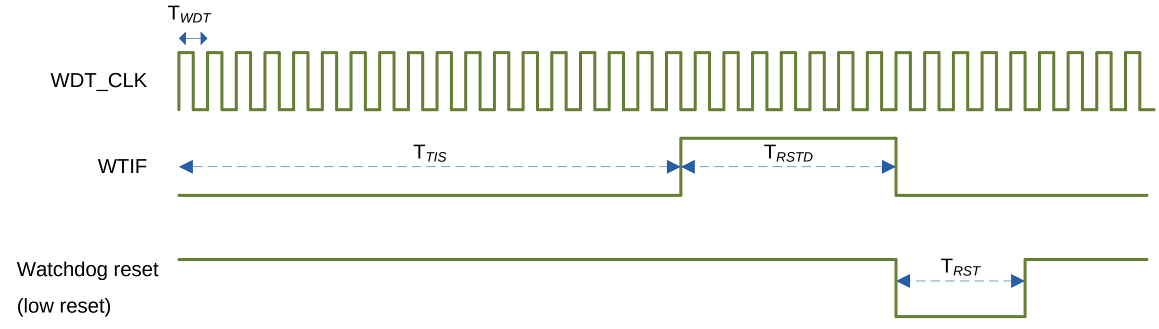
5.12.4 Functional Description

The watchdog timer includes an 18-bit free running counter with programmable time-out intervals. Table 5-9 shows the watchdog time-out interval selection and Figure 5-71 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 woken up from Power-down state. For 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, chip enters Power-down state. After 2^4 * T_{WDT} time is elapsed, chip is woken 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, chip enters Power-down state. After 2^{18} * T_{WDT} time is elapsed, chip is woken up from Power-down state. Note if WTRE (WDTCR [1]) is set to 1, after chip is woken 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	Time-out Interval Period T_{TIS}	Interrupt Period T_{RSTD}
000	2^4 * T_{WDT}	(1/16/128/1024 + 2) * T_{WDT}
001	2^6 * T_{WDT}	(1/16/128/1024 + 2) * T_{WDT}
010	2^8 * T_{WDT}	(1/16/128/1024 + 2) * T_{WDT}
011	2^{10} * T_{WDT}	(1/16/128/1024 + 2) * T_{WDT}
100	2^{12} * T_{WDT}	(1/16/128/1024 + 2) * T_{WDT}
101	2^{14} * T_{WDT}	(1/16/128/1024 + 2) * T_{WDT}
110	2^{16} * T_{WDT}	(1/16/128/1024 + 2) * T_{WDT}
111	2^{18} * T_{WDT}	(1/16/128/1024 + 2) * T_{WDT}

Table 5-9 Watchdog Time-out Interval Selection



- *T_{WDT} : Watchdog Clock Time Period*
- *T_{TIS} : Watchdog Timeout Interval Period ($(2^4 \sim 2^{18}) * T_{WDT}$)*
- *T_{RSTD} : Watchdog Reset Delay Period ($(1/16/128/1024 + 2) * T_{WDT}$)*
- *T_{RST} : Watchdog Reset Period ($63 * T_{WDT}$)*

Figure 5-71 Timing of Interrupt and Reset Signal

5.12.5 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
WTCRALT	WDT_BA+0x04	R/W	Watchdog Timer Alternative Control Register	0x0000_0000

5.12.6 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">Description</th></tr>
<tr><td>[31]</td><td><b>DBGACK_WDT</b></td><td><b>ICE Debug Mode Acknowledge Disable (Write-protection Bit)</b>
0 = ICE debug mode acknowledgement affects 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><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[10:8]</td><td><b>WTIS</b></td><td><b>Watchdog Timer Interval Selection (Write-protection Bit)</b>
These three bits select the time-out interval for the Watchdog timer.
<table>
<tr><th>WTIS</th><th>Time-out Interval Selection</th><th>Interrupt Period</th><th>WTR Time-out 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 = Watchdog timer Disabled (This action will reset the internal counter). 1 = Watchdog timer Enabled.
[6]	**WTIE**	**Watchdog Timer Interrupt Enable (Write-protection Bit)** 0 = Watchdog timer interrupt Disabled. 1 = Watchdog timer interrupt Enabled.
[5]	**WTWKF**	**Watchdog Timer Wake-up Flag** If Watchdog timer causes chip to wake up from Power-down mode, this bit will be set to high. It must be cleared by software by writing 1 to this bit. 0 = Watchdog timer does not cause chip wake up. 1 = Chip woken up from Idle or Power-down mode by Watchdog time-out.
[4]	**WTWKE**	**Watchdog Timer Wake-up Function Enable bit (Write-protection Bit)** 0 = Watchdog timer Wake-up chip function Disabled. 1 = Wake-up function Enabled so that Watchdog timer time-out can wake-up chip from Power-down mode. **Note:** Chip can be woken up by WDT only if WDT clock source select RC10K.
[3]	**WTIF**	**Watchdog Timer Interrupt Flag** If the Watchdog timer interrupt is enabled, 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 occurred. 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, the Watchdog timer has no effect on this bit. 0 = Watchdog timer reset did not occur. 1 = Watchdog timer reset occurred. 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 = Watchdog timer reset function Disabled. 1 = Watchdog timer reset function Enabled.
[0]	**WTR**	**Clear Watchdog Timer (Write-protection Bit)** Setting this bit will clear the Watchdog timer. 0 = No effect.

		1 = Reset the contents of the Watchdog timer. **Note:** This bit will be automatically cleared by hardware.

Watchdog Timer Alternative Control Register (WTCRALT)

Register	Offset	R/W	Description	Reset Value
WTCRALT	WDT_BA+0x04	R/W	Watchdog Timer Alternative Control Register	0x0000_0000

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

Bits	Description	
[31:2]	**Reserved**	Reserved
[1:0]	**WTRDSEL**	**Watchdog Timer Reset Delay Select (Write-protection Bits)** When watchdog time-out happened, software has a time named watchdog reset delay period to clear watchdog timer to prevent watchdog reset happened. Software can select a suitable value of watchdog reset delay period for different watchdog time-out period. 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. 00 = Watchdog reset delay period is 1024 watchdog clock 01 = Watchdog reset delay period is 128 watchdog clock 10 = Watchdog reset delay period is 16 watchdog clock 11 = Watchdog reset delay period is 1 watchdog clock This register will be reset if watchdog reset happened

5.13 Window Watchdog Timer (WWDT)

5.13.1 Overview

The purpose of Window Watchdog Timer is to perform a system reset within a specified window period to prevent software run to uncontrollable status by any unpredictable condition.

5.13.2 Features

- 6-bit down counter and 6-bit compare value to make the window period flexible
- Selectable WWDT clock pre-scale counter to make WWDT time-out interval variable

5.13.3 Block Diagram

The Window Watchdog Timer block diagram is shown as follows.

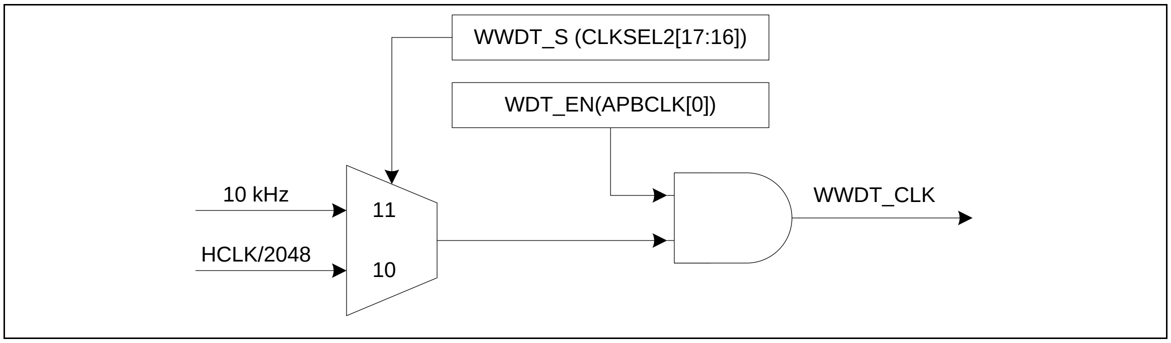


Figure 5-72 Window Watchdog Timer Clock Control

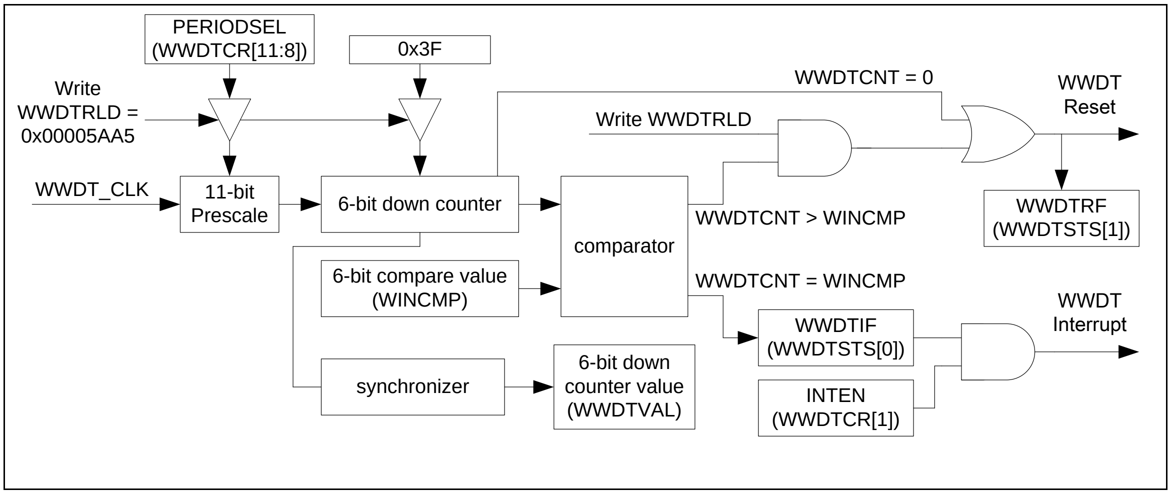


Figure 5-73 Window Watchdog Timer Block Diagram

5.13.4 Functional Description

The window watchdog timer includes a 6-bit down counter with programmable prescaler to define different time-out intervals.

The clock source of 6-bit window watchdog timer is based on system clock divide 2048 or internal 10 kHz oscillator with a programmable 11-bit prescaler. The programmable 11-bit prescaler is controlled by register PERIODSEL (WWDTCR[11:8]) and the correlate of PERIODSEL and prescaler value is listed in Table 5-10.

PERIODSEL	Prescaler Value	Time-out Period	Time-out Interval (WWDT_CLK=10 kHz)
0000	1	1 * 64 * T_{WWDT}	6.4 ms
0001	2	2 * 64 * T_{WWDT}	12.8 ms
0010	4	4 * 64 * T_{WWDT}	25.6 ms
0011	8	8 * 64 * T_{WWDT}	51.2 ms
0100	16	16 * 64 * T_{WWDT}	102.4 ms
0101	32	32 * 64 * T_{WWDT}	204.8 ms
0110	64	64 * 64 * T_{WWDT}	409.6 ms
0111	128	128 * 64 * T_{WWDT}	819.2 ms
1000	192	192 * 64 * T_{WWDT}	1.2288 s
1001	256	256 * 64 * T_{WWDT}	1.6384 s
1010	384	384 * 64 * T_{WWDT}	2.4576 s
1011	512	512 * 64 * T_{WWDT}	3.2768 s
1100	768	768 * 64 * T_{WWDT}	4.9152 s
1101	1024	1024 * 64 * T_{WWDT}	6.5536 s
1110	1536	1536 * 64 * T_{WWDT}	9.8304 s
1111	2048	2048 * 64 * T_{WWDT}	13.1072 s

Table 5-10 Window Watchdog Prescaler Value Selection

Window watchdog timer can be enabled by software setting WWDTEN (WWDTCR[0]) to 1. As window watchdog timer is enabled, the down counter will start counting from 0x3F and cannot be stopped by software.

During WWDT down counting, the WWDT interrupt will happen if the counter value is equal to window watchdog timer compare value WINCMP (WWDTCR[21:16]) and INTEN(WWDTCR[1]) is set to 1. The WWDT reset will happen if the WWDT counter value reaches to 0. Before WWDT counter down to 0, software can write certain value (0x00005AA5) to register WWDTRLD to reload 0x3F to WWDT counter to prevent WWDT reset happen and this reload action only active when WWDT counter value is equal to or smaller than WINCMP. If software writes WWDTRLD during the period that WWDT counter larger than WINCMP, additional WWDT reset will happen to cause chip be reset.

When software writes certain value (0x00005AA5) to register WWDTRLD to reload WWDT counter, it need 3 window watchdog clocks to sync reload command to actually perform reload action. It means if software set window watchdog clock prescaler as divide 1, the compare value WINCMP (WWDTCR[21:16]) should larger than 2 or software will not able to reload WWDT counter before WWDT reset happened.

To prevent program run to unexpected code to disable window watchdog, the control register WWDTCR can only be write one time after chip is powered on or reset. Software cannot disable window watchdog, change pre-scale period or change window compare value as window watchdog is enabled by software unless chip is reset.

5.13.5 Register Map

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

Register	Offset	R/W	Description	Reset Value
WDT_BA = 0x4000_4100				
WWDTRLD	WWDT_BA+0x00	W	Window Watchdog Timer Reload Counter Register	0x0000_0000
WWDTCR	WWDT_BA+0x04	R/W	Window Watchdog Timer Control Register	0x003F_0800
WWDTSTS	WWDT_BA+0x08	R/W	Window Watchdog Timer Status Register	0x0000_0000
WWDTCVR	WWDT_BA+0x0C	R	Window Watchdog Counter Value Register	0x0000_003F

5.13.6 Register Description

Window Watchdog Timer Reload Counter Register (WWDTRLD)

Register	Offset	R/W	Description	Reset Value
WWDTRLD	WWDT_BA+0x00	W	Window Watchdog Timer Reload Counter Register	0x0000_0000

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

Bits	Description	
[31:0]	**WWDTRLD**	**WWDT Reload Counter Register** Writing 0x00005AA5 to this register will reload the Window Watchdog Timer counter value to 0x3F. **Note:** Software can only write WWDTRLD when WWDT counter value between 0 and WINCMP. If software writes WWDTRLD when WWDT counter value larger than WINCMP, WWDT will generate RESET signal.

Window Watchdog Timer Control Register (WWDTCR)

Register	Offset	R/W	Description	Reset Value
WWDTCR	WWDT_BA+0x04	R/W	Window Watchdog Timer Control Register	0x003F_0800

Note: This register can be written only once after chip is powered on or reset.

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

Bits	Description	
[31]	**DBGACK_WWDT**	**ICE debug mode acknowledge Disable** 0 = WWDT count stopped if system is in Debug mode. 1 = WWDT still count even system is in Debug mode.
[30:22]	**Reserved**	Reserved
[21:16]	**WINCMP**	**WWDT Window Compare Register** Set this register to adjust the valid reload window. **Note:** Software can only write WWDTRLD when WWDT counter value between 0 and WINCMP. If software writes WWDTRLD when WWDT counter value is larger than WWCMP, WWDT will generate RESET signal.
[15:12]	**Reserved**	Reserved
[11:8]	**PERIODSEL**	**WWDT Pre-scale Period Select** These three bits select the pre-scale for the WWDT counter period.

PERIOD SEL	Prescaler Value	Time-out Period	Time-out Interval (WWDT_CLK = 10 kHz)
0000	1	1 * 64 * T_{WWDT}	6.4 ms
0001	2	2 * 64 * T_{WWDT}	12.8 ms
0010	4	4 * 64 * T_{WWDT}	25.6 ms
0011	8	8 * 64 * T_{WWDT}	51.2 ms
0100	16	16 * 64 * T_{WWDT}	102.4 ms
0101	32	32 * 64 * T_{WWDT}	204.8 ms

<table>
<tr><td></td><td></td><td>
<table>
<tr><td>0110</td><td>64</td><td>$64 * 64 * T_{WWDT}$</td><td>409.6 ms</td></tr>
<tr><td>0111</td><td>128</td><td>$128 * 64 * T_{WWDT}$</td><td>819.2 ms</td></tr>
<tr><td>1000</td><td>192</td><td>$192 * 64 * T_{WWDT}$</td><td>1.2288 s</td></tr>
<tr><td>1001</td><td>256</td><td>$256 * 64 * T_{WWDT}$</td><td>1.6384 s</td></tr>
<tr><td>1010</td><td>384</td><td>$384 * 64 * T_{WWDT}$</td><td>2.4576 s</td></tr>
<tr><td>1011</td><td>512</td><td>$512 * 64 * T_{WWDT}$</td><td>3.2768 s</td></tr>
<tr><td>1100</td><td>768</td><td>$768 * 64 * T_{WWDT}$</td><td>4.9152 s</td></tr>
<tr><td>1101</td><td>1024</td><td>$1024 * 64 * T_{WWDT}$</td><td>6.5536 s</td></tr>
<tr><td>1110</td><td>1536</td><td>$1536 * 64 * T_{WWDT}$</td><td>9.8304 s</td></tr>
<tr><td>1111</td><td>2048</td><td>$2048 * 64 * T_{WWDT}$</td><td>13.1072 s</td></tr>
</table>
</td></tr>
<tr><td>[7:2]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[1]</td><td><b>WWDTIE</b></td><td><b>WWDT Interrupt Enable</b>
Set this bit to enable the Watchdog timer interrupt function.
0 = Watchdog timer interrupt function Disabled.
1 = Watchdog timer interrupt function Enabled.</td></tr>
<tr><td>[0]</td><td><b>WWDTEN</b></td><td><b>WWDT Enable</b>
Set this bit to enable the Window Watchdog timer.
0 = Window Watchdog timer function Disabled.
1 = Window Watchdog timer function Enabled.</td></tr>
</table>

Window Watchdog Timer Status Register (WWDTSR)

Register	Offset	R/W	Description	Reset Value
WWDTSR	WWDT_BA+0x08	R/W	Window Watchdog Timer Status Register	0x0000_0000

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

Bits	Description	
[31:2]	**Reserved**	Reserved
[1]	**WWDTRF**	**WWDT Reset Flag** When WWDT counter counts down to 0 or writes WWDTRLD during WWDT counter larger than WINCMP, chip will be reset and this bit is set to 1. Software can write 1 to clear this bit to 0.
[0]	**WWDTIF**	**WWDT Compare Match Interrupt Flag** When WWCMP matches the WWDT counter, this bit is set to 1. This bit will be cleared by software write 1 to this bit.

Window Watchdog Timer Counter Value Register (WWDTCVR)

Register	Offset	R/W	Description	Reset Value
WWDTCVR	WWDT_BA+0x0C	R	Window Watchdog Timer Counter Value Register	0x0000_003F

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

Bits	Description	
[31:6]	**Reserved**	Reserved
[5:0]	**WWDTCVAL**	**WWDT Counter Value** This register reflects the counter value of window watchdog. This register is read only.

5.14 UART Interface Controller (UART)

This chip provides up to two channels of Universal Asynchronous Receiver/Transmitters (UART). The UART controller UART0 and UART1 support flow control function.

5.14.1 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 and RS-485 mode functions. Each UART channel supports six 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) and Buffer error interrupt (INT_BUF_ERR). Interrupts of UART0 number is 12 (vector number is 28); Interrupt number 13 (vector number is 29) supports UART1 interrupt. Refer to Nested Vectored Interrupt Controller chapter for System Interrupt Map.

The UART0 and UART1 are built-in with a 16-byte transmitter FIFO (TX_FIFO) and a 16-byte receiver FIFO (RX_FIFO) that reduces the number of interrupts presented to the CPU. The CPU can read the status of the UART at any time during 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 5-11 lists the equations in the various conditions and Table 5-12 lists the UART baud rate settings.

Mode	DIV_X_EN	DIV_X_ONE	Divider X	BRD	Baud Rate Equation
0	0	0	Don’t care	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 >=9

Table 5-11 UART Baud Rate Equation

System Clock = Internal 22.1184 MHz High Speed Oscillator						
Baud Rate	Mode 0		Mode 1		Mode 2	
	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 5-12 UART Baud Rate Setting Table

The UART0 and UART1 controllers support auto-flow control function that uses two low-level signals (the level can be change by configure UA_MSR and UA_MCR register), /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 is equal to 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.

For NuMicro™ NUC123 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.

5.14.2 Features

- Full duplex, asynchronous communications
- Separates receive / transmit 16 bytes entry FIFO for data payloads
- Supports hardware auto flow control/flow control function (CTS, RTS) and programmable RTS flow control trigger level.
- Programmable receiver buffer trigger level
- Supports programmable baud-rate generator for each channel individually
- Supports CTS wake-up function.
- Supports 8-bit receiver buffer time out detection function
- UART0/UART1 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
- Supports 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
- Supports IrDA SIR function mode
  - Supports for 3-/16-bit duration for normal mode
- Supports RS-485 function mode.
  - Supports RS-485 9-bit mode
  - Supports hardware or software direct enable control provided by RTS pin

5.14.3 Block Diagram

The UART clock control and block diagram are shown in Figure 5-74 and Figure 5-75.

