ARM®Cortex®-M

32-bit Microcontroller

NuMicro® Family
NM1530 Series
Technical Reference Manual

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

www.nuvoton.com

Table of Contents

List of Figures

List of Tables

1 GENERAL DESCRIPTION

The NuMicro®NM1530Series 32-bit microcontroller is embedded with the newest ARM® Cortex®-M0 core at a cost equivalent to traditional 8-bit microcontroller for industrial control and applications which need high performance.

The NuMicro®NM1530Seriesembedded with the Cortex®-M0 core runsup to 72 MHz and supports a variety of industrial control and applications which need high CPU performance. TheNuMicro®NM1530Seriesprovides 128K/64K/32bytes embedded flash, 4Kbytes data flash, 8Kbytes flash for the ISP, and 16K/8K/4K bytes embedded SRAM. This MCU includes advanced PWM function, MDU (Motor Drive Unit), QEI (Quadrature Encoder Interface) and ECAP (Enhance Input Capture Timer) which are specially designed for motor driving application. It is also equipped with plenty of peripheral devices, such as Timers, Watchdog Timer, UART, SPI, I2C, PWM Timer, GPIO, 12-bit ADC, Low Voltage Detector and Brown-out detector. These useful functions make the NuMicro®NM1530Series powerful for a wide range of applications.

In addition, the NuMicro®NM1530Seriesis equipped with ISP (In-System Programming), ICP (In-Circuit Programming) functions and IAP (In-Application Programming)which allow user to update the program memory without removing the chip from the actual end product.

2 FEATURES

- Core
  - ARM® Cortex®-M0 core running up to 72 MHz
  - One 24-bit system timer
  - Supports Low Power Sleepmode by WFI instructions
  - Single-cycle 32-bit hardware multiplier
  - Supportsprogrammable 4 level priorities of Nested Vectored Interrupt Controller (NVIC)
  - Supports Serial Wire Debug (SWD) support with two watchpoints and four breakpoints
- Built-in LDO for wide operating voltage ranged from 2.5V to 5.5V
- Memory
  - 128K/64K/32K bytes Flash for program memory (APROM)
  - 4KB Flash for data memory (Data Flash)
  - 8KB Flash for loader (LDROM)
  - Supports In-system program (ISP) and In-application program (IAP) application code update
  - Supports 2-wired ICP update through SWD/ICE interface
  - 16K/8K/4K bytes embedded SRAM
- Clock Control
  - Built-in 22.1184 MHz internal high speed RC oscillator (HIRC) for system operation (variation < 2% at -40˚C ~ +105˚C)
  - Built-in 10 kHz internal low speed RC oscillator (LIRC) for Watchdog Timer and wake-up operation
  - Built-in 4~24 MHz external high speed crystal oscillator (HXT) for precise timing operation
  - Supports one PLL up to 72 MHz for high performance system operation, sourced from HIRC and HXT
  - Supportsclock output
- Hardware divider
  - Supports signed 32-bit dividend, 16-bit divisor operation
- GPIO port
  - Four I/O modes:
  - TTL/Schmitt trigger input selectable
  - Bit control available
  - I/O pin configured as interrupt source with edge/level trigger setting
  - Supports high driver and high sink current I/O (up to 16 mA at 5V)
  - INT0 and INT1 pins with individual interrupt vectors
  - Supports up to 82/51/38 GPIOs for LQFP100/64/48 respectively
- Timers
  - Supports 4 sets of 32-bit timers with 24-bit up-timer and one 8-bit prescale counter
  - Provides One-shot, Periodic, Toggle and ContinuousCounting operation modes
  - Supports event counting function to count the event from external pin
- Watchdog Timer
  - Supports multiple clock sourcesfrom LIRC(default selection) and HCLK/2048
  - 8 selectable time-out period from 1.6ms ~ 26.0sec (depending on clock source)
  - Able to wakeup from Power-down or Idle mode
  - Interrupt or reset selectable on watchdog time-out
  - Time-out reset delay period time can be selected
- Window Watchdog Timer
  - Supports multiple clock sources from HCLK/2048 (default selection) and LIRC

- Window set by 6-bit counter with 11-bit prescale
- Able to wakeup from Power-down or Idle mode