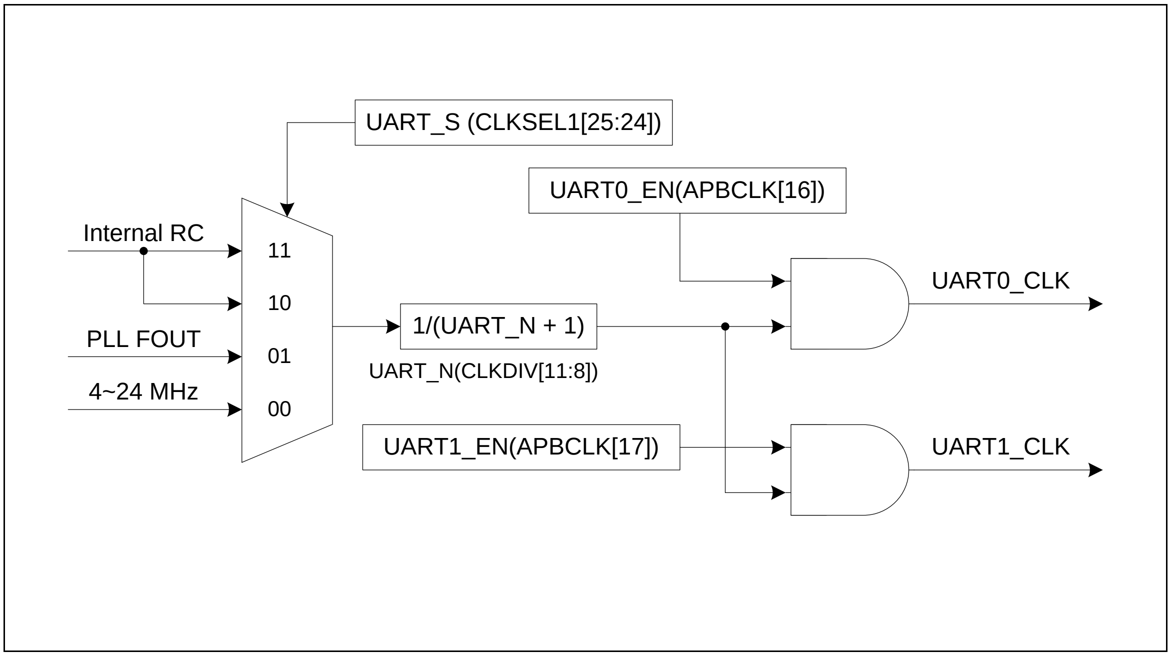


Figure 5-74 UART Clock Control Diagram

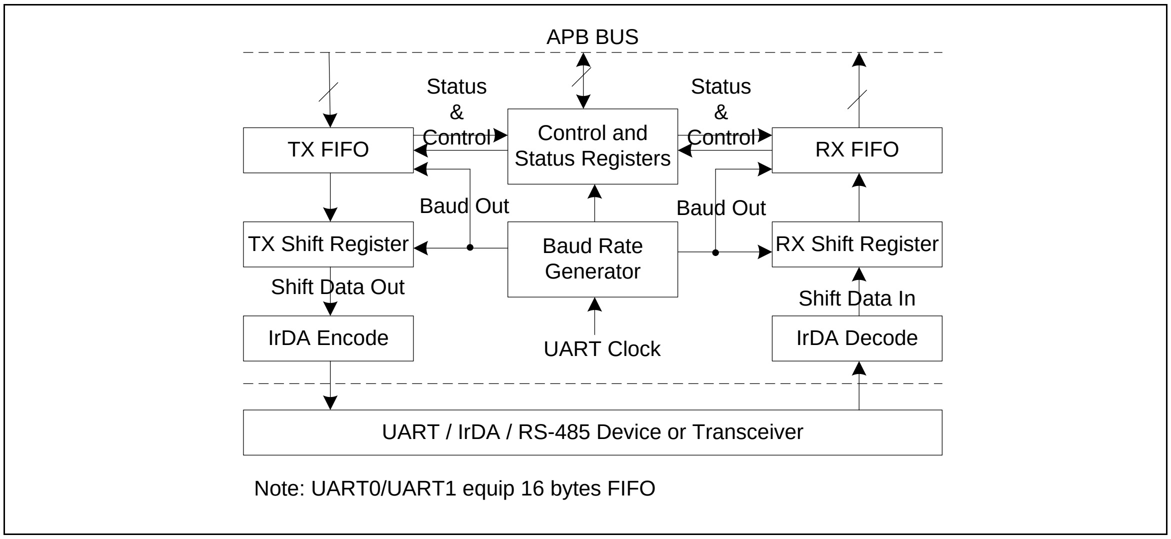


Figure 5-75 UART Block Diagram

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

RX_FIFO
The receiver is buffered with a 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.

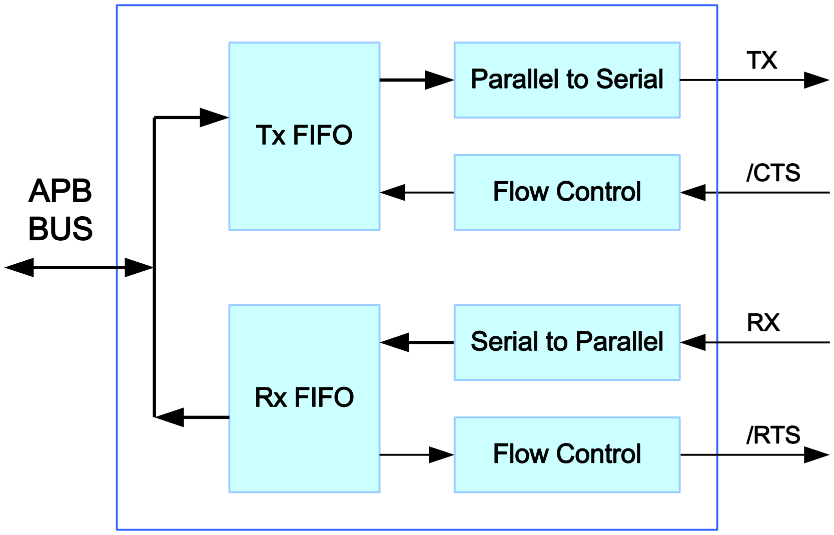
Baud Rate Generator
Divide the external clock by the divisor to get the desired baud rate clock. Refer to baud rate equation.

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 six 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) and Buffer error interrupt (INT_BUF_ERR).

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



Note: It is supported in UART0 and UART1.

Figure 5-76 Auto Flow Control Block Diagram

5.14.4 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.

In IrDA mode, the UA_BAUD [DIV_X_EN] register must be disabled.

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

The following diagram demonstrates the IrDA control block diagram.

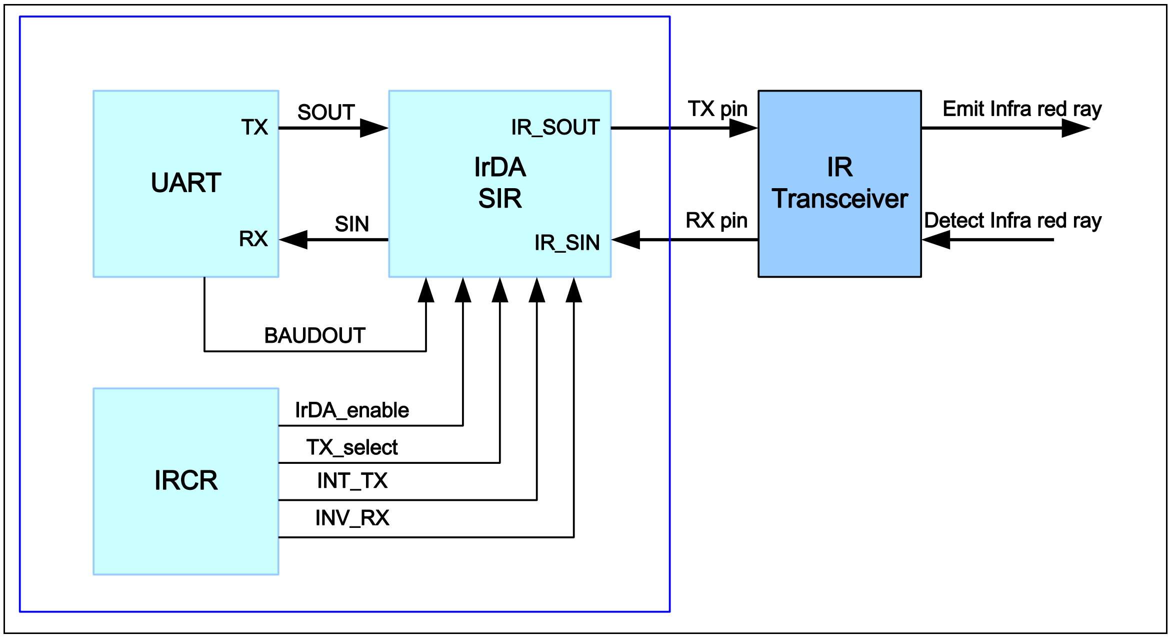


Figure 5-77 IrDA Block Diagram

5.14.4.1 *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.

5.14.4.2 *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

5.14.4.3 *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:

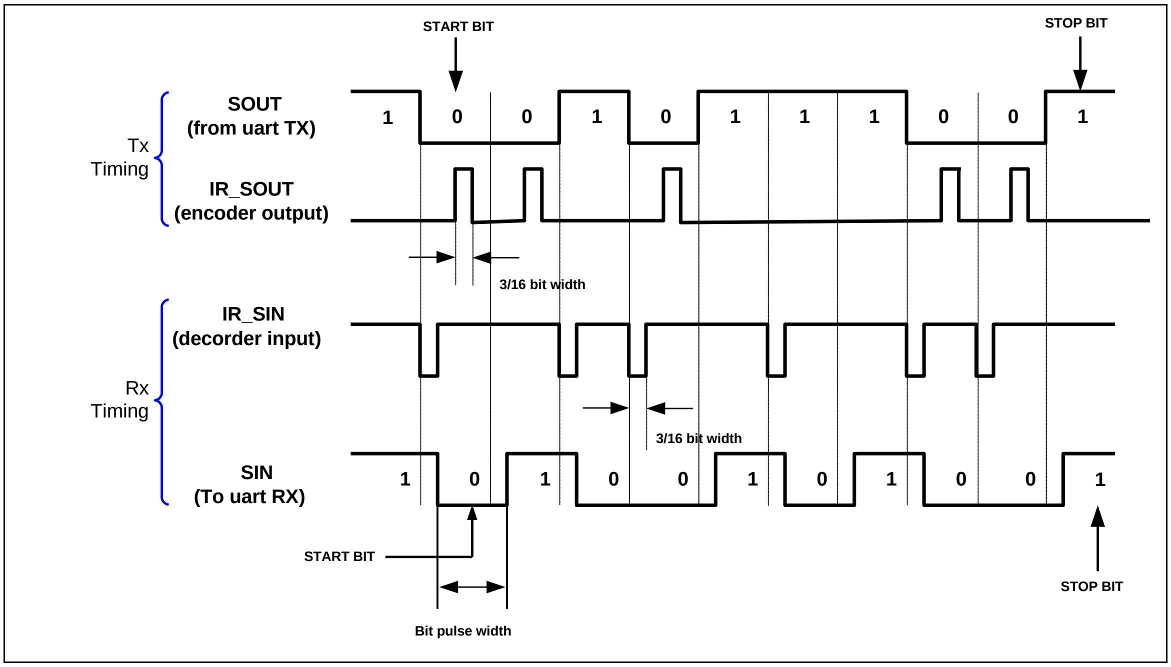


Figure 5-78 IrDA TX/RX Timing Diagram

5.14.5 RS-485 Function Mode

The UART supports **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.

In RS-485 mode, the controller can configure 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_ALT_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, 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[RX_DIS] then enable UA_ALT_CSR[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 disables UART_FCR [RX_DIS] and then enables UA_ALT_CSR [RS485_NMM] and the receiver will receive 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_FCR [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_FCR [RX_DIS] register, when a next address byte be detected, the controller will clear the UA_FCR [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_ALT_CSR [ADDR_MATCH] value. The address byte data will be stored in the RX-FIFO and the following received byte data (bit9 = 0) will be accepted and stored in the RX-FIFO until a next address byte data be detected (bit9 =1) and the address byte data do not match the UA_ALT_CSR [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 and the LEV_RTS = 0) enables the RS-485 driver. Setting the RTS line to low (logic 0 and the LEV_RTS = 1) 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.

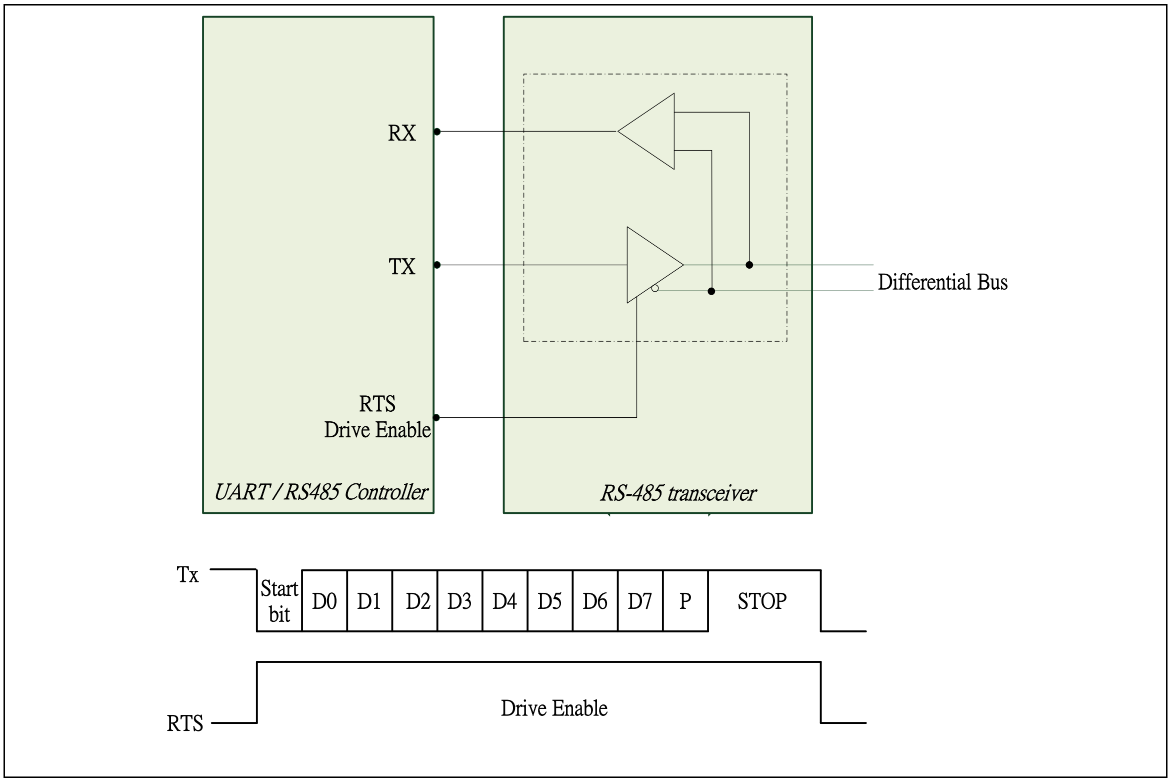


Figure 5-79 Structure of RS-485 Frame

5.14.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
UART Base Address : **Channel0 : UART0_BA = 0x4005_0000** **Channel1 : UART1_BA = 0x4015_0000**				
UA_RBR	UART0_BA+0x00	R	UART0 Receive Buffer Register	Undefined
	UART1_BA+0x00	R	UART1 Receive Buffer Register	Undefined
UA_THR	UART0_BA+0x00	W	UART0 Transmit Holding Register	Undefined
	UART1_BA+0x00	W	UART1 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
UA_FCR	UART0_BA+0x08	R/W	UART0 FIFO Control Register	0x0000_0101
	UART1_BA+0x08	R/W	UART1 FIFO Control Register	0x0000_0101
UA_LCR	UART0_BA+0x0C	R/W	UART0 Line Control Register	0x0000_0000
	UART1_BA+0x0C	R/W	UART1 Line Control Register	0x0000_0000
UA_MCR	UART0_BA+0x10	R/W	UART0 Modem Control Register	0x0000_0200
	UART1_BA+0x10	R/W	UART1 Modem Control Register	0x0000_0200
UA_MSR	UART0_BA+0x14	R/W	UART0 Modem Status Register	0x0000_0110
	UART1_BA+0x14	R/W	UART1 Modem Status Register	0x0000_0110
UA_FSR	UART0_BA+0x18	R/W	UART0 FIFO Status Register	0x1040_4000
	UART1_BA+0x18	R/W	UART1 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
UA_TOR	UART0_BA+0x20	R/W	UART0 Time Out Register	0x0000_0000
	UART1_BA+0x20	R/W	UART1 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
UA_IRCR	UART0_BA+0x28	R/W	UART0 IrDA Control Register	0x0000_0040
	UART1_BA+0x28	R/W	UART1 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
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

5.14.7 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

31	30	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	Description	
[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

31	30	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	Description	
[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

31	30	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		
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	Description	
[31:16]	**Reserved**	Reserved
[15]	**DMA_RX_EN**	**RX DMA Enable** This bit can enable or disable RX DMA service. 1 = RX DMA Enabled. 0 = RX DMA Disabled.
[14]	**DMA_TX_EN**	**TX DMA Enable** This bit can enable or disable TX DMA service. 1 = TX DMA Enabled. 0 = TX DMA Disabled.
[13]	**AUTO_CTS_EN**	**CTS Auto Flow Control Enable** 1 = CTS auto flow control Enabled. 0 = CTS auto flow control Disabled. 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** 1 = RTS auto flow control Enabled. 0 = RTS auto flow control Disabled. When RTS auto-flow is enabled, if the number of bytes in the RX FIFO is equal to the UA_FCR [RTS_TRI_LEV], the UART will de-assert RTS signal.

[11]	**TIME_OUT_EN**	**Time Out Counter Enable** 1 = Time-out counter Enabled. 0 = Time-out counter Disabled.
[10:7]	**Reserved**	Reserved
[6]	**WAKE_EN**	**UART Wake-up Function Enable** 0 = UART wake-up function Disabled. 1 = UART wake-up function Enabled, when 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 = INT_BUF_ERR Enabled. 0 = INT_BUF_ERR Masked off.
[4]	**RTO_IEN**	**RX Time Out Interrupt Enable** 1 = INT_TOUT Enabled. 0 = INT_TOUT Masked off.
[3]	**MODEM_IEN**	**Modem Status Interrupt Enable** 1 = INT_MODEM Enabled. 0 = INT_MODEM Masked off.
[2]	**RLS_IEN**	**Receive Line Status Interrupt Enable** 1 = INT_RLS Enabled. 0 = INT_RLS Masked off.
[1]	**THRE_IEN**	**Transmit Holding Register Empty Interrupt Enable** 1 = INT_THRE Enabled. 0 = INT_THRE Masked off.
[0]	**RDA_IEN**	**Receive Data Available Interrupt Enable**. 1 = INT_RDA Enabled. 0 = INT_RDA Masked off.

FIFO Control Register (UA_FCR)

Register	Offset	R/W	Description	Reset Value
UA_FCR	UART0_BA+0x08	R/W	UART0 FIFO Control Register	0x0000_0101
	UART1_BA+0x08	R/W	UART1 FIFO Control Register	0x0000_0101

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

<table>
<tr><th>Bits</th><th colspan="2">Description</th></tr>
<tr><td>[31:20]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[19:16]</td><td><b>RTS_TRI_LEV</b></td><td><b>RTS Trigger Level for Auto-flow Control Use</b>
<table>
<tr><th>RTS_TRI_LEV</th><th>Trigger Level (Bytes)</th></tr>
<tr><td>0000</td><td>01</td></tr>
<tr><td>0001</td><td>04</td></tr>
<tr><td>0010</td><td>08</td></tr>
<tr><td>0011</td><td>14</td></tr>
<tr><td>others</td><td>14</td></tr>
</table>
<b>Note:</b> This field is used for auto RTS flow control.</td></tr>
<tr><td>[15:9]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[8]</td><td><b>RX_DIS</b></td><td><b>Receiver Disable Register</b>
The receiver is enabled or disabled.
1 = Receiver Disabled.
0 = Receiver Enabled.
<b>Note:</b> This field is used for RS-485 Normal Multi-drop mode. It should be programmed before UA_ALT_CSR [RS-485_NMM] is programmed.</td></tr>
<tr><td>[7:4]</td><td><b>RFITL</b></td><td><b>RX FIFO Interrupt (INT_RDA) Trigger Level</b>
When the number of bytes in the received FIFO is equal to the RFITL, the RDA_IF will be set (if UA_IER [RDA_IEN] is enabled, an interrupt will be generated).</td></tr>
</table>

<table>
<tr><td></td><td></td><td><table><tr><th>RFITL</th><th>INTR_RDA Trigger Level (Bytes)</th></tr><tr><td>0000</td><td>01</td></tr><tr><td>0001</td><td>04</td></tr><tr><td>0010</td><td>08</td></tr><tr><td>0011</td><td>14</td></tr><tr><td>others</td><td>14</td></tr></table></td></tr>
<tr><td>[3]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[2]</td><td><b>TFR</b></td><td><b>TX Field Software Reset</b>
When TX_RST is set, all the byte in the transmit FIFO and TX internal state machine are cleared.
1 = Reset the TX internal state machine and pointers.
0 = No effect.
Note: This bit will be automatically cleared at least 3 UART engine clock cycles.</td></tr>
<tr><td>[1]</td><td><b>RFR</b></td><td><b>RX Field Software Reset</b>
When RX_RST is set, all the byte in the receiver FIFO and RX internal state machine are cleared.
1 = Reset the RX internal state machine and pointers.
0 = No effect.
<b>Note:</b> This bit will be automatically cleared at least 3 UART engine clock cycles.</td></tr>
<tr><td>[0]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
</table>

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

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

Bits	Description	
[31:7]	**Reserved**	Reserved
[6]	**BCB**	**Break Control Bit** 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" are generated when 6-, 7- and 8-bit word length is selected.
[1:0]	**WLS**	**Word Length Selection**

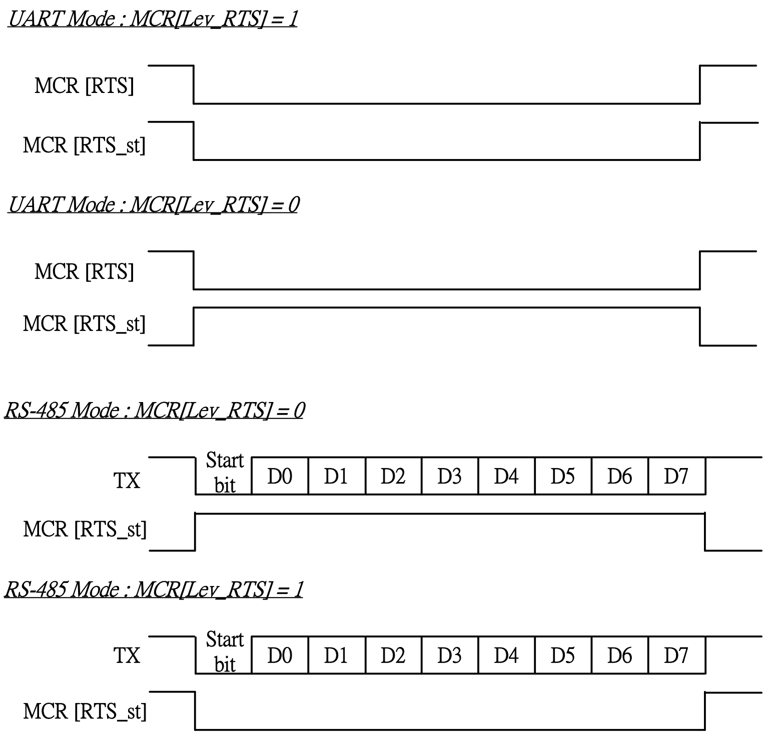
WLS[1:0]	Character Length
00	5-bit
01	6-bit
10	7-bit
11	8-bit

MODEM Control Register (UA_MCR)

Register	Offset	R/W	Description	Reset Value
UA_MCR	UART0_BA+0x10	R/W	UART0 Modem Control Register	0x0000_0200
	UART1_BA+0x10	R/W	UART1 Modem Control Register	0x0000_0200

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

Bits	Description	
[31:14]	**Reserved**	Reserved
[13]	**RTS_ST**	**RTS Pin State (Read Only)** This bit is the output pin status of RTS.
[12:10]	**Reserved**	Reserved

[9]	**LEV_RTS**	**RTS Trigger Level** This bit can change the RTS trigger level. 1= High level triggered 0= Low level triggered  UART Mode : MCR[Lev_RTS] = 1 MCR [RTS] MCR [RTS_st] UART Mode : MCR[Lev_RTS] = 0 MCR [RTS] MCR [RTS_st] RS-485 Mode : MCR[Lev_RTS] = 0 TX Start bit D0 D1 D2 D3 D4 D5 D6 D7 MCR [RTS_st] RS-485 Mode : MCR[Lev_RTS] = 1 TX Start bit D0 D1 D2 D3 D4 D5 D6 D7 MCR [RTS_st]
[8:2]	**Reserved**	Reserved
[1]	**RTS**	**RTS (Request-To-Send) Signal** 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)

Register	Offset	R/W	Description	Reset Value
UA_MSR	UART0_BA+0x14	R/W	UART0 Modem Status Register	0x0000_01X0
	UART1_BA+0x14	R/W	UART1 Modem Status Register	0x0000_01X0

31	30	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	Description	
[31:9]	**Reserved**	Reserved
[8]	**LEV_CTS**	**CTS Trigger Level** 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)** This bit is the pin status of CTS.
[3:1]	**Reserved**	Reserved
[0]	**DCTSF**	**Detect CTS State Change Flag (Read Only)** 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

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	Description	
[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, it is cleared by hardware.
[22]	**TX_EMPTY**	**Transmitter FIFO Empty (Read Only)** This bit indicates TX FIFO is 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, it 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

31	30	29	28	27	26	25	24
Reserved		HW_BUF_ERR_INT	HW_TOUT_INT	HW_MODEM_INT	HW_RLS_INT	Reserved	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved		HW_BUF_ERR_IF	HW_TOUT_IF	HW_MODEM_IF	HW_RLS_IF	Reserved	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		BUF_ERR_INT	TOUT_INT	MODEM_INT	RLS_INT	THRE_INT	RDA_INT
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		BUF_ERR_IF	TOUT_IF	MODEM_IF	RLS_IF	THRE_IF	RDA_IF

Bits	Description	
[31:30]	**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 = 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 = 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)** This bit is set if MODEM_IEN and HW_MODEM_IF are both set to 1. 1 = 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 = RLS interrupt is generated in DMA mode. 0 = No RLS interrupt is generated in DMA mode.
[25:22]	**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)** 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:** 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.
[17:14]	**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 = 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 = Tout interrupt is generated. 0 = No Tout interrupt is generated.
[11]	**MODEM_INT**	**MODEM Status Interrupt Indicator (Read Only).** This bit is set if MODEM_IEN and MODEM_IF are both set to 1. 1 = 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 = 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 = 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 = RDA interrupt is generated 0 = No RDA interrupt is generated
[7:6]	**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)** 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:** 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 is equal to the RFITL, 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
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 Time-out 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 5-13 UART Interrupt Sources and Flags Table in DMA Mode

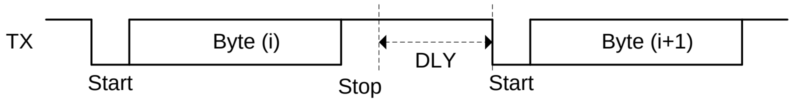
UART Interrupt Source	Interrupt Enable Bit	Interrupt Indicator to Interrupt Controller	Interrupt Flag	Flag Cleared by
Buffer Error Interrupt INT_BUF_ERR	BUF_ERR_IEN	BUF_ERR_INT	**BUF_ERR_IF** = (TX_OVER_IF or RX_OVER_IF)	Write '1' to TX_OVER_IF/ RX_OVER_IF
RX Time-out Interrupt INT_TOUT	RTO_IEN	TOUT_INT	**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 5-14 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

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

Bits	Description	
[31:16]	**Reserved**	Reserved
[15:8]	**DLY**	**TX Delay time value** This field is used to programming the transfer delay time between the last stop bit and next start bit.  TX Byte (i) DLY Byte (i+1) Start Stop Start
[7:0]	**TOIC**	**Time Out Interrupt Comparator** The time out counter resets and starts counting (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. Thus, for example, if TOIC is set as 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

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	Description	
[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 = Divider X Enabled (the equation of M = X+1, but DIVIDER_X [27:24] must >= 8). 0 = Divider X Disabled (the equation of M = 16) Refer to the table below for more information. **Note:** In IrDA mode, this bit must be disabled.
[28]	**DIV_X_ONE**	**Divider X Equal to 1** 1 = Divider M = 1 (the equation of M = 1, but BRD [15:0] must >= 8). 0 = Divider M = X (the equation of M = X+1, but DIVIDER_X[27:24] must >= 8) Refer to the Table 5-15 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 indicates the baud rate divider.

Mode	DIV_X_EN	DIV_X_ONE	DIVIDER X	BRD	Baud Rate Equation
0	Disable	0	Don’t care	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 >=8

Table 5-15 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

31	30	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	Description	
[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: In IrDA mode, the UA_BAUD [DIV_X_EN] register must be disabled (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

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**
Reserved							

Bits	Description	
[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 used to enable RS-485 address detection mode. 1 = Address detection mode Enabled. 0 = Address detection mode Disabled. **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 = RS-485 Auto Direction Operation mode (AUD) Enabled. 0 = RS-485 Auto Direction Operation mode (AUD) Disabled. **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 = RS-485 Auto Address Detection Operation mode (AAD) Enabled. 0 = RS-485 Auto Address Detection Operation mode (AAD) Disabled. **Note:** It can't be active with RS-485_NMM operation mode.
[8]	**RS485_NMM**	**RS-485 Normal Multi-drop Operation Mode (NMM)**

		1 = RS-485 Normal Multi-drop Operation mode (NMM) Enabled. 0 = RS-485 Normal Multi-drop Operation mode (NMM) Disabled. **Note:** It can’t be active with RS-485_AAD operation mode.
[7:0]	**Reserved**	Reserved

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

31	30	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	Description	
[31:2]	**Reserved**	Reserved
[1:0]	**FUN_SEL**	**Function Selection Enable** 00 = UART Function 01 = Reserved 10 = IrDA Function Enabled 11 = RS-485 Function Enabled

5.15 PS/2 Device Controller (PS2D)

5.15.1 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.

5.15.2 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

5.15.3 Block Diagram

The PS/2 device controller consists of APB interface and timing control logic for DATA and CLK lines.

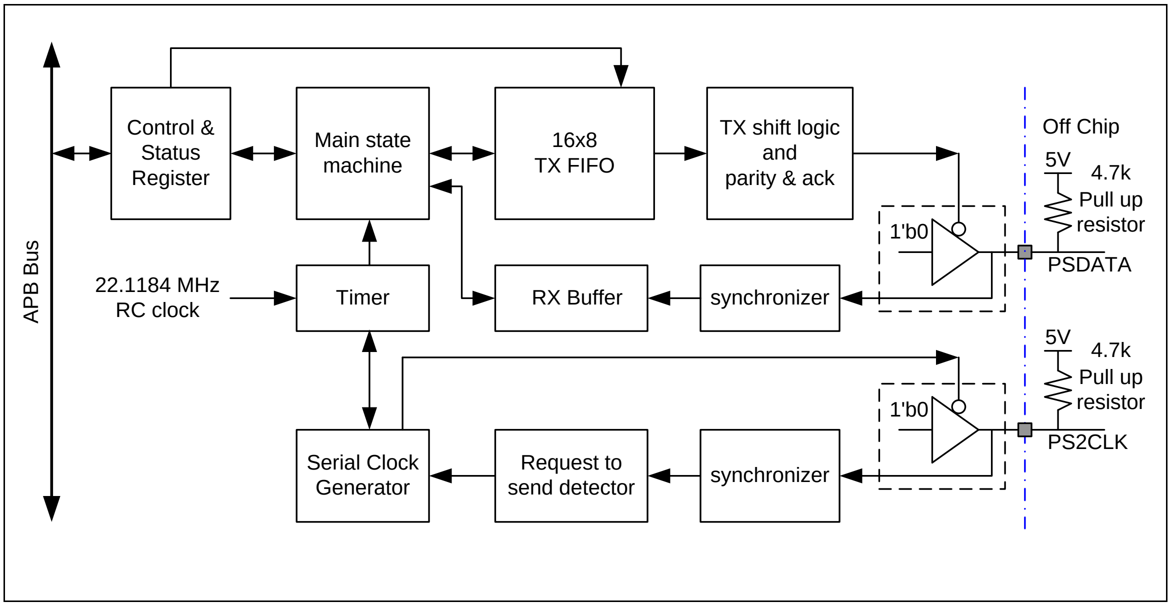


Figure 5-80 PS/2 Device Block Diagram

5.15.4 Functional Description

5.15.4.1 *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 adds 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, which is illustrated in Figure 5-81.

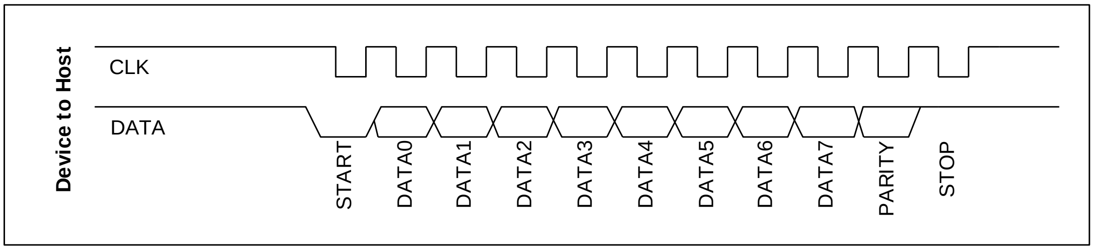


Figure 5-81 Data Format of Device-to-Host

Host-to-Device:

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, and then release CLK

The device should check for the 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.

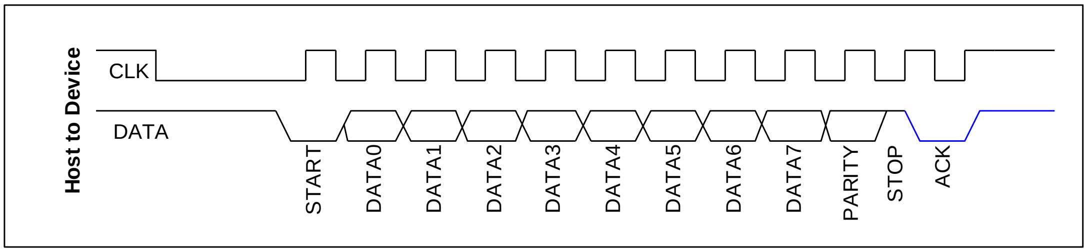


Figure 5-82 Data Format of Host-to-Device

The detailed host and the device DATA and CLK timing for communication is shown below:

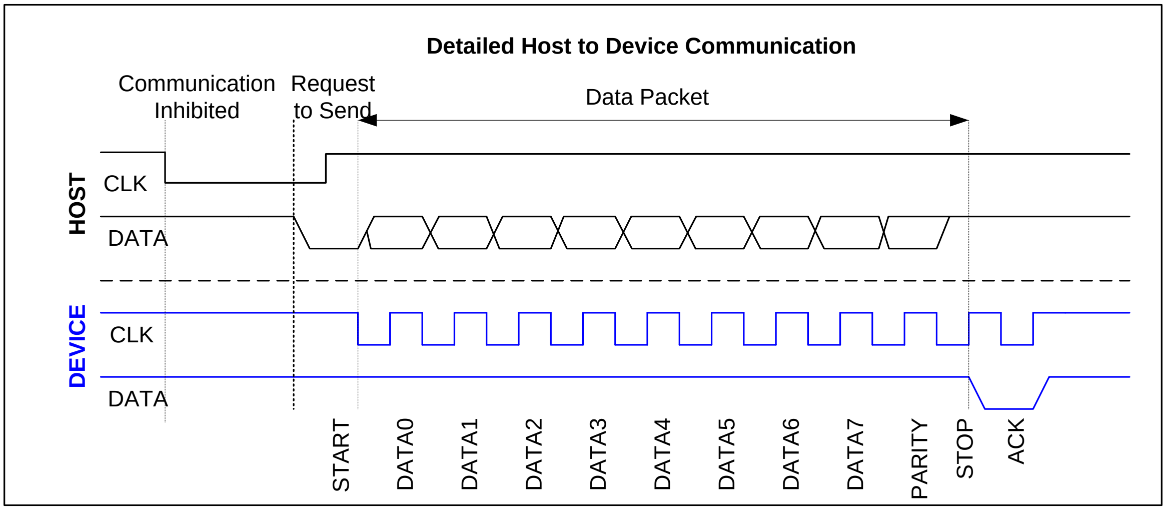


Figure 5-83 PS/2 Bit Data Format

5.15.4.2 *PS/2 Bus Timing Specification*

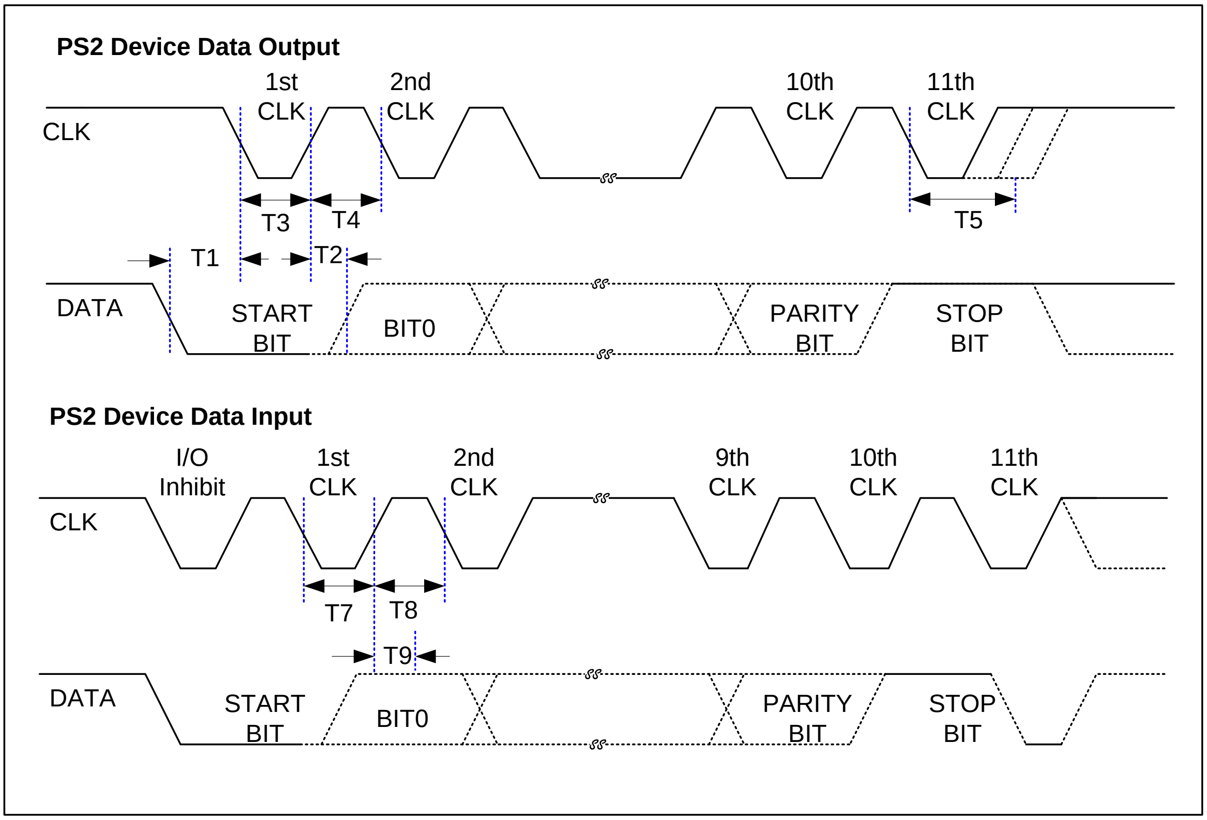


Figure 5-84 PS/2 Bus Timing

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, used for time auxiliary device sample DATA	5us	25us

5.15.4.3 *TX FIFO Operation*

Writing PS2TXDATA0 register starts device to host communication. Software is required to define TXFIFO depth before writing transmission data to TX FIFO. 1st START bit is sent to PS/2 bus 100us after software writes TX FIFO, if there is more than 4 bytes data need to be sent, Software can write residual data to PS2TXDATA1-3 before 4th byte transmit complete. A time delay 100us is added between two consecutive bytes.

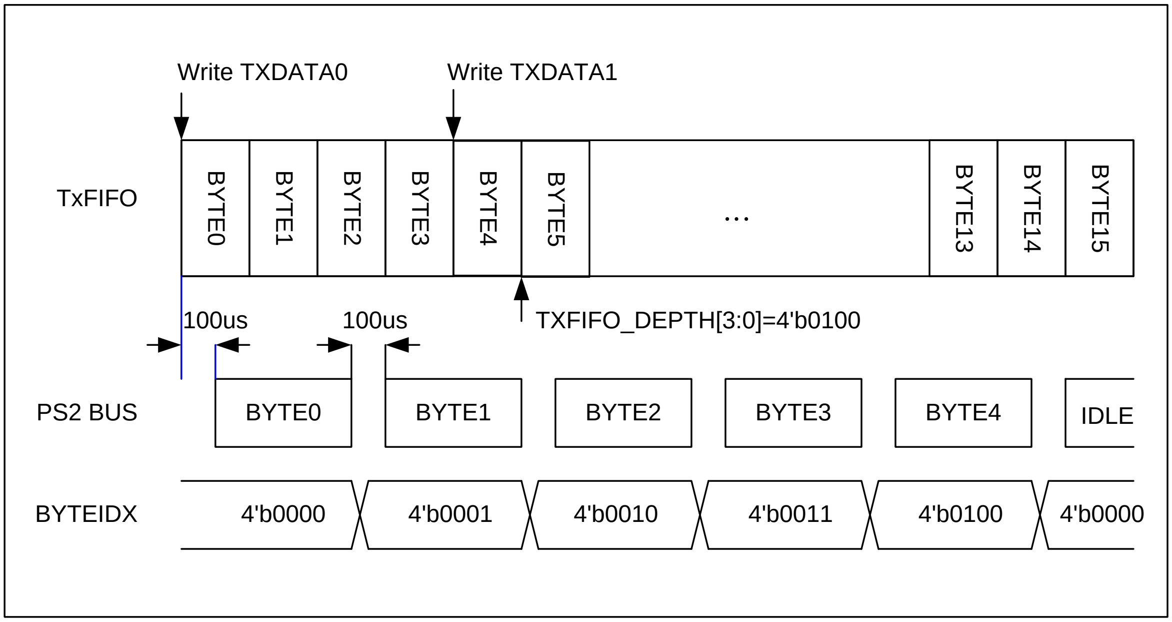


Figure 5-85 PS/2 Data Format

5.15.5 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

5.15.6 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	Description	
[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 software. 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 sends acknowledge to host at 12th clock for host to device communication.
[6:3]	**TXFIFODIPTH**	**Transmit Data FIFO Depth** There is 16-byte buffer for data transmit. Software can define the FIFO depth from 1 to 16 bytes depending on the application. 0 = 1 byte 1 = 2 bytes … 14 = 15 bytes 15 = 16 bytes
[2]	**RXINTEN**	**Enable Receive Interrupt** 1 = Data receive complete interrupt Enabled. 0 = Data receive complete interrupt Disabled.
[1]	**TXINTEN**	**Enable Transmit Interrupt** 1 = Data transmit complete interrupt Enabled. 0 = Data transmit complete interrupt Disabled.
[0]	**PS2EN**	**Enable PS/2 Device** Enable PS/2 device controller. 1 = Enabled. 0 = Disabled.

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	Description	
[31:0]	**PS2TXDATAx**	**Transmit data** Write data to this register starts device to host communication if bus is in IDLE state. Software 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	Description	
[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">Description</th></tr>
<tr><td>[31:12]</td><td><b>Reserved</b></td><td>Reserved</td></tr>
<tr><td>[11:8]</td><td><b>BYTEIDX</b></td><td><b>Byte Index</b>
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><b>TXEMPTY</b></td><td><b>TX FIFO Empty</b>
When software 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>
</table>

[6]	**RXOVF**	**RX Buffer Overwrite** 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, software overrides PS2CLK to send clock till PS2DATA release to high state. After that, device sends a “Resend” command to host. 1 = Frame error occurred . 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	Description	
[31:3]	**Reserved**	Reserved
[1]	**TXINT**	**Transmit Interrupt** This bit is set to 1 after STOP bit is transmitted. Interrupt occurs 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.

5.16 I^2S Controller (I^2S)

5.16.1 Overview

The I^2S 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.