- BasicPWM
  - 1 unit of 16-bit basic PWM, up to 2ch output
  - Alternative function as input capture timer
- EnhancedPWM
  - 2 units of 16-bit enhanced PWM, up to 6ch output with dead-zone control, brake and polarity control for motor drive
  - Default tri-state during any reset
- EnhancedInput Capture
  - Up to 2 units of 24-bit input capture
  - Each unit has 3 inputs: ECAPx_IC0, ECAPx_IC1 and ECAPx_IC2
- QEI (Quadrature Encoder Interface)
  - Up to 2 units of Quadrature Encoder Interface
  - Each unit has 3 inputs: QEIA, QEIB and IDX
- MDU (Motor Drive Unit)
  - Built-in Independently mathematical operation modules of PI + FOC + SVPWM
  - Output Space Vector PWM duty to PWM unit 0/1
- UART
  - Up to two 16550 compatible UART devices
  - Programmable baud-rate generator
  - Buffered receiving and transmitting, each with 16 bytes FIFO
  - Supports flow control (TX, RX, CTS and RTS)
  - Supports IrDA(SIR) function
  - Supports RS-485
- SPI
  - Up to three sets of SPI device
  - 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
  - Rx and Tx on both rising or falling edge of serial clock independently
  - Supports Byte Suspend mode in 32-bit transmission
- I^2C
  - Master/Slave up to 1 Mbit/s
  - Bi-directional data transfer between masters and slaves
  - Multi-master bus (no central master)
  - Arbitration between simultaneously transmitting masters
  - Programmable clocks allow versatile rate control
  - Multiple address recognition (four slave address with mask option)
- ADC
  - Two A/D converters
  - Each ADC with up to 8 channel, 12-bit resolution with 10-bit accuracy
  - 16 result registers
  - Sampling rate up to 800ksps
  - Two operating modes:
    - Single Sampling mode: Only one specified channel can be sampled at one time.
    - Simultaneous Sampling mode: Allowing two ADC channels to be sampled simultaneously.

  - Two converting result digital comparators
  - Conversion start by software, external pins, or linked with Timer 0~3 or PWM module
- CAN
  - CAN 2.0B protocol compatible device
  - Support 11-bit identifier as well as 29-bit identifier
  - Bit rates up to 1Mbits/s
  - NRZ bit Coding/ Encoding
  - Error Detection & Status Report
  - Bit error, Form error, Stuffing error, 15-bit CRC detection, and Acknowledge error Interrupt
  - Bit Timing Synchronization
  - Acceptance filter extension
- Up to three Analog Comparators
- Up to two OPA (operational amplifier)
- Brown-out detector
  - 4 levels: 4.4V/3.7V/2.7V/2.2V
  - Optional brown-out interrupt or reset
- Built-in LDO for Wide Operating Voltage Range: 2.5V to 5.5V
- Low Voltage Reset
- 96-bit unique ID
- Operating Temperature: -40℃~105℃
- Develop tools: parallel writer or In-Circuit Programming (ICP) writer
- Packages:
  - LQFP 100/64/48-pin
  - Package is Halogen-free, RoHS-compliant and TSCA-compliant.

3 ABBREVIATIONS

Acronym	Description
ACMP	Analog Comparator Controller
ADC	Analog-to-Digital Converter
AES	Advanced Encryption Standard
APB	Advanced Peripheral Bus
AHB	Advanced High-Performance Bus
BOD	Brown-out Detection
CAN	Controller Area Network
DAP	Debug Access Port
DES	Data Encryption Standard
EBI	External Bus Interface
EPWM	Enhanced Pulse Width Modulation
FIFO	First In, First Out
FMC	Flash Memory Controller
FPU	Floating-point Unit
GPIO	General-Purpose Input/Output
HCLK	The Clock of Advanced High-Performance Bus
HIRC	22.1184 MHz Internal High Speed RC Oscillator
HXT	4~24 MHz External High Speed Crystal Oscillator
IAP	In Application Programming
ICP	In Circuit Programming
ISP	In System Programming
LDO	Low Dropout Regulator
LIN	Local Interconnect Network
LIRC	10 kHz internal low speed RC oscillator (LIRC)
MPU	Memory Protection Unit
NVIC	Nested Vectored Interrupt Controller
PCLK	The Clock of Advanced Peripheral Bus
PDMA	Peripheral Direct Memory Access
PLL	Phase-Locked Loop
PWM	Pulse Width Modulation
QEI	Quadrature Encoder Interface
SDIO	Secure Digital Input/Output
SPI	Serial Peripheral Interface