5.16.2 Features

- Operated as either master or slave
- Capable of handling 8-, 16-, 24- and 32-bit word sizes
- Supports Mono and stereo audio data
- Supports I^2S and MSB justified data format
- Provides two 8 word FIFO data buffers, one for transmitting and the other for receiving
- Generates interrupt requests when buffer levels cross a programmable boundary
- Two DMA requests, one for transmitting and the other for receiving

5.16.3 Block Diagram

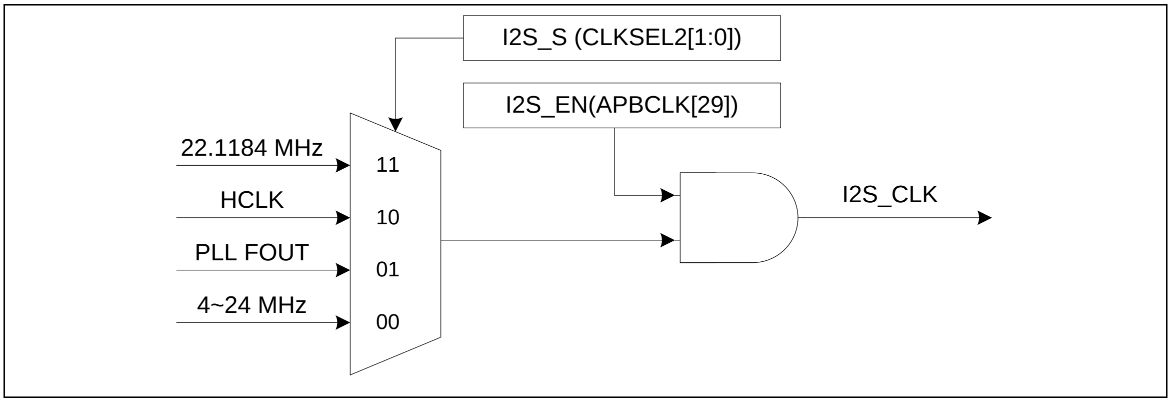


Figure 5-86 I^{2}S Clock Control Diagram

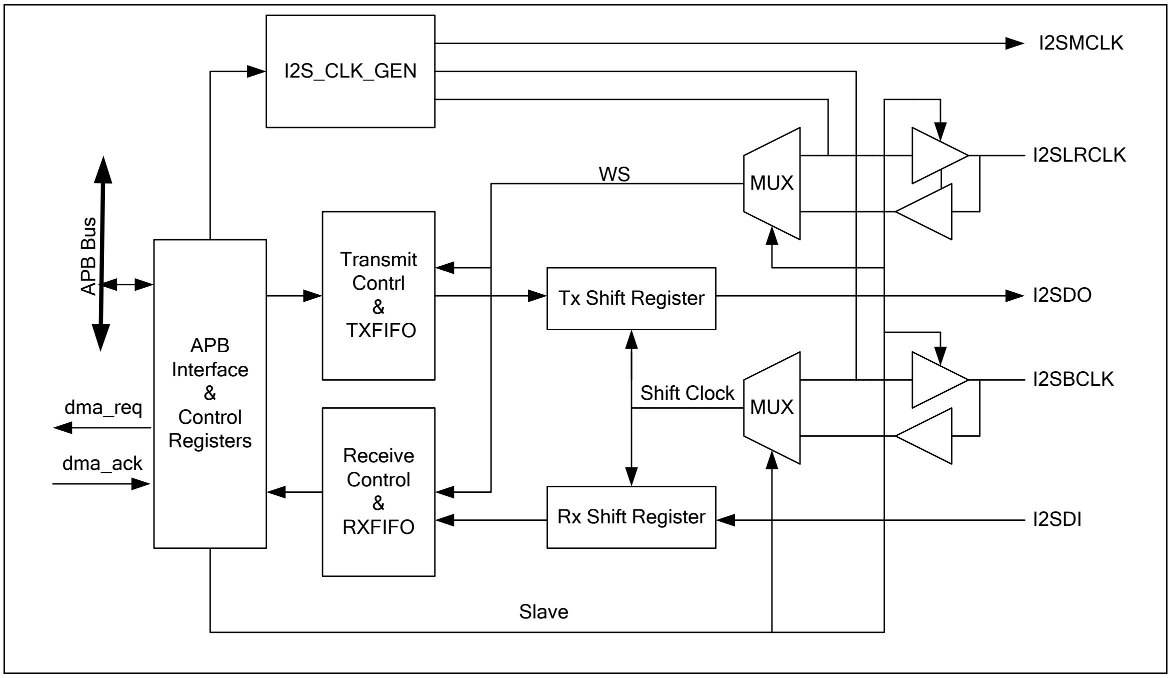


Figure 5-87 I^{2}S Controller Block Diagram

5.16.4 Functional Description

5.16.4.1 *I²S Operation*

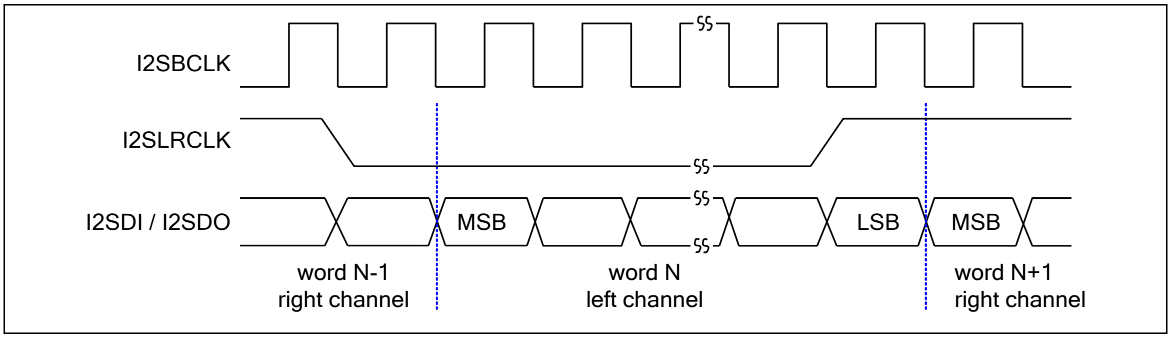


Figure 5-88 I²S Bus Timing Diagram (Format = 0)

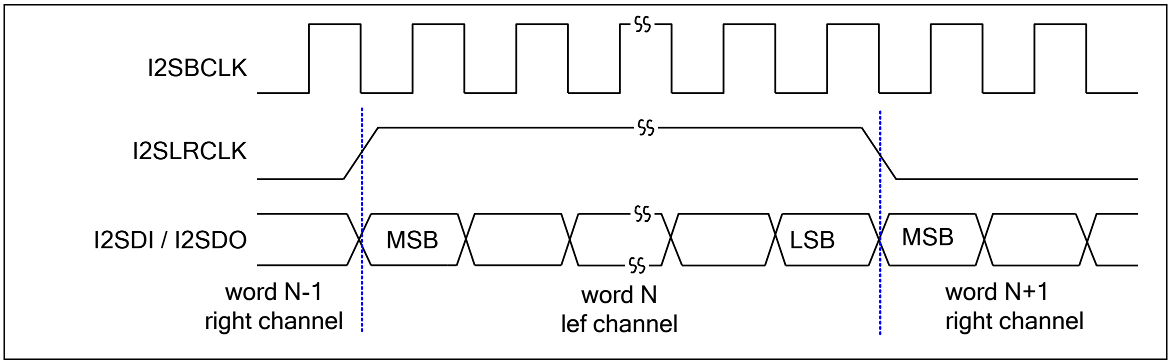


Figure 5-89 MSB Justified Timing Diagram (Format = 1)

5.16.4.2 *FIFO operation*

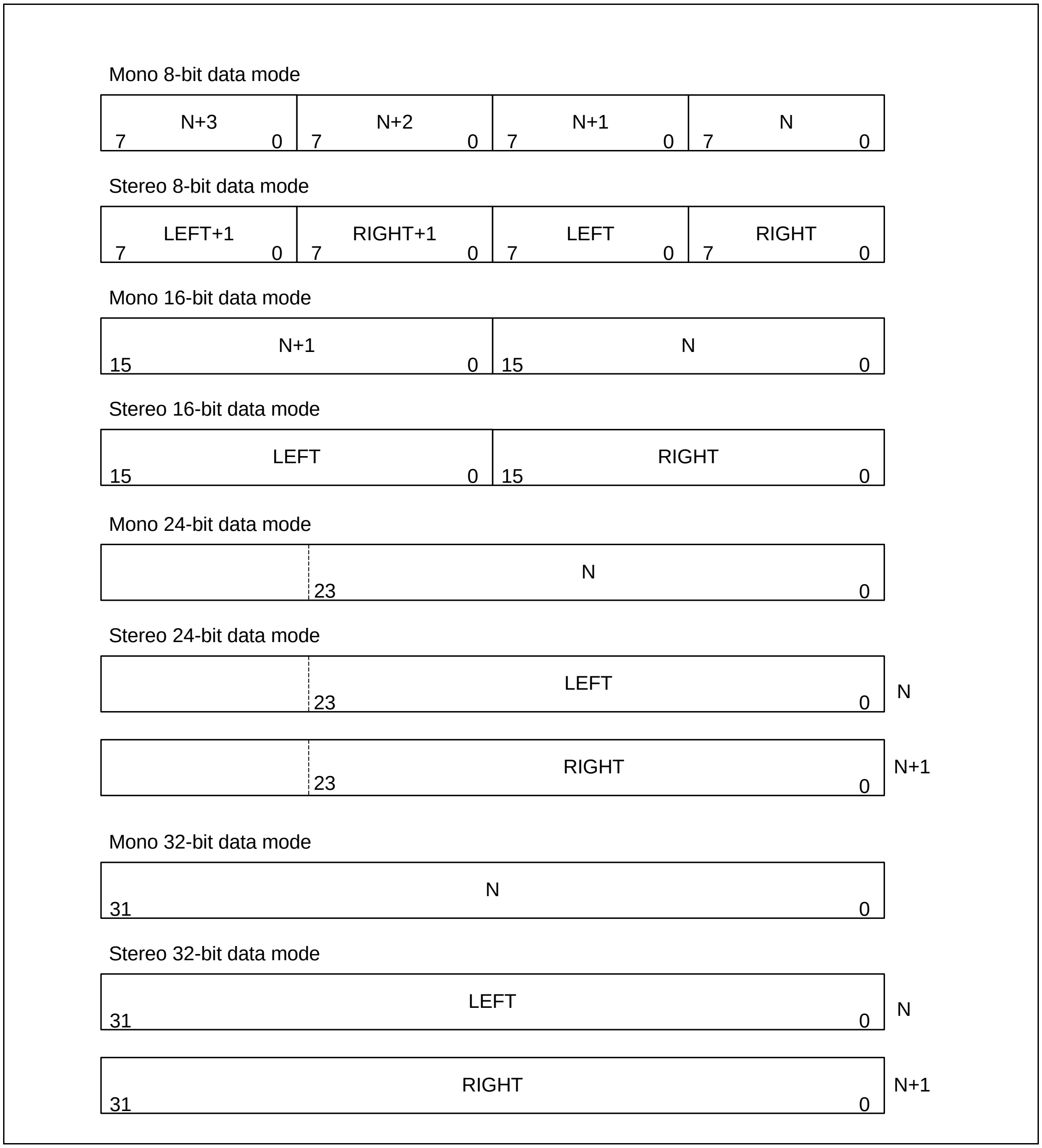


Figure 5-90 FIFO Contents for Various I^{2}S Modes

5.16.5 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^{2}S Control Register	0x0000_0000
I2S_CLKDIV	I2S_BA+0x04	R/W	I^{2}S Clock Divider Register	0x0000_0000
I2S_IE	I2S_BA+0x08	R/W	I^{2}S Interrupt Enable Register	0x0000_0000
I2S_STATUS	I2S_BA+0x0C	R/W	I^{2}S Status Register	0x0014_1000
I2S_TXFIFO	I2S_BA+0x10	R/W	I^{2}S Transmit FIFO Register	0x0000_0000
I2S_RXFIFO	I2S_BA+0x14	R/W	I^{2}S Receive FIFO Register	0x0000_0000

5.16.6 Register Description

I^{2}S Control Register (I2S_CON)

Register	Offset	R/W	Description	Reset Value
I2S_CON	I2S_BA+0x00	R/W	I^{2}S Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							**PCM**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
RXLCH	**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	Description	
[31:25]	**Reserved**	Reserved
[24]	**PCM**	**PCM Interface Enable** 1 = PCM interface. 0 = I^{2}S Interface.
[23]	**RXLCH**	**Receive Left Channel Enable** When monaural format is selected (MONO = 1), I^{2}S will receive right channel data if RXLCH is set to 0, and receive left channel data if RXLCH is set to 1. 1 = Receives left channel data when monaural format is selected. 0 = Receives right channel data when monaural format is selected.
[22]	**Reserved**	Reserved
[21]	**RXDMA**	**Enable Receive DMA** When RX DMA is enabled, I^{2}S requests DMA to transfer data from receive FIFO to SRAM if FIFO is not empty. 1 = RX DMA Enabled. 0 = RX DMA Disabled.
[20]	**TXDMA**	**Enable Transmit DMA** When TX DMA is enables, I^{2}S request DMA to transfer data from SRAM to transmit FIFO if FIFO is not full. 1 = TX DMA Enabled. 0 = TX DMA Disabled.

[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, reading it returns 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 cleared by hardware automatically, reading it returns 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 = Left channel zero-cross detect Enabled. 0 = Left channel zero-cross detect Disabled.
[16]	**RCHZCEN**	**Right 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 = Right channel zero-cross detect Enabled. 0 = Right channel zero-cross detect Disabled.
[15]	**MCLKEN**	**Master clock enable** For NuMicro™ NUC123 series, if the external crystal clock is frequency 2*N*256fs, software can program MCLK_DIV[2:0] in I2S_CLKDIV register to get 256fs clock to audio codec chip. 1 = Master clock Enabled. 0 = Master clock Disabled.
[14:12]	**RXTH[2:0]**	**Receive FIFO threshold level** When received data word(s) in buffer is equal to or higher than threshold level, the 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, the 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 be operated as Master or Slave mode. For Master mode, I2S_BCLK and I2S_LRCLK pins are output mode and send bit clock from NuMicro™ NUC123 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 Selection** If PCM=0, 1 = MSB justified data format 0 = I^{2}S data format If PCM=1, 1 = PCM mode B. 0 = PCM mode A.
[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 = Data receiving Enabled. 0 = Data receiving Disabled.

[1]	**TXEN**	**Transmit enable** 1 = Data transmission Enabled. 0 = Data transmission Disabled.
[0]	**I2SEN**	**I^{2}S Controller Enable** 1 = Enabled. 0 = Disabled.

I²S Clock Divider (I2S_CLKDIV)

Register	Offset	R/W	Description	Reset Value
I2S_CLKDIV	I2S_BA+0x04	R/W	I²S 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	Description	
[31:16]	**Reserved**	Reserved
[15:8]	**BCLK_DIV [7:0]**	**Bit Clock Divider** If I²S operates in Master mode, bit clock is provided by NuMicro™ NUC123 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)

I²S Interrupt Enable Register (I2S_IE)

Register	Offset	R/W	Description	Reset Value
I2S_IE	I2S_BA+0x08	R/W	I²S 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	Description	
[31:13]	**Reserved**	Reserved
[12]	**LZCIE**	**Left Channel Zero-cross Interrupt Enable** Interrupt occurs if this bit is set to 1 and left channel zero-cross. 1 = Interrupt Enabled. 0 = Interrupt Disabled.
[11]	**RZCIE**	**Right Channel Zero-cross Interrupt Enable** 1 = Interrupt Enabled. 0 = Interrupt Disabled.
[10]	**TXTHIE**	**Transmitted FIFO Threshold Level Interrupt Enable** Interrupt occurs if this bit is set to 1 and data words in transmit FIFO are less than TXTH[2:0]. 1 = Interrupt Enabled. 0 = Interrupt Disabled.
[9]	**TXOVFIE**	**Transmitted FIFO Overflow Interrupt Enable** Interrupt occurs if this bit is set to 1 and transmitted FIFO overflow flag is set to 1. 1 = Interrupt Enabled. 0 = Interrupt Disabled.
[8]	**TXUDFIE**	**Transmitted FIFO Underflow Interrupt Enable** Interrupt occurs if this bit is set to 1 and transmitted FIFO underflow flag is set to 1. 1 = Interrupt Enabled. 0 = Interrupt Disabled.

[7:3]	**Reserved**	Reserved
[2]	**RXTHIE**	**Received FIFO Threshold Level Interrupt Enable** When data word in receive FIFO is equal to or higher then RXTH[2:0] and the RXTHF bit is set to 1. If RXTHIE bit is enabled, interrupt occurs. 1 = Interrupt Enabled. 0 = Interrupt Disabled.
[1]	**RXOVFIE**	**Receive FIFO Overflow Interrupt Enable** 1 = Interrupt Enabled. 0 = Interrupt Disabled.
[0]	**RXUDFIE**	**Receive FIFO Underflow Interrupt Enable** If software reads the received FIFO when it is empty, RXUDF flag in I2SSTATUS register is set to 1. 1 = Interrupt Enabled. 0 = Interrupt Disabled.

I²S Status Register (I2S_STATUS)

Register	Offset	R/W	Description	Reset Value
I2S_STATUS	I2S_BA+0x0C	R/W	I²S 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	Description	
[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 the 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 cleared to 0 when all data in transmit FIFO and shift buffer is shifted out, and set to 1 when the 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 to 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 occurred. 0 = No overflow occurred. 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 occurred. 0 = No underflow occurred. Software can write 1 to clear this bit to zero.
[7:4]	**Reserved**	Reserved
[3]	**RIGHT**	**Right Channel** This bit indicates the 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	Description	
[31:0]	**TXFIFO**	**Transmit FIFO register** I^{2}S contains 8 words (8x32 bit) data buffer for data transmission. Write data to this register to prepare data for transmit. The remaining 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	Description	
[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.

5.17 Analog-to-Digital Converter (ADC)

5.17.1 Overview

NuMicro™ NUC123 Series contains one 10-bt 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.

5.17.2 Features

- Analog input voltage range: 0~Vref (Max to 5.0V)
- 10-bit resolution and 8-bit accuracy is guaranteed
- Up to 8 single-end analog input channels
- Maximum ADC clock frequency to 6 MHz
- Up to 150K 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
- A/D conversion started by:
  - Software write 1 to ADST bit
  - External pin STADC
  - PWM output trigger
- Conversion results 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 2 input sources: external analog voltage and internal fixed band-gap voltage

5.17.3 Block Diagram

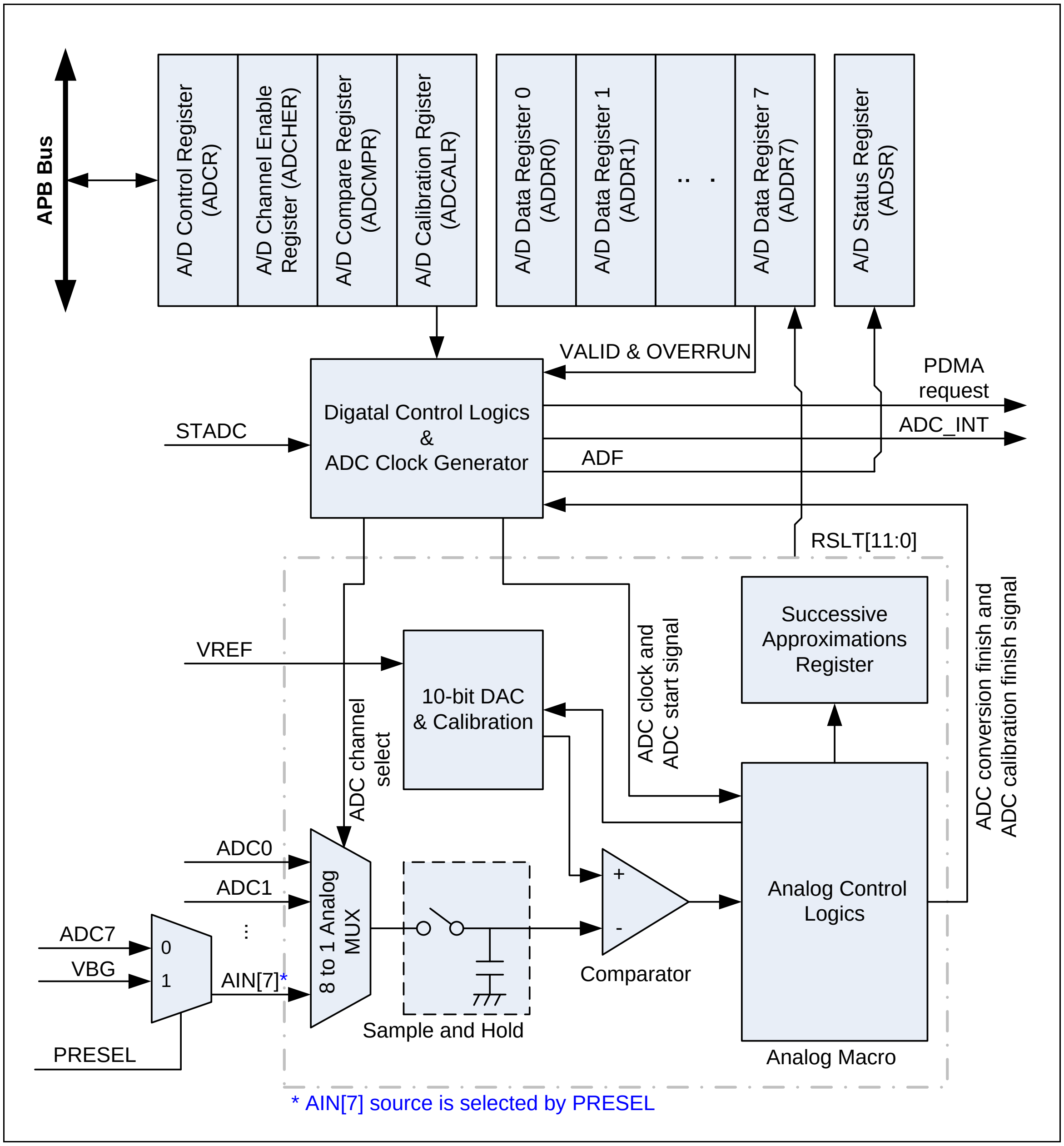

APB Bus
A/D Control Register (ADCR)
A/D Channel Enable Register (ADCHER)
A/D Compare Register (ADCMPR)
A/D Calibration Rgister (ADCALR)
A/D Data Register 0 (ADDR0)
A/D Data Register 1 (ADDR1)
. . .
A/D Data Register 7 (ADDR7)
A/D Status Register (ADSR)
VALID & OVERRUN
PDMA request
ADC_INT
ADF
STADC
Digatal Control Logics & ADC Clock Generator
RSLT[11:0]
Successive Approximations Register
VREF
10-bit DAC & Calibration
ADC channel select
ADC clock and ADC start signal
ADC conversion finish and ADC calibration finish signal
ADC0
ADC1
⋮
ADC7
VBG
0
1
AIN[7]*
8 to 1 Analog MUX
PRESEL
Sample and Hold
+
-
Comparator
Analog Control Logics
Analog Macro
* AIN[7] source is selected by PRESEL

Figure 5-91 ADC Controller Block Diagram

5.17.4 Functional Description

The A/D converter is operated by successive approximation with 10-bit resolution. 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.

5.17.4.1 *ADC Clock Generator*

The maximum sampling rate is up to 142SPS. 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.

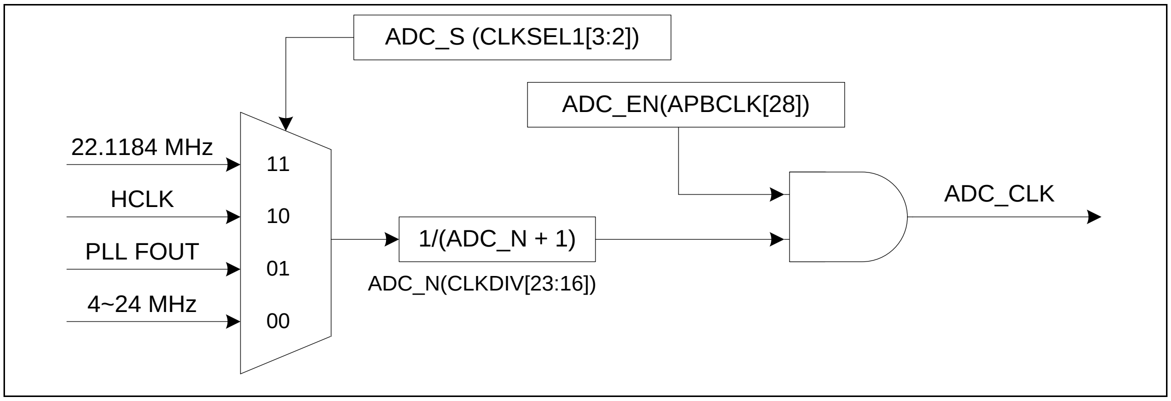


Figure 5-92 ADC Clock Control

5.17.4.2 *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.

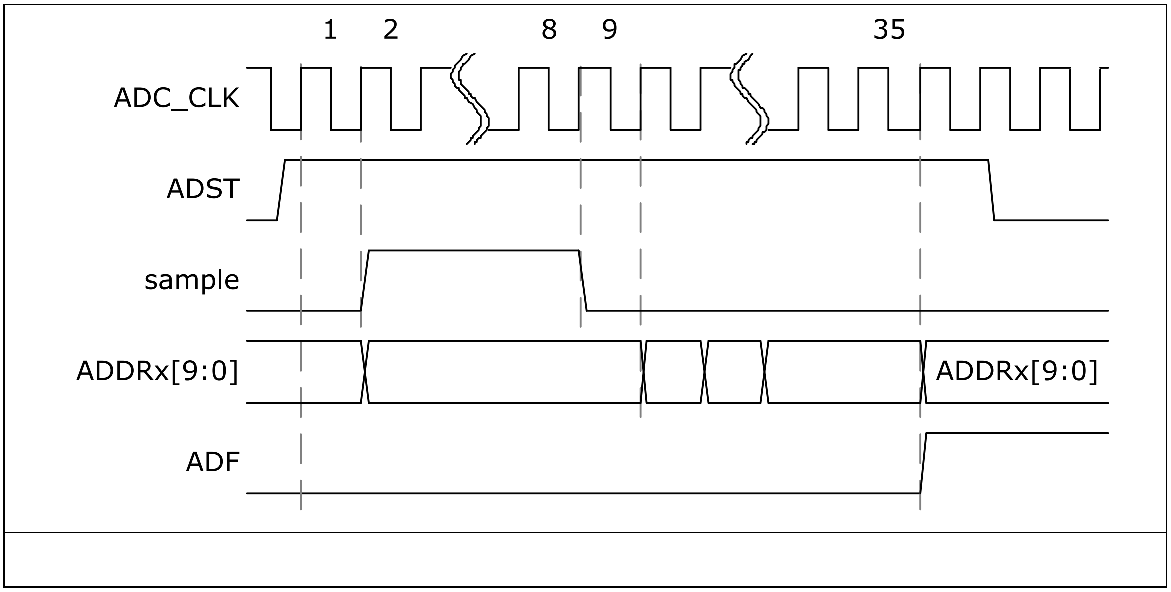


Figure 5-93 Single Mode Conversion Timing Diagram

5.17.4.3 *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 save the result to the ADDRx of the current conversion channel. 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 below:

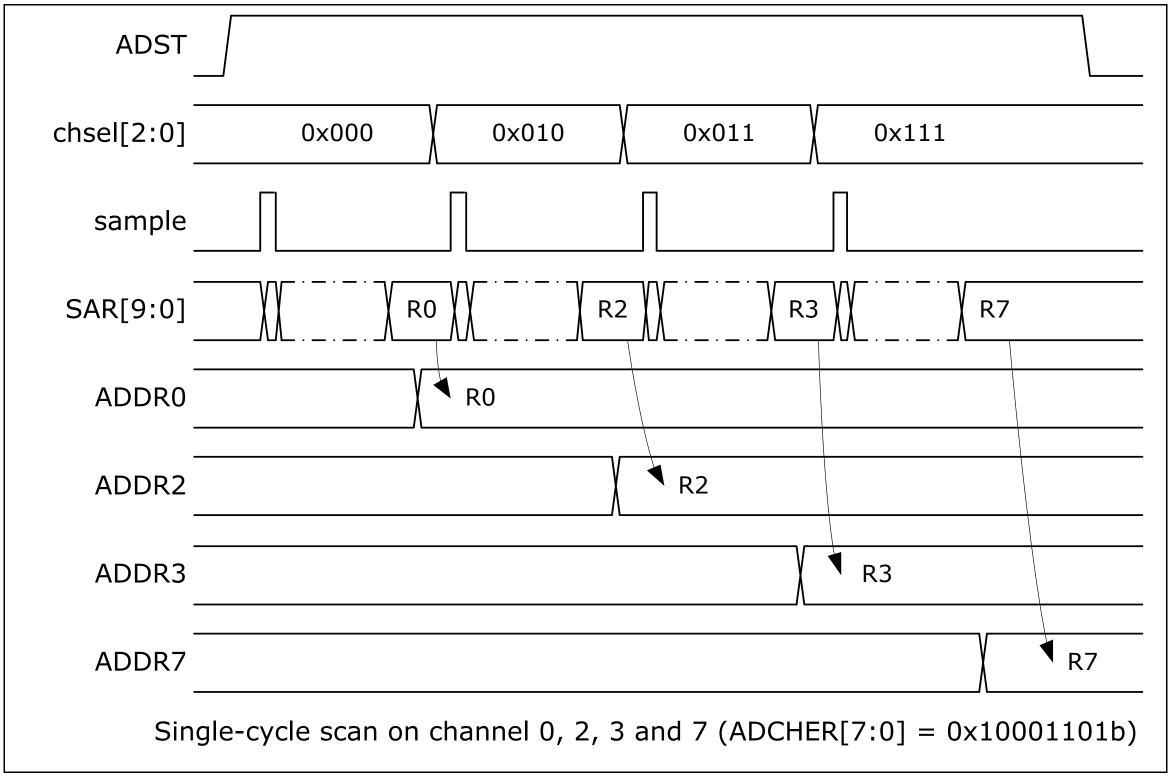


Figure 5-94 Single-Cycle Scan on Enabled Channels Timing Diagram

5.17.4.4 *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 stop conversion.

An example timing diagram for continuous scan on enabled channels (0, 2, 3 and 7) is shown below:

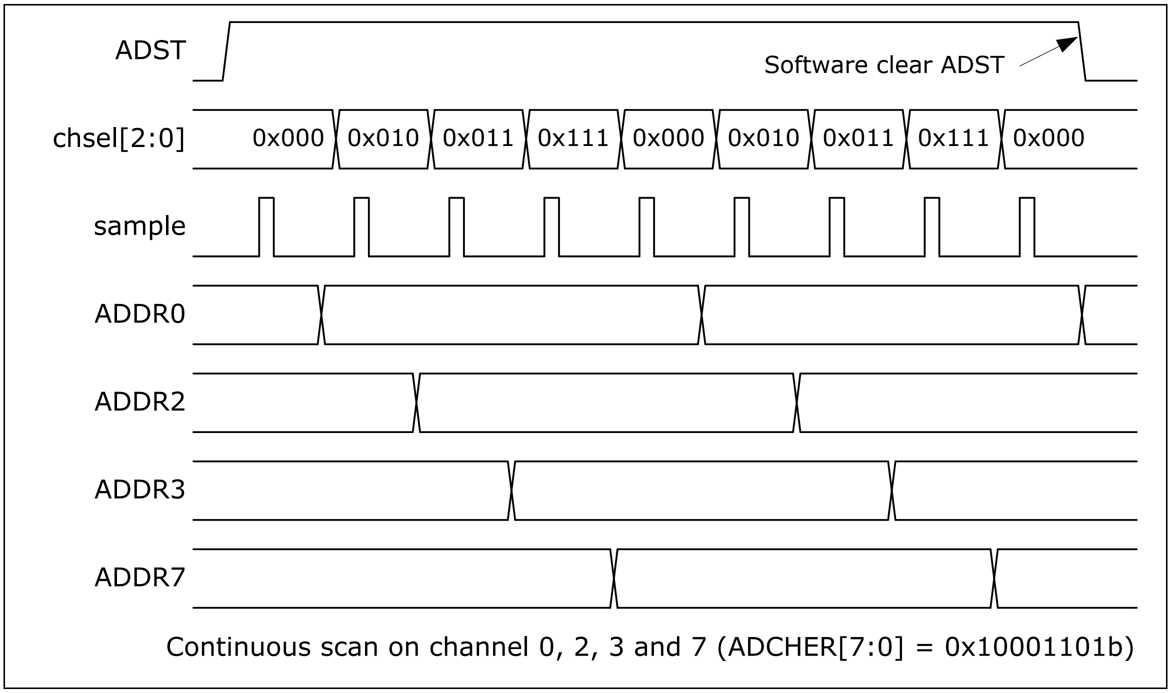


Figure 5-95 Continuous Scan on Enabled Channels Timing Diagram

5.17.4.5 *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.

5.17.4.6 *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 5-96. 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 below:

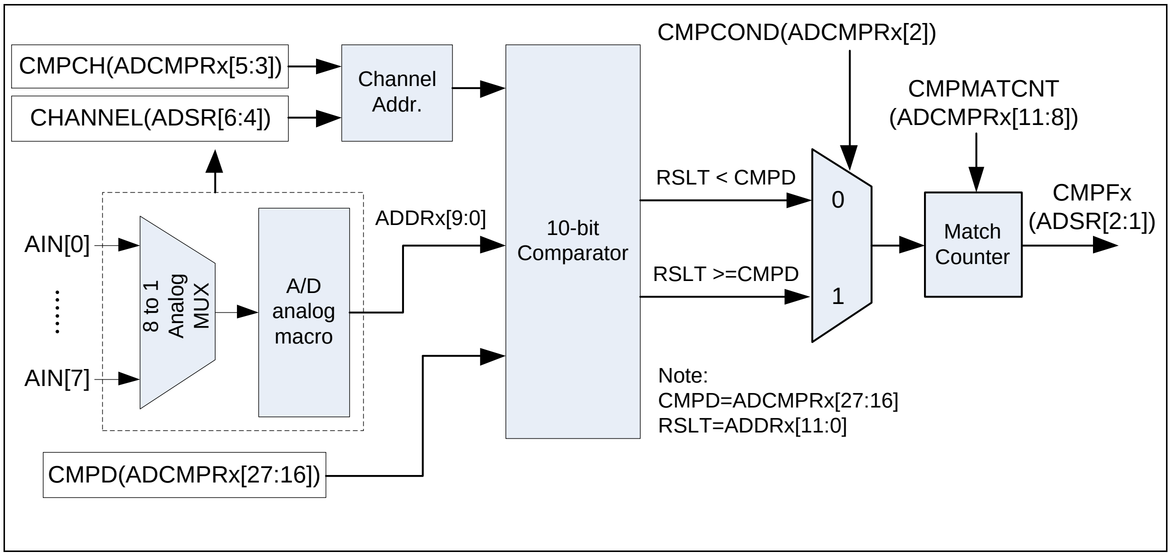


Figure 5-96 A/D Conversion Result Monitor Logics Diagram

5.17.4.7 *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.

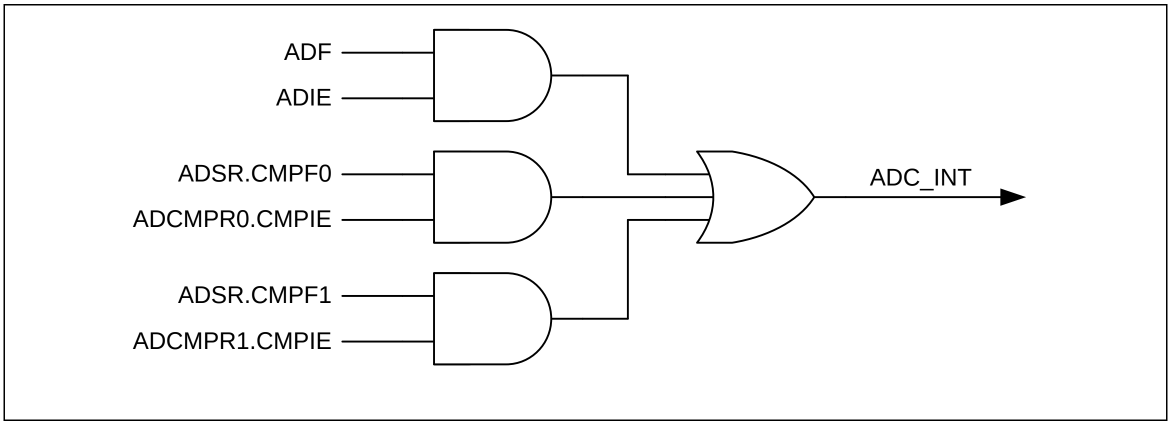


Figure 5-97 A/D Controller Interrupt

5.17.4.8 *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.

5.17.5 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
ADPDMA	ADC_BA+0x40	R	ADC PDMA current transfer data	0x0000_0000

5.17.6 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	Description	
[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 the 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 overwritten. 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 10 bits conversion result of ADC.

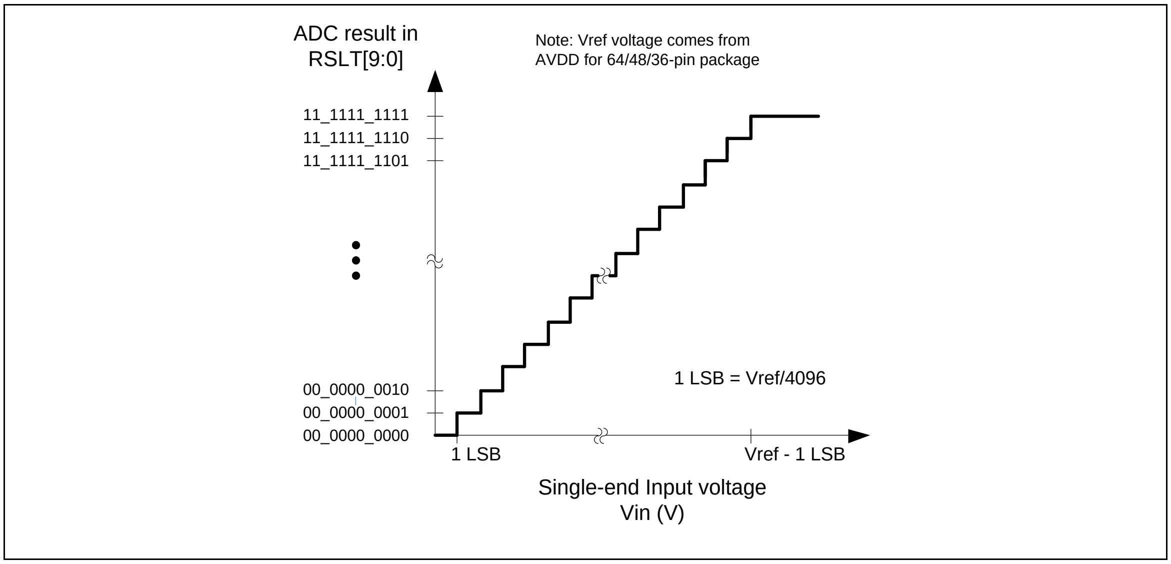


Figure 5-98 ADC Single-end Input Conversion Voltage and Conversion Result Mapping Diagram

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
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**ADST**	**Reserved**	**PTEN**	**TRGEN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TRGCOND		**TRGS**		**ADMD**		**ADIE**	**ADEN**

Bits	Description	
[30:12]	**Reserved**	Reserved
[11]	**ADST**	**A/D Conversion Start** 1 = Conversion started. 0 = Conversion stopped and A/D converter entering Idle state. ADST bit can be set to 1 from three sources: software and external pin STADC, and pwm output. 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.
[9]	**PTEN**	**PDMA Transfer Enable** 1 = PDMA data transfer in ADDR 0~7 Enabled. 0 = PDMA data transfer Disabled. 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.
[8]	**TRGEN**	**External Trigger Enable** Enable or disable triggering of A/D conversion by external STADC pin. 1= Enabled. 0= Disabled. ADC external trigger function is only supported in Single-cycle Scan mode.

[7:6]	**TRGCOND**	**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
[5:4]	**TRGS**	**Hardware Trigger Source** 00 = A/D conversion is started by external STADC pin. 00 = A/D conversion is started by PWM center-aligned trigger. 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.
[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 = A/D interrupt function Enabled. 0 = A/D interrupt function Disabled. A/D conversion end interrupt request is generated if ADIE bit is set to 1.
[0]	**ADEN**	**A/D Converter Enable** 1 = Enabled. 0 = Disabled. 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
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CHEN7	CHEN6	CHEN5	CHEN4	CHEN3	CHEN2	CHEN1	CHEN0

Bits	Description	
[31:9]	**Reserved**	Reserved
[8]	**PRESEL**	**Analog Input Channel 7 Selection** 0 = External analog input. 1 = Internal band-gap voltage. **Note:** When software selects the band-gap 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 = Enabled. 0 = Disabled.
[6]	**CHEN6**	**Analog Input Channel 6 Enable** 1 = Enabled. 0 = Disabled.
[5]	**CHEN5**	**Analog Input Channel 5 Enable** 1 = Enabled. 0 = Disabled.
[4]	**CHEN4**	**Analog Input Channel 4 Enable** 1 = Enabled. 0 = Disabled.
[3]	**CHEN3**	**Analog Input Channel 3 Enable** 1 = Enabled. 0 = Disabled.

[2]	**CHEN2**	**Analog Input Channel 2 Enable** 1 = Enabled. 0 = Disabled.
[1]	**CHEN1**	**Analog Input Channel 1 Enable** 1 = Enabled. 0 = Disabled.
[0]	**CHEN0**	**Analog Input Channel 0 Enable** 1 = Enabled. 0 = Disabled.

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	Description	
[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 Conditions** 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 = Compare function interrupt Enabled. 0 = Compare function interrupt Disabled. 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 = Compare function Enabled. 0 = Compare function Disabled. 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	Description	
[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. It is cleared by writing 1 to itself. 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 itself. 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 itself.

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

<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"><b>Reserved</b></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"><b>Reserved</b></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="6"><b>Reserved</b></td><td colspan="2"><b>AD_PDMA[9:8]</b></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"><b>AD_PDMA[7:0]</b></td></tr>
</table>

Bits	Description	
[31:10]	**Reserved**	Reserved
[9:0]	**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.

5.18 PDMA Controller (PDMA)

5.18.1 Overview

The NuMicro™ NUC123 DMA contains six-channel peripheral direct memory access (PDMA) controller and a cyclic redundancy check (CRC) generator.

The PDMA can transfers data to and from memory or transfer data to and from APB devices. For PDMA channel (PDMA CH0~CH5), 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.

The DMA controller contains a cyclic redundancy check (CRC) generator that can perform CRC calculation with programmable polynomial settings. The CRC engine supports CPU PIO mode and DMA transfer mode.

5.18.2 Features

- Supports six PDMA channels and one CRC channel. Each PDMA channel can support a unidirectional transfer
- AMBA AHB master/slave interface compatible, for data transfer and register read/write
- Hardware round robin priority scheme. DMA channel 0 has the highest priority
- PDMA
  - Peripheral-to-memory, memory-to-peripheral, and memory-to-memory transfer
  - Supports word/half-word/byte transfer data width from/to peripheral
  - Supports address direction: increment, fixed
- Cyclic Redundancy Check (CRC)
  - Supports four common polynomials CRC-CCITT, CRC-8, CRC-16, and CRC-32

    -- CRC-CCITT: $X^{16} + X^{12} + X^{5} + 1$

    -- CRC-8: $X^{8} + X^{2} + X + 1$

    -- CRC-16: $X^{16} + X^{15} + X^{2} + 1$

    -- CRC-32: $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^{8} + X^{7} + X^{5} + X^{4} + X^{2} + X + 1$
  - Programmable seed value
  - Supports programmable order reverse setting for input data and CRC checksum
  - Supports programmable 1’s complement setting for input data and CRC checksum.
  - Supports CPU PIO mode or DMA transfer mode
  - Supports 8/16/32-bit of data width in CPU PIO mode

    -- 8-bit write mode: 1-AHB clock cycle operation

    -- 16-bit write mode: 2-AHB clock cycle operation

    -- 32-bit write mode: 4-AHB clock cycle operation
  - Supports byte alignment transfer length in CRC DMA mode

5.18.3 Block Diagram

The DMA block diagram, DMA clock control diagram and CRC block diagram are shown as follows.

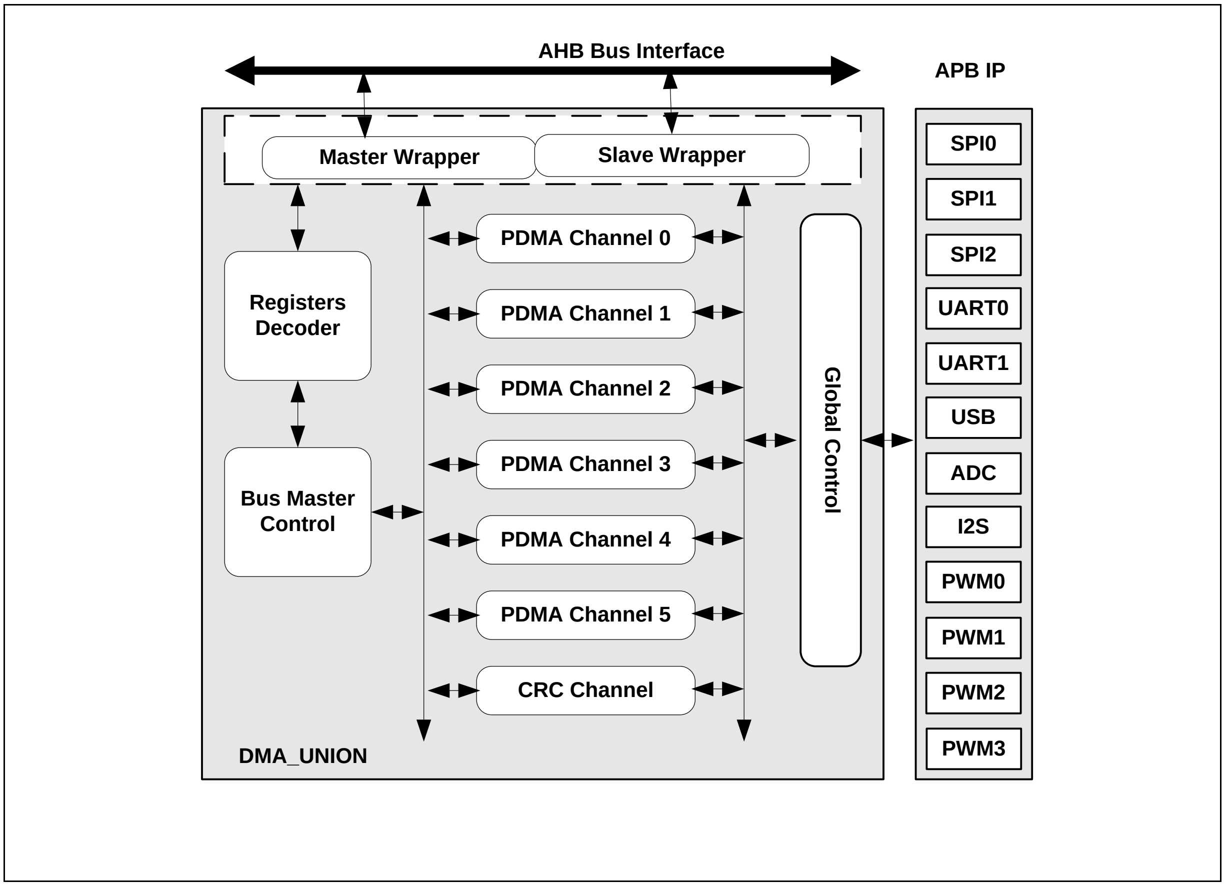


Figure 5-99 DMA Controller Block Diagram

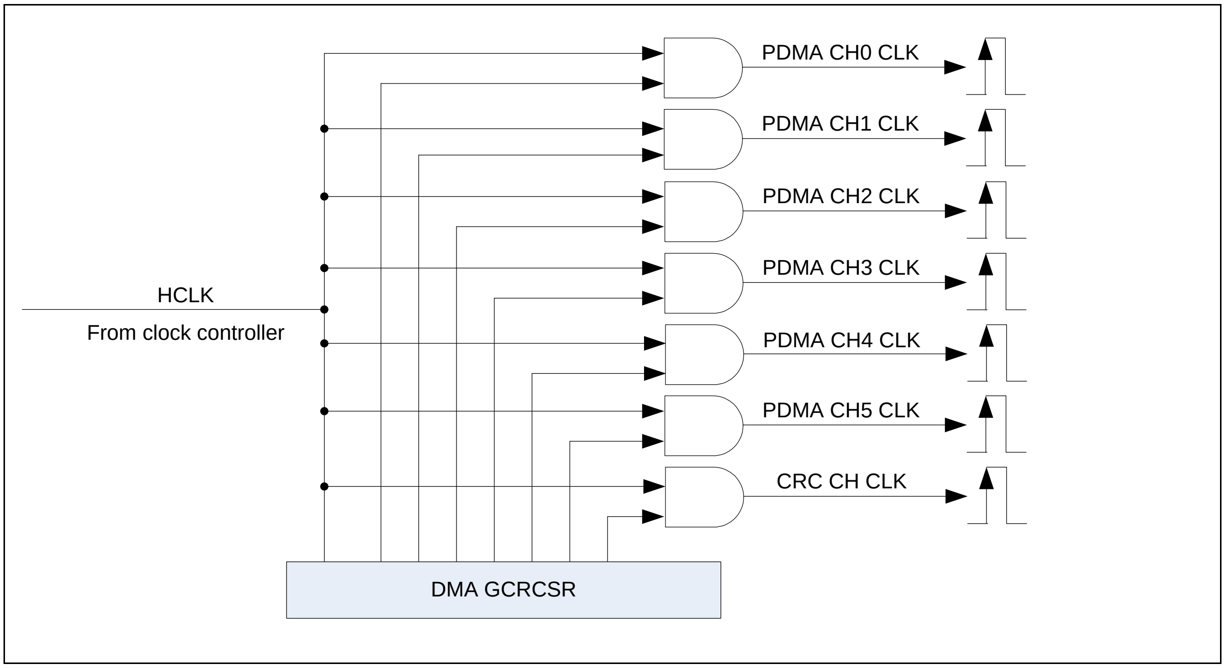


Figure 5-100 DMA Clock Controller Block Diagram

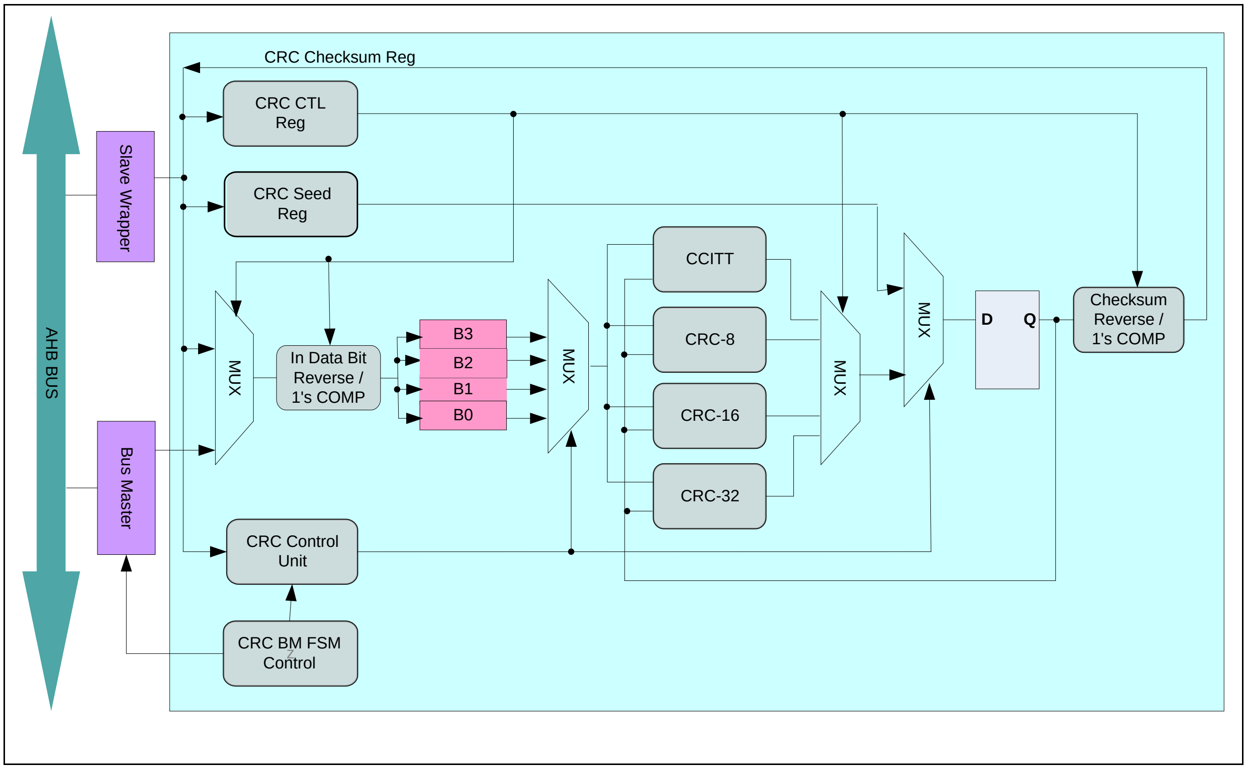


Figure 5-101 CRC Generator Block Diagram

5.18.4 Functional Description

The direct memory access (DMA) controller module transfers data from one address to another address, without CPU intervention. The DMA controller contains six PDMA (Peripheral-to-Memory or Memory-to-Peripheral or Memory-to-Memory) channels and one CRC generator channel.

The CPU can recognize the completion of a DMA operation by software polling or when it receives an internal DMA interrupt.

✧ PDMA

The DMA controller has six channels PDMA (Peripheral-to-Memory or Memory-to-Peripheral or Memory-to-Memory). As to the source and destination address, the PDMA controller has two modes: increased and fixed.

Every PDMA channel behavior is not pre-defined, so user must configure the channel service settings of PDMA_PDSSR0, PDMA_PDSSR1 and PDMA_PDSSR2 before start the related PDMA channel.

Software must enable PDMA channel by setting PDMACEN bit 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 PDMA_CSRx [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.

✧ CRC

The DMA controller contains a cyclic redundancy check (CRC) generator that can perform CRC calculation with programmable polynomial settings. The operation polynomial includes CRC-CCITT, CRC-8, CRC-16 and CRC-32; Software can choose the operation polynomial mode by setting CRC_MODE fields in CRC _CTL register.

The CRC engine support CPU PIO mode (CRC_CTL [CRCCEN] = 1, CRC_CTL [TRIG_EN] = 0) and DMA transfer mode (CRC_CTL [CRCCEN] = 1, CRC_CTL [TRIG_EN] = 1). The following sequence is a program sequence example.

Procedure when operation in CPU PIO mode:

- ➢ Enable CRC engine by setting CRCCEN bit in CRC_CTL register.
- ➢ Initial Setting. Setting the data format (WDATA_RVS, CHECKSUM_RVS, WDATA_COM and CHECKSUM_COM by setting CRC_CTL register), initial seed value (CRC_SEED) and select the data length by setting CRC_CTL [CPU_WDLEN] register.
- ➢ Setting CRC reset to load the initial seed value to CRC circuit by setting CRC_RST bit in CRC_CTL register.
- ➢ Write data to CRC_WDATA to perform CRC calculation.
- ➢ Get the CRC checksum result by reading CRC_CHECKSUM register.

Procedure when operation in CRC DMA mode:

- ➢ Enable CRC engine by setting CRCCEN bit in CRC_CTL register.

- Initial Setting. Setting the data format (WDATA_RVS, CHECKSUM_RVS, WDATA_COM and CHECKSUM_COM by setting CRC_CTL register), initial seed value (CRC_SEED).
- Give a valid source address and transfer count by setting CRC_DMASAR and CRC_DMABCR.
- Enable CRC_CTL [TRIG_EN] and then hardware will reset the seed value and then read memory data to perform CRC calculation.
- Wait CRC DMA transfer and CRC calculation done and than get the CRC checksum result by reading CRC_CHECKSUM register.

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_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_SBUF_cx	PDMA_BA_chx+0x80	R	PDMA Shared Buffer FIFO	0x0000_0000
CRC_BA = 0x5000_8E00				
CRC_CTL	CRC_BA+0x00	R/W	CRC Control Register	0x2000_0000
CRC_DMASAR	CRC_BA+0x04	R/W	CRC DMA Source Address Register	0x0000_0000
CRC_DMABCR	CRC_BA+0x0C	R/W	CRC Transfer Byte Count Register	0x0000_0000
CRC_DMACSAR	CRC_BA+0x14	R/W	CRC Current Source Address Register	0x0000_0000
CRC_DMACBCR	CRC_BA+0x1C	R/W	CRC Current Transfer Byte Count Register	0x0000_0000
CRC_DMAIER	CRC_BA+0x20	R/W	CRC Interrupt Enable Register	0x0000_0001
CRC_DMAISR	CRC_BA+0x24	R/W	CRC Interrupt Status Register	0x0000_0000
CRC_WDATA	CRC_BA+0x80	R/W	CRC Write Data Register	0x0000_0000
CRC_SEED	CRC_BA+0x84	R/W	CRC Seed Register	0xFFFF_FFFF
CRC_CHECKSUM	CRC_BA+0X88	R	CRC Check Sum Register	0x0000_0000

DMA_GCR_BA = 0x5000_8F00				
DMA_GCRCSR	DMA_GCR_BA+0x00	R/W	DMA Global Control Register	0x0000_0000
DMA_PDSSR0	DMA_GCR_BA+0x04	R/W	DMA Service Selection Control Register 0	0x00FF_FFFF
DMA_PDSSR1	DMA_GCR_BA+0x08	R/W	DMA Service Selection Control Register 1	0x0FFF_FFFF
DMA_GCRISR	DMA_GCR_BA+0x0C	R/W	DMA Global Interrupt Register	0x0000_0000
DMA_PDSSR2	DMA_GCR_BA+0x10	R/W	DMA Service Selection Control Register 2	0x00FF_FFFF

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

<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>TRIG_EN</td><td colspan="2">Reserved</td><td colspan="2">APB_TWS</td><td colspan="3">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="2">DAD_SEL</td><td colspan="2">SAD_SEL</td><td colspan="2">MODE_SEL</td><td>SW_RST</td><td>PDMACEN</td></tr>
</table>

Bits	Description	
[31:24]	**Reserved**	Reserved
[23]	**TRIG_EN**	**TRIG_EN** 1 = PDMA data read or write transfer Enabled. 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 Selection** 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 Selection** 00 = Transfer destination address is increasing successively. 01 = Reserved. 10 = Transfer destination address is fixed (This feature can be used when data transferred from multiple sources to a single destination). 11 = Reserved.
[5:4]	**SAD_SEL**	**Transfer Source Address Direction Selection** 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 Selection** 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 = No effect. 1 = Reset the internal state machine, pointers and internal buffer. The contents of control register will not be cleared. This bit will be automatically cleared 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.

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

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	Description	
[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

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	Description	
[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

31	30	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	Description	
[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

31	30	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	Description	
[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

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	Description	
[31:0]	**PDMA_CSAR**	**PDMA Current Source Address Register (Read Only)** This field indicates the source address where the PDMA transfer just occurs.

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

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	Description	
[31:0]	**PDMA_CDAR**	**PDMA Current Destination Address Register (Read Only)** This field indicates the destination address where the PDMA transfer just occurs.

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

31	30	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	Description	
[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

31	30	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	Description	
[31:2]	**Reserved**	Reserved
[1]	**BLKD_IE**	**PDMA Transfer Done Interrupt Enable** 1 = Interrupt generator Enabled when PDMA transfer is done. 0 = Interrupt generator Disabled when PDMA transfer is done.
[0]	**TABORT_IE**	**PDMA Read/Write Target Abort Interrupt Enable** 1 = Target abort interrupt generation Enabled during PDMA transfer. 0 = Target abort interrupt generation Disabled 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	0x0000_0000
PDMA_ISR1	PDMA_BA_ch1+0x24	R/W	PDMA Interrupt Status Register CH1	0x0000_0000
PDMA_ISR2	PDMA_BA_ch2+0x24	R/W	PDMA Interrupt Status Register CH2	0x0000_0000
PDMA_ISR3	PDMA_BA_ch3+0x24	R/W	PDMA Interrupt Status Register CH3	0x0000_0000
PDMA_ISR4	PDMA_BA_ch4+0x24	R/W	PDMA Interrupt Status Register CH4	0x0000_0000
PDMA_ISR5	PDMA_BA_ch5+0x24	R/W	PDMA Interrupt Status Register CH5	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						**BLKD_IF**	**TABORT_IF**

Bits	Description	
[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. 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: PDMA_ISR [TABORT_IF] indicate if 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_SBUF_cx)

Register	Offset	R/W	Description	Reset Value
PDMA_SBUF_c0	PDMA_BA_ch0+0x080	R	PDMA Shared Buffer FIFO Register CH0	0x0000_0000
PDMA_SBUF_c1	PDMA_BA_ch1+0x180	R	PDMA Shared Buffer FIFO Register CH1	0x0000_0000
PDMA_SBUF_c2	PDMA_BA_ch2+0x280	R	PDMA Shared Buffer FIFO Register CH2	0x0000_0000
PDMA_SBUF_c3	PDMA_BA_ch3+0x380	R	PDMA Shared Buffer FIFO Register CH3	0x0000_0000
PDMA_SBUF_c4	PDMA_BA_ch4+0x480	R	PDMA Shared Buffer FIFO Register CH4	0x0000_0000
PDMA_SBUF_c5	PDMA_BA_ch5+0x580	R	PDMA Shared Buffer FIFO Register CH5	0x0000_0000

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

Bits	Description	
[31:0]	**PDMA_SBUF**	**PDMA Shared Buffer FIFO (Read Only)** Each channel has its own 1 word internal buffer.

CRC Control Register (CRC_CTL)

Register	Offset	R/W	Description	Reset Value
CRC_CTL	CRC_BA+0x00	R/W	CRC Control Register	0x2000_0000

31	30	29	28	27	26	25	24
CRC_MODE		**CPU_WDLEN**		**CHECKSUM_COM**	**WDATA_COM**	**CHECKSUM_RVS**	**WDATA_RVS**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
TRIG_EN	**Reserved**						
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**CRC_RST**	**CRCCEN**

Bits	Description	
[31:30]	**CRC_MODE**	**CRC Polynomial Mode** 00 = CRC-CCITT Polynomial mode 01 = CRC-8 Polynomial mode 10 = CRC-16 Polynomial mode 11 = CRC-32 Polynomial mode
[29:28]	**CPU_WDLEN**	**CPU Write Data Length** When operation in CPU PIO mode (CRCCEN= 1, TRIG_EN = 0), this field indicates the write data length. 00 = Data length is 8-bit mode 01 = Data length is 16-bit mode 1x = Data length is 32-bit mode **Note1:** This field is used for CPU PIO mode. **Note2:** When the data length is 8-bit mode, the valid data is CRC_WDATA [7:0]; if the data length is 16-bit mode, the valid data is CRC_WDATA [15:0].
[27]	**CHECKSUM_COM**	**Checksum Complement** 1 = 1’s complement for CRC checksum. 0 = No 1’s complement for CRC checksum.
[26]	**WDATA_COM**	**Write Data Complement** 1 = 1’s complement for CRC write data in. 0 = No 1’s complement for CRC write data in.

[25]	**CHECKSUM_RVS**	**Checksum Reverse** 1 = Bit order reverse for CRC checksum. 0 = No bit order reverse for CRC checksum. **Note:** If the checksum data is 0XDD7B0F2E, the bit order reversed for CRC checksum is 0x74F0DEBB.
[24]	**WDATA_RVS**	**Write Data Order Reverse** 1 = Bit order reversed for CRC write data in (per byte) 0 = No bit order reversed for CRC write data in. **Note:** If the write data is 0xAABBCCDD, the bit order reverse for CRC write data in is 0x55DD33BB
[23]	**TRIG_EN**	**TRIG_EN** 1 = CRC DMA data read or write transfer Enabled. 0 = No effect. **Note1:** If this bit assert indicates the CRC engine operation in CRC DMA mode, do not fill in any data in CRC_WDATA register. **Note2:** When CRC DMA transfer is completed, this bit will be cleared automatically. **Note3:** If the bus error occurs, all CRC DMA transfer will be stopped. Software must reset all DMA channel, and then trigger again.
[22:2]	**Reserved**	Reserved
[1]	**CRC_RST**	**CRC Engine Reset** 0 = No effect. 1 = Reset the internal CRC state machine and internal buffer. The contents of control register will not be cleared. This bit will automatically be cleared after few clock cycles. **Note:** When operated in CPU PIO mode, setting this bit will reload the initial seed value.
[0]	**CRCCEN**	**CRC Channel Enable** Setting this bit to 1 enables CRC's operation. When operation in CRC DMA mode (TRIG_EN = 1), if user clears this bit, the DMA operation will be continuous until all CRC DMA operation is done, and the TRIG_EN bit will asserted until all CRC DMA operation done. But in this case, the CRC_DMAISR [BLKD_IF] flag will inactive, user can read CRC result by reading CRC_CHECKSUM register when TRIG_EN = 0 When operation in CRC DMA mode (TRIG_EN = 1), if user wants to stop the transfer immediately, user can write 1 to CRC_RST bit to stop the transmission.

CRC DMA Transfer Source Address Register (CRC_DMASAR)

Register	Offset	R/W	Description	Reset Value
CRC_DMASAR	CRC_BA+0x04	R/W	CRC DMA Transfer Source Address Register	0x0000_0000

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

Bits	Description	
[31:0]	**CRC_DMASAR**	**CRC DMA Transfer Source Address Register** This field indicates a 32-bit source address of CRC DMA. **Note:** The source address must be word alignment.

CRC DMA Transfer Byte Count Register (CRC_DMABCR)

Register	Offset	R/W	Description	Reset Value
CRC_DMABCR	CRC_BA+0x0C	R/W	CRC DMA Transfer Byte Count 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**
CRC_DMABCR [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CRC_DMABCR [7:0]							

Bits	Description	
[31:16]	**Reserved**	Reserved
[15:0]	**CRC_DMABCR**	**CRC DMA Transfer Byte Count Register** This field indicates a 16-bit transfer byte count number of CRC DMA.

CRC DMA Current Source Address Register (CRC_DMACSAR)

Register	Offset	R/W	Description	Reset Value
CRC_DMACSAR	CRC_BA+0x14	R	CRC DMA Current Source Address Register	0x0000_0000

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

Bits	Description	
[31:0]	**CRC_DMACSAR**	**CRC DMA Current Source Address Register (Read Only)** This field indicates the source address where the CRC DMA transfer just occurs.

CRC DMA Current Byte Count Register (CRC_DMACBCR)

Register	Offset	R/W	Description	Reset Value
CRC_DMACBCR	CRC_BA+0x1C	R	CRC DMA Current Byte Count 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**
CRC_DMACBCR [15:8]							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CRC_DMACBCR [7:0]							

Bits	Description	
[31:16]	**Reserved**	Reserved
[15:0]	**CRC_DMACBCR**	**CRC DMA Current Byte Count Register (Read Only)** This field indicates the current remained byte count of CRC_DMA. **Note:** CRC_RST will clear this register value.

CRC DMA Interrupt Enable Control Register (CRC_DMAIER)

Register	Offset	R/W	Description	Reset Value
CRC_DMAIER	CRC_BA+0x20	R/W	CRC DMA Interrupt Enable Control Register	0x0000_0001

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

Bits	Description	
[31:2]	**Reserved**	Reserved
[1]	**BLKD_IE**	**CRC DMA Transfer Done Interrupt Enable** 1 = Interrupt generator Enabled when CRC DMA transfer is done. 0 = Interrupt generator Disabled when CRC DMA transfer is done.
[0]	**TABORT_IE**	**CRC DMA Read/Write Target Abort Interrupt Enable** 1 = Target abort interrupt generation Enabled during CRC DMA transfer. 0 = Target abort interrupt generation Disabled during CRC DMA transfer.

CRC DMA Interrupt Status Register (CRC_DMAISR)

Register	Offset	R/W	Description	Reset Value
CRC_DMAISR	CRC_BA+0x24	R/W	CRC DMA 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						**BLKD_IF**	**TABORT_IF**

Bits	Description	
[31:2]	**Reserved**	Reserved
[1]	**BLKD_IF**	**Block Transfer Done Interrupt Flag** This bit indicates that CRC DMA has finished all transfer. 1 = Done 0 = Not finished. Software can write 1 to clear this bit to zero.
[0]	**TABORT_IF**	**CRC DMA 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: CRC_DMAISR [TABORT_IF] indicate if bus master received ERROR response or not. If bus master received ERROR response, it means that target abort is happened. DMA will stop transfer and respond this event to software then go to IDLE state. When target abort occurred, software must reset DMA, and then transfer those data again.

CRC Write Data Register (CRC_WDATA)

Register	Offset	R/W	Description	Reset Value
CRC_WDATA	CRC_BA+0x80	R/W	CRC Write Data Register	0x0000_0000

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

Bits	Description	
[31:0]	**CRC_WDATA**	**CRC Write Data Register** When operated in CPU PIO (CRC_CTL [CRCCEN] = 1, CRC_CTL [TRIG_EN] = 0) mode, software can write data to this field to perform CRC operation; When operated in CRC DMA mode (CRC_CTL [CRCCEN] = 1, CRC_CTL [TRIG_EN] = 0), this field will be used for DMA internal buffer. **Note1:** When operated in CRC DMA mode, so don't filled any data in this field. **Note2:** The CRC_CTL [WDATA_COM] and CRC_CTL [WDATA_RVS] bit setting will affect this field; for example, if WDATA_RVS = 1, if the write data in CRC_WDATA register is 0xAABBCCDD, the read data from CRC_WDATA register will be 0x55DD33BB.

CRC Seed Register (CRC_SEED)

Register	Offset	R/W	Description	Reset Value
CRC_SEED	CRC_BA+0x84	R/W	CRC Seed Register	0xFFFF_FFFF

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

Bits	Description	
[31:0]	**CRC_SEED**	**CRC Seed Register** This field indicates the CRC seed value.

CRC Checksum Register (CRC_CHECKSUM)

Register	Offset	R/W	Description	Reset Value
CRC_CHECKSUM	CRC_BA+0x88	R	CRC Checksum Register	0x0000_0000

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

Bits	Description	
[31:0]	**CRC_CHECKSUM**	**CRC Checksum Register** This field indicates the CRC checksum.

DMA Global Control Register (DMA_GCRCSR)

Register	Offset	R/W	Description	Reset Value
DMA_GCRCSR	DMA_GCR_BA+0x00	R/W	DMA Global Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							**CRC_CLK_EN**
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved		**CLK5_EN**	**CLK4_EN**	**CLK3_EN**	**CLK2_EN**	**CLK1_EN**	**CLK0_EN**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							

Bits	Description	
[31:25]	**Reserved**	Reserved
[24]	**CRC_CLK_EN**	**CRC Controller Clock Enable Control** 0 = Disabled. 1 = Enabled.
[23:14]	**Reserved**	Reserved
[13]	**CLK5_EN**	**PDMA Controller Channel 5 Clock Enable Control** 0 = Disabled. 1 = Enabled.
[12]	**CLK4_EN**	**PDMA Controller Channel 4 Clock Enable Control** 0 = Disabled. 1 = Enabled.
[11	**CLK3_EN**	**PDMA Controller Channel 3 Clock Enable Control** 0 = Disabled. 1 = Enabled.
[10	**CLK2_EN**	**PDMA Controller Channel 2 Clock Enable Control** 0 = Disabled. 1 = Enabled.
[9]	**CLK1_EN**	**PDMA Controller Channel 1 Clock Enable Control** 0 = Disabled. 1 = Enabled.

[8]	**CLK0_EN**	**PDMA Controller Channel 0 Clock Enable Control** 0 = Disabled. 1 = Enabled.
[7:0]	**Reserved**	Reserved

PDMA Service Selection Control Register 0 (DMA_PDSSR0)

Register	Address	R/W	Description	Reset Value
DMA_PDSSR0	DMA_GCR_BA+0x04	R/W	DMA Service Selection Control Register 0	0x00FF_FFFF

31	30	29	28	27	26	25	24
Reserved							
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	Description	
[31:24]	**Reserved**	Reserved
[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 Others: Reserved **Note:** For example, SPI0_RXSEL = 4’b0100 means SPI0_RX is connected to PDMA_CH4.

DMA Service Selection Control Register 1 (DMA_PDSSR1)

Register	Address	R/W	Description	Reset Value
DMA_PDSSR1	DMA_GCR_BA+0x08	R/W	DMA Service Selection Control Register 1	0x0FFF_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	Description	
[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 Others : Reserved **Note:** For example, UART0_RXSEL = 4’b0100 means UART0_RX is connected to PDMA_CH4.

DMA Global Interrupt Status Register (DMA_GCRISR)

Register	Offset	R/W	Description	Reset Value
DMA_GCRISR	DMA_GCR_BA+0x0C	R	DMA 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							**CRC_INTR**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**INTR5**	**INTR4**	**INTR3**	**INTR2**	**INTR1**	**INTR0**

Bits	Description	
[31]	**INTR**	**Interrupt Pin Status** This bit is the Interrupt status of PDMA controller. **Note:** This bit is read only
[30:17]	**Reserved**	Reserved
[6]	**CRC_INTR**	**Interrupt Pin Status of CRC Controller** This bit is the Interrupt status of CRC controller. **Note:** This bit is read only.
[15:6]	**Reserved**	Reserved
[5]	**INTR5**	**Interrupt Pin Status of Channel 5** This bit is the Interrupt status of PDMA channel 5. **Note:** This bit is read only.
[4]	**INTR4**	**Interrupt Pin Status of Channel 4** This bit is the Interrupt status of PDMA channel 4. **Note:** This bit is read only.
[3]	**INTR3**	**Interrupt Pin Status of Channel 3** This bit is the Interrupt status of PDMA channel 3. **Note:** This bit is read only.
[2]	**INTR2**	**Interrupt Pin Status of Channel 2** This bit is the Interrupt status of PDMA channel 2. **Note:** This bit is read only.

[1]	**INTR1**	**Interrupt Pin Status of Channel 1** This bit is the Interrupt status of PDMA channel 1. **Note:** This bit is read only.
[0]	**INTR0**	**Interrupt Pin Status of Channel 0** This bit is the Interrupt status of PDMA channel 0. **Note:** This bit is read only.

DMA Service Selection Control Register 2 (DMA_PDSSR2)

Register	Offset	R/W	Description	Reset Value
DMA_PDSSR2	DMA_BA_GCR+0x10	R/W	DMA Service Selection Control Register 2	0x00FF_FFFF

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
PWM3_RXSEL				**PWM2_RXSEL**			
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
PWM1_RXSEL				**PWM0_RXSEL**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
I2S_TXSEL				**I2S_RXSEL**			

Bits	Description	
[31:24]	**Reserved**	Reserved
[23:20]	**PWM3_RXSEL**	**PDMA PWM3 RX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral PWM3 RX. Software can configure the RX channel setting by PWM3_RXSEL. The channel configuration is the same as I2S_RXSEL field. Please refer to the explanation of I2S_RXSEL.
[19:16]	**PWM2_RXSEL**	**PDMA PWM2 RX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral PWM2 RX. Software can configure the RX channel setting by PWM2_RXSEL. The channel configuration is the same as I2S_RXSEL field. Please refer to the explanation of I2S_RXSEL.
[15:12]	**PWM1_RXSEL**	**PDMA PWM1 RX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral PWM1 RX. Software can configure the RX channel setting by PWM1_RXSEL. The channel configuration is the same as I2S_RXSEL field. Please refer to the explanation of I2S_RXSEL.
[11:8]	**PWM0_RXSEL**	**PDMA PWM0 RX Selection** This filed defines which PDMA channel is connected to the on-chip peripheral PWM0 RX. Software can configure the RX channel setting by PWM0_RXSEL. The channel configuration is the same as I2S_RXSEL field. Please refer to the explanation of I2S_RXSEL.
[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 Others : Reserved **Note:** For example, I2S_RXSEL = 4’b0100, which means I2S_RX is connected to PDMA_CH4.

6 ARM® CORTEX™-M0 CORE

6.1 Overview

The Cortex™-M0 processor, a configurable, multistage, 32-bit RISC processor, has an AMBA AHB-Lite interface and includes an NVIC component. The processor has optional hardware debug functionality, can execute Thumb code, and is compatible with other Cortex-M profile processors. The profile supports two modes -Thread mode and Handler mode. Handler mode is entered as a result of an exception. An exception return can only be issued in Handler mode. Thread mode is entered on Reset, and can be entered as a result of an exception return. Figure 6-1 shows the functional controller of processor.

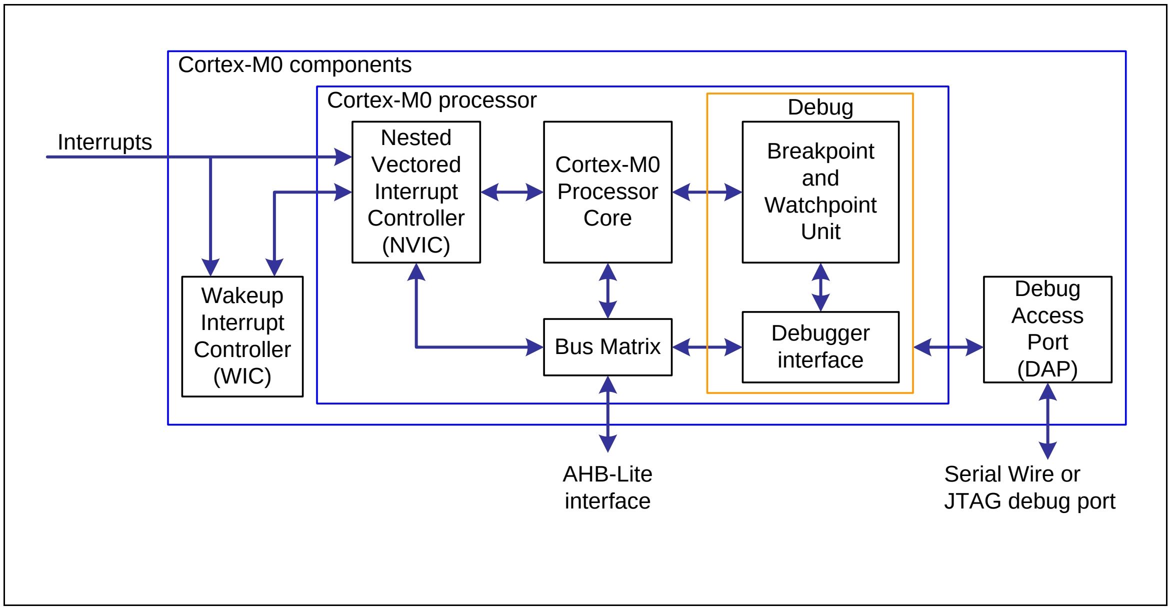


Figure 6-1 Functional Controller Diagram

6.2 Features

- A low gate count processor:
  - ARMv6-M Thumb® instruction set
  - Thumb-2 technology
  - ARMv6-M compliant 24-bit SysTick timer
  - A 32-bit hardware multiplier
  - System interface supporting little-endian data accesses
  - Ability to have deterministic, fixed-latency, interrupt handling
  - Load/store-multiples and multicycle-multiplies 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 :
  - 32 external interrupt inputs, each with four levels of priority
  - Dedicated Non-Maskable Interrupt (NMI) input
  - Supports both level-sensitive and pulse-sensitive interrupt lines
  - Supports Wake-up Interrupt Controller (WIC) with ultra-low power sleep mode
- 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 providing simple integration to all system peripherals and memory
  - Single 32-bit slave port supporting the DAP (Debug Access Port)

6.3 System Timer (SysTick)

The Cortex-M0 includes an integrated system timer, SysTick, which provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism. The counter can be used as a Real Time Operating System (RTOS) tick timer or as a simple counter.

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

The SYST_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 "ARM® Cortex™-M0 Technical Reference Manual" and "ARM® v6-M Architecture Reference Manual".

6.3.1 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

6.3.2 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	Description	
[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 = Counter will operate in a multi-shot manner. 0 = Counter 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	Description	
[31:24]	**Reserved**	Reserved
[23:0]	**RELOAD**	The 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	Description	
[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. Unsupported bits RAZ (see SysTick Reload Value register).

6.4 System Control Register

The 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 “ARM® Cortex™-M0 Technical Reference Manual” and “ARM® v6-M Architecture Reference Manual”.

6.4.1 System Control Register Memory Map

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

6.4.2 System Control Register

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	Description	
[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	Description	
[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, Rxcluding 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. Non-zero = 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 Non-zero = The exception number of the currently active exception.

Application Interrupt and Reset Control Register (AIRCR)

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

31	30	29	28	27	26	25	24
VECTORKEY[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	Description	
[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 and indicate a reset is requested. The bit is a write only and self-cleared 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	Description	
[31:5]	**Reserved**	Reserved
[4]	**SEVONPEND**	Send Event on Pending bit: 0 = Only enabled interrupts or events can wake up the processor, while disabled interrupts are excluded 1 = Enabled events and all interrupts, including disabled interrupts, can wake up the processor. When an event or interrupt enters pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE. The processor also wakes up on execution of an SEV instruction or an external event.
[3]	**Reserved**	Reserved
[2]	**SLEEPDEEP**	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 = Enters sleep, or deep sleep, on return from an ISR to Thread mode. Setting this bit to 1 enables an interrupt driven application to avoid returning to an empty main application.
[0]	**Reserved**	Reserved

System Handler Priority Register 2 (SHPR2)

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

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

Bits	Description	
[31:30]	**PRI_11**	**Priority of system handler 11 – SVCall** “0” denotes the highest priority and “3” denotes the lowest priority.
[29:0]	**Reserved**	Reserved

System Handler Priority Register 3 (SHPR3)

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

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

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

7 ELECTRICAL CHARACTERISTICS

7.1 Absolute Maximum Ratings

SYMBOL	PARAMETER	MIN	MAX	UNIT
DC Power Supply	V_{DD}–V_{SS}	-0.3	+7.0	V
Input Voltage	V_{IN}	V_{SS} -0.3	V_{DD} +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 an I/O pin			35	mA
Maximum Current sourced by an 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™ NUC123 DC Electrical Characteristics

(VDD-VSS=3.3 V, TA = 25°C, FOSC = 72 MHz unless otherwise specified.)

PARAMETER	SYM	SPECIFICATIONS				TEST CONDITIONS
		MIN	TYP	MAX	UNIT	
Operation voltage	V_{DD}	2.5		5.5	V	V_{DD} = 2.5V ~ 5.5V up to 72 MHz
VDD rise rate to ensure internal operation correctly	V_{RISE}	0.05			V/ms	
Power ground	V_{SS} AV_{SS}	-0.3			V	
LDO output voltage	V_{LDO}	-10%	1.8	+10%	V	V_{DD} > 2.7V
Analog operating voltage	AV_{DD}	0		V_{DD}	V	
Operating current Normal Run mode at 72 MHz	I_{DD1}		36		mA	V_{DD} = 5.5V at 72 MHz, All IP and PLL Enabled, XTAL = 12 MHz
	I_{DD2}		21		mA	V_{DD} = 5.5V at 72 MHz, All IP Disabled and PLL Enabled, XTAL = 12 MHz
	I_{DD3}		35		mA	V_{DD} = 3V at 72 MHz, All IP and PLL enabled, XTAL = 12 MHz
	I_{DD4}		20		mA	V_{DD} = 3V at 72 MHz, All IP Disabled and PLL Enabled, XTAL = 12 MHz
Operating current Normal Run mode at 12 MHz	I_{DD5}		7		mA	V_{DD} = 5.5V at 12 MHz, All IP Enabled and PLL Disabled, XTAL = 12 MHz
	I_{DD6}		4		mA	V_{DD} = 5.5V at 12 MHz, All IP and PLL Disabled, XTAL = 12 MHz
	I_{DD7}		6		mA	V_{DD} = 3V at 12 MHz, All IP Enabled and PLL Disabled, XTAL = 12 MHz

PARAMETER	SYM	SPECIFICATIONS				TEST CONDITIONS
		MIN	TYP	MAX	UNIT	
	I_{DD8}		3		mA	V_{DD} = 3V at 12 MHz, All IP and PLL Disabled, XTAL = 12 MHz
Operating current Normal Run mode at 4 MHz	I_{DD9}		4		mA	V_{DD} = 5V at 4 MHz, All IP Enabled and PLL Disabled, XTAL = 4 MHz
	I_{DD10}		3		mA	V_{DD} = 5V at 4 MHz, All IP and PLL Disabled, XTAL = 4 MHz
	I_{DD11}		4		mA	V_{DD} = 3V at 4 MHz, All IP Enabled and PLL Disabled, XTAL = 4 MHz
	I_{DD12}		2		mA	V_{DD} = 3V at 4 MHz, All IP and PLL Disabled, XTAL = 4 MHz
Operating current Idle mode at 72 MHz	I_{IDLE1}		29		mA	V_{DD} = 5.5V at 72 MHz, All IP and PLL Enabled, XTAL = 12 MHz
	I_{IDLE2}		14		mA	V_{DD} = 5.5V at 72 MHz, All IP Disabled and PLL Enabled, XTAL = 12 MHz
	I_{IDLE3}		28		mA	V_{DD} = 3V at 72 MHz, All IP and PLL Enabled, XTAL = 12 MHz
	I_{IDLE4}		13		mA	V_{DD} = 3V at 72 MHz, All IP Disabled and PLL Enabled, XTAL=12 MHz
Operating current Idle mode at 12 MHz	I_{IDLE5}		6		mA	V_{DD} = 5.5V at 12 MHz, All IP Enabled and PLL Disabled, XTAL = 12 MHz
	I_{IDLE6}		3		mA	V_{DD} = 5.5V at 12 MHz, All IP and PLL Disabled, XTAL = 12 MHz
	I_{IDLE7}		5		mA	V_{DD} = 3V at 12 MHz, All IP Enabled and PLL Disabled, XTAL = 12 MHz

PARAMETER	SYM	SPECIFICATIONS				TEST CONDITIONS
		MIN	TYP	MAX	UNIT	
	I_{IDLE8}		2		mA	V_{DD} = 3 V at 12 MHz, All IP and PLL Disabled, XTAL = 12 MHz
Operating current Idle mode at 4 MHz	I_{IDLE9}		3		mA	V_{DD} = 5V at 4 MHz, All IP Enabled and PLL Disabled, XTAL = 4 MHz
	I_{IDLE10}		2		mA	V_{DD} = 5V at 4 MHz, All IP and PLL Disabled, XTAL = 4 MHz
	I_{IDLE11}		2		mA	V_{DD} = 3V at 4 MHz, All IP Enabled and PLL Disabled, XTAL = 4 MHz
	I_{IDLE12}		1		mA	V_{DD} = 3V at 4 MHz, All IP and PLL Disabled, XTAL = 4 MHz
Operating current Idle mode at 10 kHz	I_{IDLE5}		131		uA	V_{DD} = 5.5V at 10 kHz, All IP Enabled and PLL Disabled, LIRC 10 kHz Enabled
	I_{IDLE6}		129		uA	V_{DD} = 5.5V at 10 kHz, All IP and PLL Disabled, LIRC 10 kHz Enabled
	I_{IDLE7}		125		uA	V_{DD} = 3V at 10 kHz, All IP Enabled and PLL Disabled, LIRC 10 kHz Enabled
	I_{IDLE8}		124		uA	V_{DD} = 3 V at 10 kHz, All IP and PLL Disabled, LIRC 10 kHz Enabled
Standby current Power-down mode	I_{PWD1}		12		μA	V_{DD} = 5.5V, No load when BOV function Disabled
	I_{PWD2}		9		μA	V_{DD} = 3.3V, No load when BOV function Disabled
Input Current PA, PB, PC, PD, PE, PF (Quasi-bidirectional mode)	I_{IN1}		-64		μA	V_{DD} = 5.5V, V_{IN} = 0V or $V_{IN} = V_{DD}$
Input Current at /RESET[1]	I_{IN2}	-55	-45	-30	μA	V_{DD} = 3.3V, V_{IN} = 0.45V
Input Leakage Current PA, PB, PC, PD, PE, PF	I_{LK}	-2	-	+2	μA	V_{DD} = 5.5V, $0 < V_{IN} < V_{DD}$

PARAMETER	SYM	SPECIFICATIONS				TEST CONDITIONS
		MIN	TYP	MAX	UNIT	
Logic 1 to 0 Transition Current PA~PF (Quasi-bidirectional mode)	I_{TL} [3]	-650	-	-200	μA	V_{DD} = 5.5V, V_{IN} < 2.0V
Input Low Voltage PA, PB, PC, PD, PE, PF (TTL input)	V_{IL1}	-0.3	-	0.8	V	V_{DD} = 4.5V
		-0.3	-	0.6		V_{DD} = 2.5V
Input High Voltage PA, PB, PC, PD, PE, PF (TTL input)	V_{IH1}	2.0	-	V_{DD} +0.2	V	V_{DD} = 5.5V
		1.5	-	V_{DD} +0.2		V_{DD} = 3.0V
Input Low Voltage PA, PB, PC, PD, PE, PF (Schmitt input)	V_{IL2}	-0.5	-	0.35 V_{DD}	V	
Input High Voltage PA, PB, PC, PD, PE, PF (Schmitt input)	V_{IH2}	0.65 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} = 4.5V
		0	-	0.4		V_{DD} = 3.0V
Input High Voltage XT1[*2]	V_{IH3}	3.5	-	V_{DD} +0.2	V	V_{DD} = 5.5V
		2.4	-	V_{DD} +0.2		V_{DD} = 3.0V
Negative going threshold (Schmitt input), /RESET	V_{ILS}	-0.5	-	0.2 V_{DD}	V	
Positive going threshold (Schmitt input), /RESET	V_{IHS}	0.6 V_{DD}	-	V_{DD}+0.5	V	
Source Current PA, PB, PC, PD, PE, PF (Quasi-bidirectional Mode)	I_{SR11}	-300	-370	-450	μA	V_{DD} = 4.5V, V_S = 2.4V
	I_{SR12}	-50	-70	-90	μA	V_{DD} = 2.7V, V_S = 2.2V
	I_{SR12}	-40	-60	-80	μA	V_{DD} = 2.5V, V_S = 2.0V
Source Current PA, PB, PC, PD, PE, PF (Push-pull Mode)	I_{SR21}	-20	-24	-28	mA	V_{DD} = 4.5V, V_S = 2.4V
	I_{SR22}	-4	-6	-8	mA	V_{DD} = 2.7V, V_S = 2.2V
	I_{SR22}	-3	-5	-7	mA	V_{DD} = 2.5V, V_S = 2.0V
Sink Current PA, PB, PC, PD, PE, PF (Quasi-bidirectional and Push-pull Mode)	I_{SK1}	10	16	20	mA	V_{DD} = 4.5V, V_S = 0.45V
	I_{SK1}	7	10	13	mA	V_{DD} = 2.7V, V_S = 0.45V
	I_{SK1}	6	9	12	mA	V_{DD} = 2.5V, V_S = 0.45V

PARAMETER	SYM	SPECIFICATIONS				TEST CONDITIONS
		MIN	TYP	MAX	UNIT	
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	$V_{BO3.8}$	3.7	3.8	3.9	V	
Brown-out voltage with BOV_VL [1:0] =11b	$V_{BO4.5}$	4.4	4.5	4.6	V	
Hysteresis range of BOD voltage	V_{BH}	30	-	150	mV	V_{DD} = 2.5V - 5.5V

Notes:

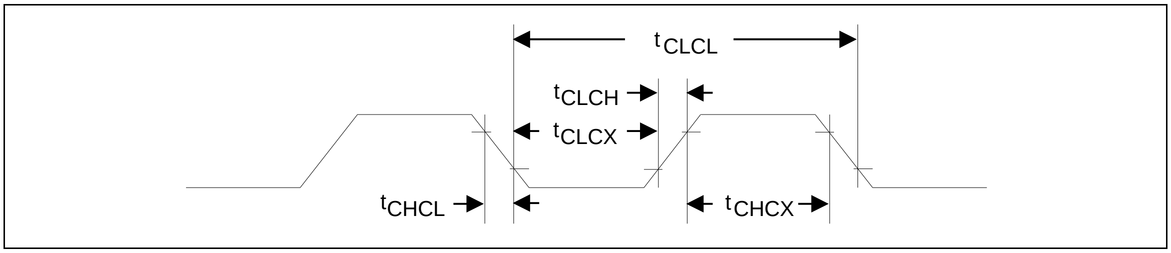
1. /RESET pin is a Schmitt trigger input.

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 2V.

7.3 AC Electrical Characteristics

7.3.1 External 4~24 MHz High Speed Oscillator



Note: Duty cycle is 50%.

SYMBOL	PARAMETER	CONDITIONS	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	CONDITIONS	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

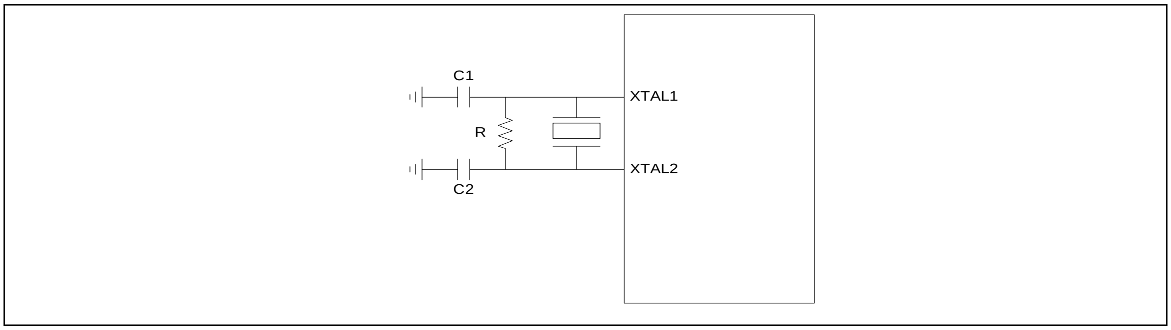


Figure 7-1 Typical Crystal Application Circuit

7.3.3 Internal 22.1184 MHz High Speed Oscillator

PARAMETER	CONDITIONS	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

7.3.4 Internal 10 kHz Low Speed Oscillator

PARAMETER	CONDITIONS	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 10-bit SARADC Specifications

PARAMETER	SYM	SPECIFICATIONS				TEST CONDITIONS
		MIN	TYP	MAX	UNIT	
Operating Voltage	AV_{DD}	2.7		5.5	V	AV_{DD} = V_{DD}
Operating Current	I_{ADC}			1.5	mA	AV_{DD} = V_{DD} = 5V, F_{SPS} = 150K
Resolution	R_{ADC}			10	bit	
Reference Voltage	V_{REF}		A_{VDD}		V	V_{REF} Connected to A_{VDD} in Chip
ADC input Voltage	V_{IN}	0		A_{VDD}	V	
Sampling Rate	F_{SPS}	150K			Hz	V_{DD} = 5V, ADC Clock = 6MHz Free Running Conversion
Integral Non-linearity Error (INL)	INL			±1	LSB	
Differential Non-linearity Error (DNL)	DNL			±1	LSB	
Gain Error	E_G			±2	LSB	
Offset Error	E_{OFFSET}		3		LSB	
Absolute Error	E_{ABS}		4		LSB	
ADC Clock Frequency	F_{ADC}	100K		6M	Hz	V_{DD} = 5V
Clock Cycle	AD_{CYC}	36			Cycle	
Bang-gap Voltage	V_{BG}	1.27	1.35	1.44	V	-40°C ~ +85°C

7.4.2 LDO and Power Management Specifications

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} > 2.7V
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	-	1	-	uF	Resr = 1Ω

Notes:

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. To ensure power stability, a 1uF (Cbp) or higher capacitor must be connected between LDO pin and the closest VSS pin of the device.

7.4.3 Low Voltage Reset Specifications

PARAMETER	CONDITIONS	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 Brown-out Detector Specifications

PARAMETER	CONDITIONS	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	4.4	4.5	4.6	V
	BOV_VL [1:0] = 10	3.7	3.8	3.9	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 Power-On Reset (5V) Specifications

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
Temperature	-	-40	25	85	℃
Reset voltage	V+	-	2	-	V
Quiescent current	Vin>reset voltage	-	1	-	nA

7.4.6 USB PHY Specifications

7.4.6.1 *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

Note: Driver output resistance doesn't include series resistor resistance.

7.4.6.2 *USB Full-Speed Driver Electrical Characteristics*

SYMBOL	PARAMETER	CONDITIONS	MIN	TYP	MAX	UNIT
T_{FR}	Rising time	C_L = 50p	4		20	ns
T_{FF}	Falling time	C_L = 50p	4		20	ns
T_{FRFF}	Rising and falling time matching	$T_{FRFF} = T_{FR}/T_{FF}$	90		111.11	%

7.4.6.3 *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	TBD	TBD	-	ns
t_{DH}	Data hold time	TBD	-	-	ns
t_V	Data output valid time	-	TBD	TBD	ns
SPI Master mode (VDD = 3.0V ~ 3.6V, 30pF loading Capacitor)					
t_{DS}	Data setup time	TBD	TBD	-	ns
t_{DH}	Data hold time	TBD	-	-	ns
t_V	Data output valid time	-	TBD	TBD	ns
SPI Slave mode (VDD = 4.5V ~ 5.5V, 30pF loading Capacitor)					
t_{DS}	Data setup time	TBD	-	-	ns
t_{DH}	Data hold time	TBD	-	-	ns
t_V	Data output valid time	-	TBD	TBD	ns
SPI Slave mode (VDD = 3.0V ~ 3.6V, 30pF loading Capacitor)					
t_{DS}	Data setup time	TBD	-	-	ns
t_{DH}	Data hold time	TBD	-	-	ns
t_V	Data output valid time	-	TBD	TBD	ns

TBD: To be defined.

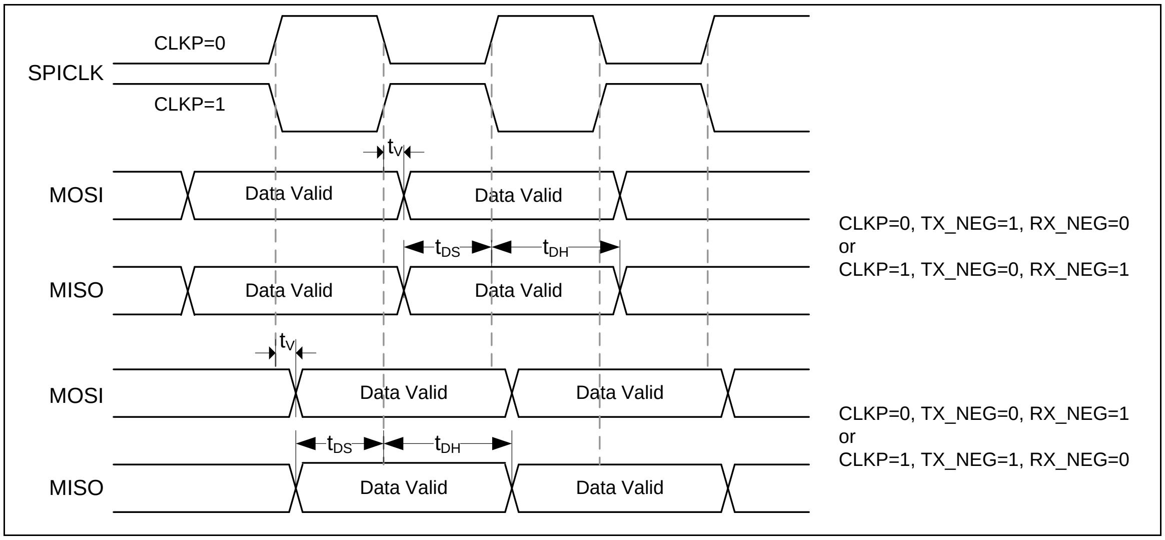

SPICLK
CLKP=0
CLKP=1
MOSI
MISO
MOSI
MISO
Data Valid
t_V
t_{DS}
t_{DH}
CLKP=0, TX_NEG=1, RX_NEG=0
or
CLKP=1, TX_NEG=0, RX_NEG=1
CLKP=0, TX_NEG=0, RX_NEG=1
or
CLKP=1, TX_NEG=1, RX_NEG=0

Figure 7-2 SPI Master Dynamic Characteristics Timing

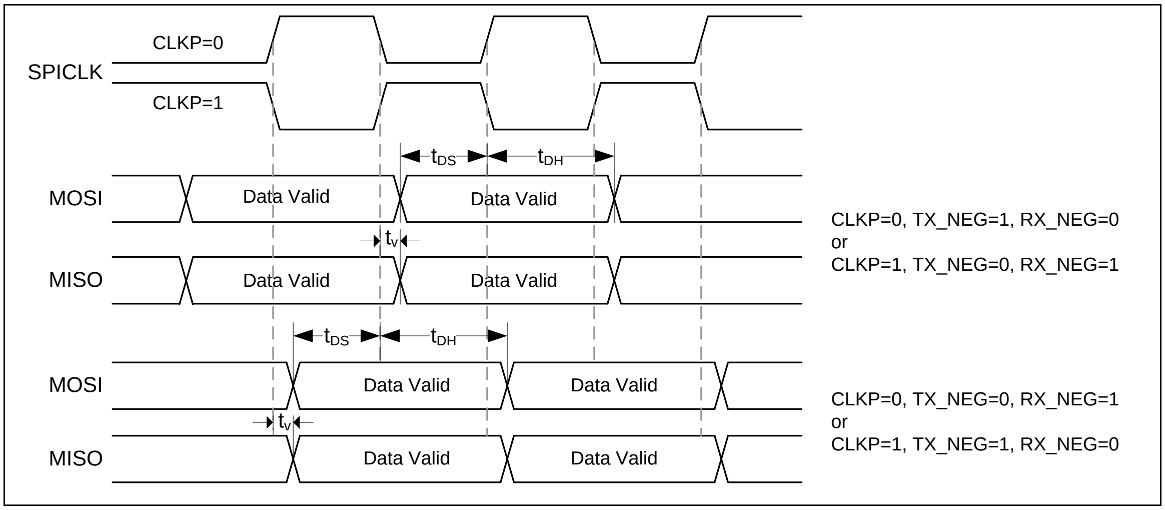
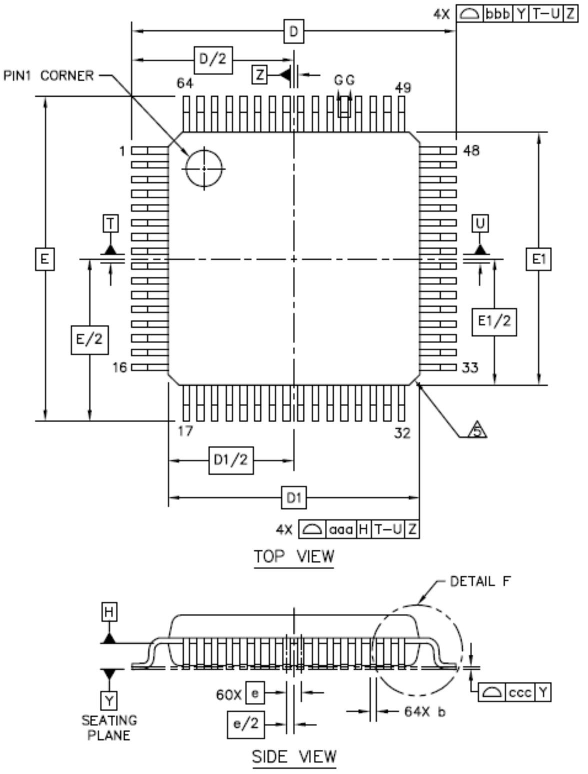


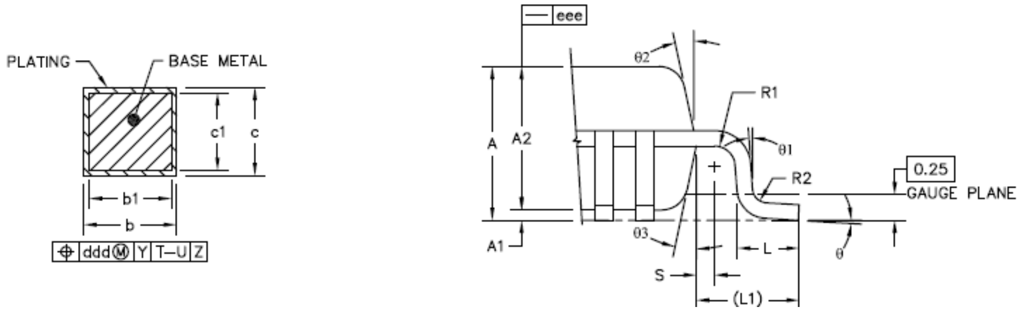
Figure 7-3 SPI Slave Dynamic Characteristics Timing

8 PACKAGE DIMENSIONS

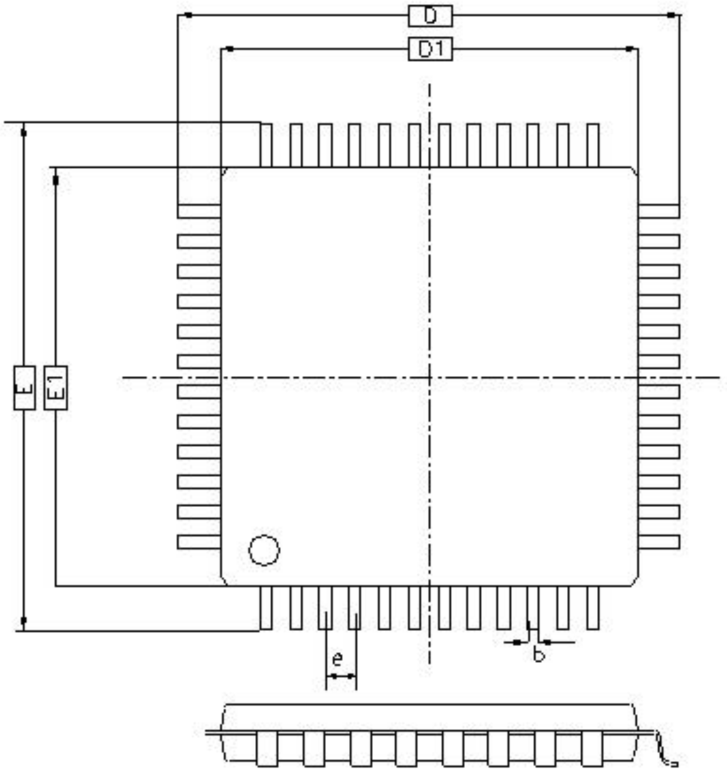
8.1 64L LQFP (7x7x1.4 mm footprint 2.0 mm)

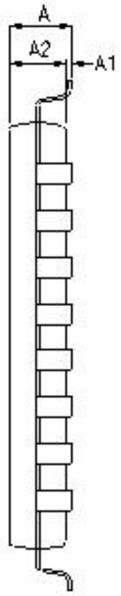


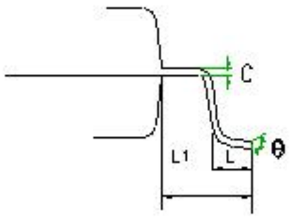
		SYMBOL	MIN	NOM	MAX
TOTAL THICKNESS		A	---	---	1.6
STAND OFF		A1	0.05	---	0.15
MOLD THICKNESS		A2	1.35	1.4	1.45
LEAD WIDTH(PLATING)		b	0.13	0.18	0.23
LEAD WIDTH		b1	0.13	0.16	0.19
L/F THICKNESS(PLATING)		c	0.09	---	0.2
L/F THICKNESS		c1	0.09	---	0.16
	X	D	9 BSC		
	Y	E	9 BSC		
BODY SIZE	X	D1	7 BSC		
	Y	E1	7 BSC		
LEAD PITCH		e	0.4 BSC		
		L	0.45	0.6	0.75
FOOTPRINT		L1	1 REF		
		θ	0°	3.5°	7°
		θ1	0°	---	---
		θ2	11°	12°	13°
		θ3	11°	12°	13°
		R1	0.08	---	---
		R2	0.08	---	0.2
		S	0.2	---	---
PACKAGE EDGE TOLERANCE		aaa	0.2		
LEAD EDGE TOLERANCE		bbb	0.2		
COPLANARITY		ccc	0.08		
LEAD OFFSET		ddd	0.07		
MOLD FLATNESS		eee	0.05		



8.2 48L LQFP (7x7x1.4 mm footprint 2.0 mm)



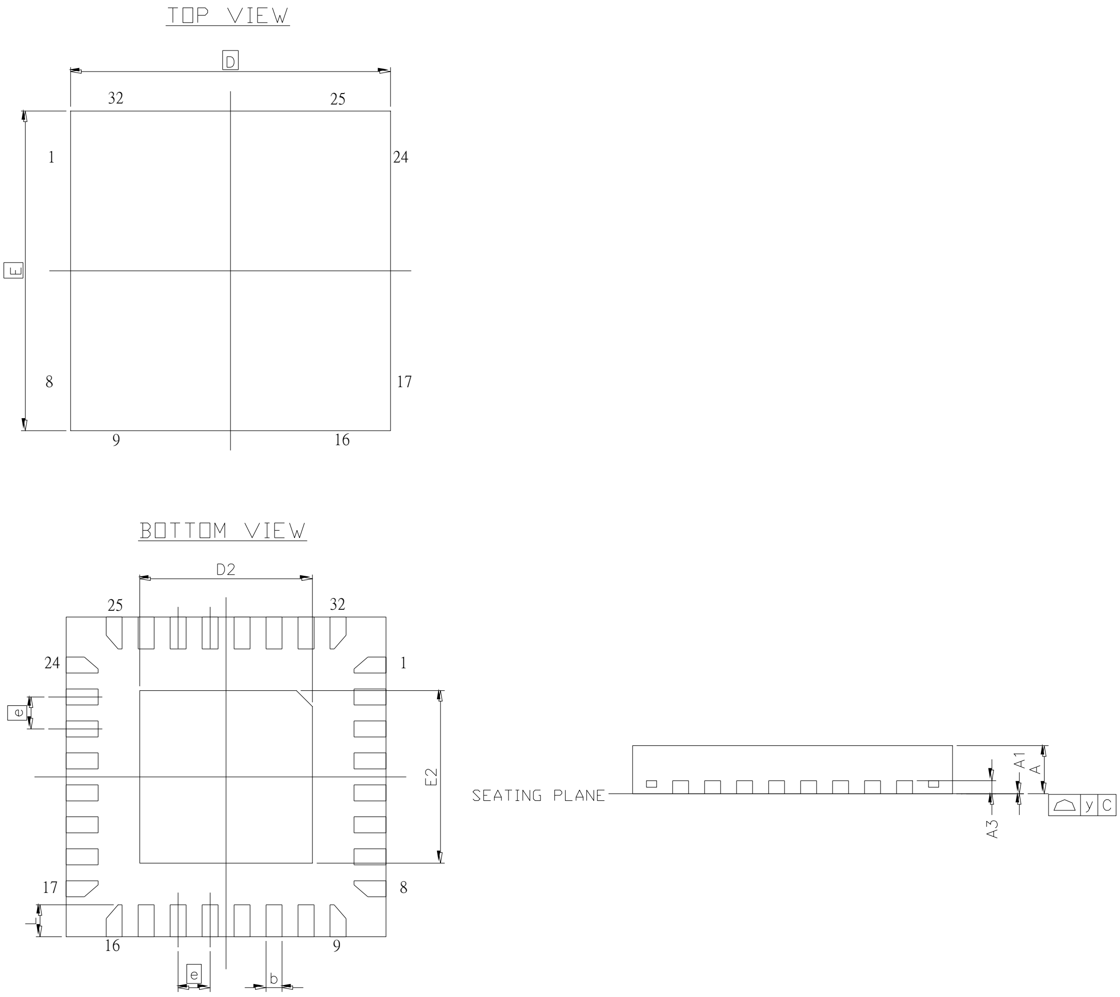




COTROL DIMENSIONS ARE IN MILLIMETERS.

SYMBOL	MILLIMETER			INCH		
	MIN	NOM.	MAX.	MIN	NOM.	MAX.
A	—	—	1.60	—	—	0.063
A1	0.05	0.10	0.15	0.002	0.004	0.006
A2	1 35	1.40	1.45	0.053	0 055	0.057
D1	6.90	7.00	7.10	0.272	0.276	0.260
E1	6 90	7.00	7 10	0.272	0.276	0.260
e	0.35	0.50	0.65	0.014	0.020	0.280
D	8.9	9.00	9.10	0.350	0.354	0.358
E	8.9	9.00	9.10	0.350	0.354	0.358
L	0 45	0.60	0 75	0.018	0 024	0 030
L1	—	1 00	—	—	0.039	—
C	0.09	—	0.20	0.0035	—	0.0079
θ	0°	—	7°	0°	—	7°
b	0.17	0.22	0.27	0.007	0.0087	0.011

8.3 33L QFN (5x5x0.8 mm)



SYMBOL	DIMENSION (MM)			DIMENSION (INCH)		
	MIN.	NOM.	MAX.	MIN.	NOM.	MAX.
A	0.70	0.75	0.80	0.0275	0.0295	0.0315
A1	0	0.02	0.05	0	0.001	0.002
A3	0.20 REF			0.008 REF		
b	0.18	0.25	0.30	0.007	0.010	0.012
D	5.00 BSC			0.197 BSC		
E	5.00 BSC			0.197 BSC		
e	0.50 BSC			0.0197 BSC		
L	0.30	0.40	0.50	0.012	0.016	0.020
y	0.10			0.0039		
D2	3.20	3.40	3.60	0.124	0.138	0.142
E2	3.20	3.40	3.60	0.124	0.138	0.142

9 REVISION HISTORY

VERSION	DATE	PAGE/ CHAPTER	DESCRIPTION
V1.00	Mar 27, 2012	-	Preliminary issued version
V1.01	May 2, 2012	Chap. 7	Update Section 7.2 DC Electrical Characteristics
V1.02	May 7, 2012	Chap. 3	Update Figure 3-1 NuMicro™ NUC123 Series Selection Code
V1.03	May 14, 2012	Chap. 3	Modify the Pin Assignment and Pin Description for LQFP 64-pin and LQFP 48-pin
V1.04	May 30, 2012	Chap. 3 Chap. 5	Modify the Pin Assignment and Pin Description for PC.12 pin. Modify Table 5-8 ISP Mode Command.
V1.05	July 20, 2012	Chap. 5 Chap. 7	Correct the GPC_MFP12 description in GPC_MFP[12] register. Remove RTC_EN description in APBCLK[1] register, modify PD_WU_STS description in PWRCON[6] register and the related RTC function in Table 5-5 (Power-Down Mode Control Table) Remove 32.768kHz crystal characteristics in Section 7.3 AC Electrical Characteristics, modify PWR_DOWN_EN description in PWRCON[7] register and the related 32.768kHz function in Table 5-5 (Power-Down Mode Control Table) Split the SPI_S cell to SPI2_S, SPI1_S and SPI_S in CLKSEL1[6:4] register. Modify all of descriptions about Serial Peripheral Interface (SPI) in whole Section 5.10. Modify the capacitor value from 10uF to 1uF for the LDO pin in Figure 5-1 NuMicro™ NUC123 Power Distribution Diagram and Section 7.4.2 LDO and Power Management Specifications.

V1.06	Aug. 21, 2012	Chap. 2 Chap. 4 Chap. 5 Chap. 7	Update the SPI item for “Master up to 32 MHz, and Slave up to 16 MHz” in Section 2.1 NuMicro™ NUC123 Features. Update the Packages item for “LQFP 64-pin” in Section 2.1 NuMicro™ NUC123 Features. Update Figure 4-1 NuMicro™ NUC123 Block Diagram. Correct the description and reset value of PDID[31:0] register. Correct the description and reset value of GFP_MFP[3:0] register. Correct the description of CRC_CTL[27:26] register. Correct the description of CBS[1:0] in User Configuration Word config0. Correct the MFOM description in FATCON[6] register. Correct the bit field of DMA_PDSSR1[23:16] as reserved. Correct the function description about Single-Cycle Scan Mode and Continuous Scan Mode of ADC in Section 5.17. Update the maximum ADC clock to 6 MHz and ADC conversion rate to 150K SPS in Features of Section 5.17 Analog-to-Digital Converter (ADC). Update the 10-bit SARADC Specifications in Section 7.4 Analog Characteristics. Correct the description of Section 5.5.5 Boot Selection and add the Section 5.5.6 In Application Programming. Add two extra notes in the description of PWM Counter (CNR) register.

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.