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

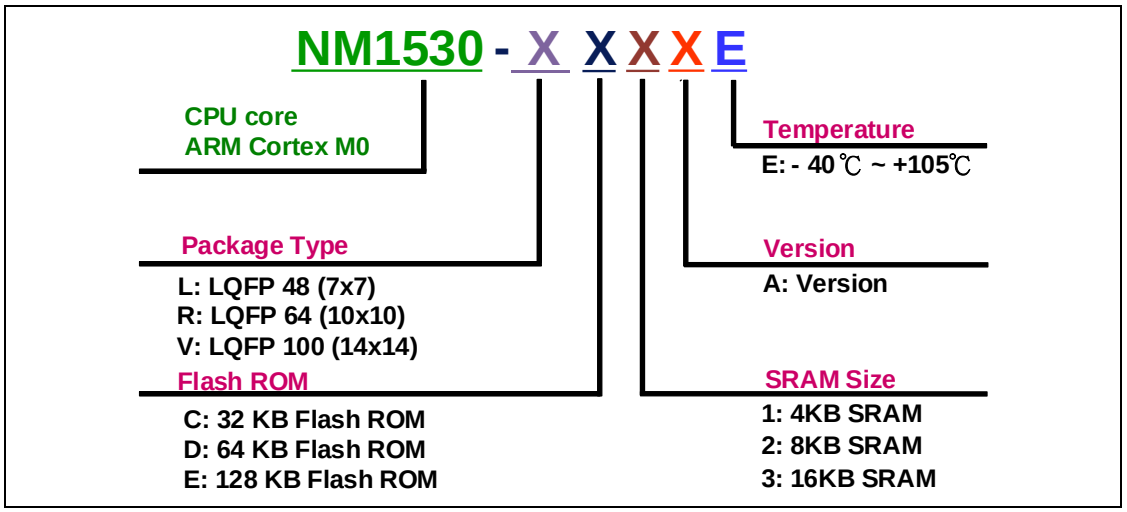
4 PARTS INFORMATION LIST AND PIN CONFIGURATION

4.1 NuMicro®NM1530Selection Guide

4.1.1 NuMicro®NM1530 Selection Guide

	NM1530VE3AE NM1530VD3AE	NM1520RD2AE NM1520RC2AE	NM1520LD2AE NM1520LC2AE	NM1510LC1AE
AP Flash	128KB/64KB	64KB/32KB	64KB/32K	32KB
RAM	16KB	8KB	8KB	4KB
Data Flash	4KB	4KB	4KB	4KB
Timer	4 x 24-bit	4 x 24-bit	4 x 24-bit	4 x 24-bit
PWM	6ch PWM0, 6ch PWM1 2ch PWM2	6ch PWM0, 6ch PWM1 1ch PWM2	6ch PWM0, 3ch PWM1	6ch PWM0, 3ch PWM1
QEI/IC	3ch x 2 sets 3ch x 2 sets	3ch x 1 set Pin shared with QEI0	3ch x 1 set Pin shared with QEI0	3ch x 1 set Pin shared with QEI0
UART	2	2	2	2
SPI	3	1	1	1
I2C	1	1	1	1
CAN 2.0B	1	1	1	-
ADC	8ch ADCA 8ch ADCB	7ch ADCA 7ch ADCB	5ch ADCA 4ch ADCB	5ch ADCA 4ch ADCB
Comparator	CMP0, CMP1, CMP2	CMP1, CMP2	CMP1	CMP1
OP	OP0, OP1	OP0, OP1	OP0, OP1	OP0, OP1
Package	LQFP100(14x14x1.4mm)	LQFP64(10x10x1.4mm)	LQFP48(7x7x1.4mm)	LQFP48(7x7x1.4mm)

4.1.2 NuMicro®NM1530Naming Rule



4.2 Pin Configuration

4.2.1 LQFP 100-pin

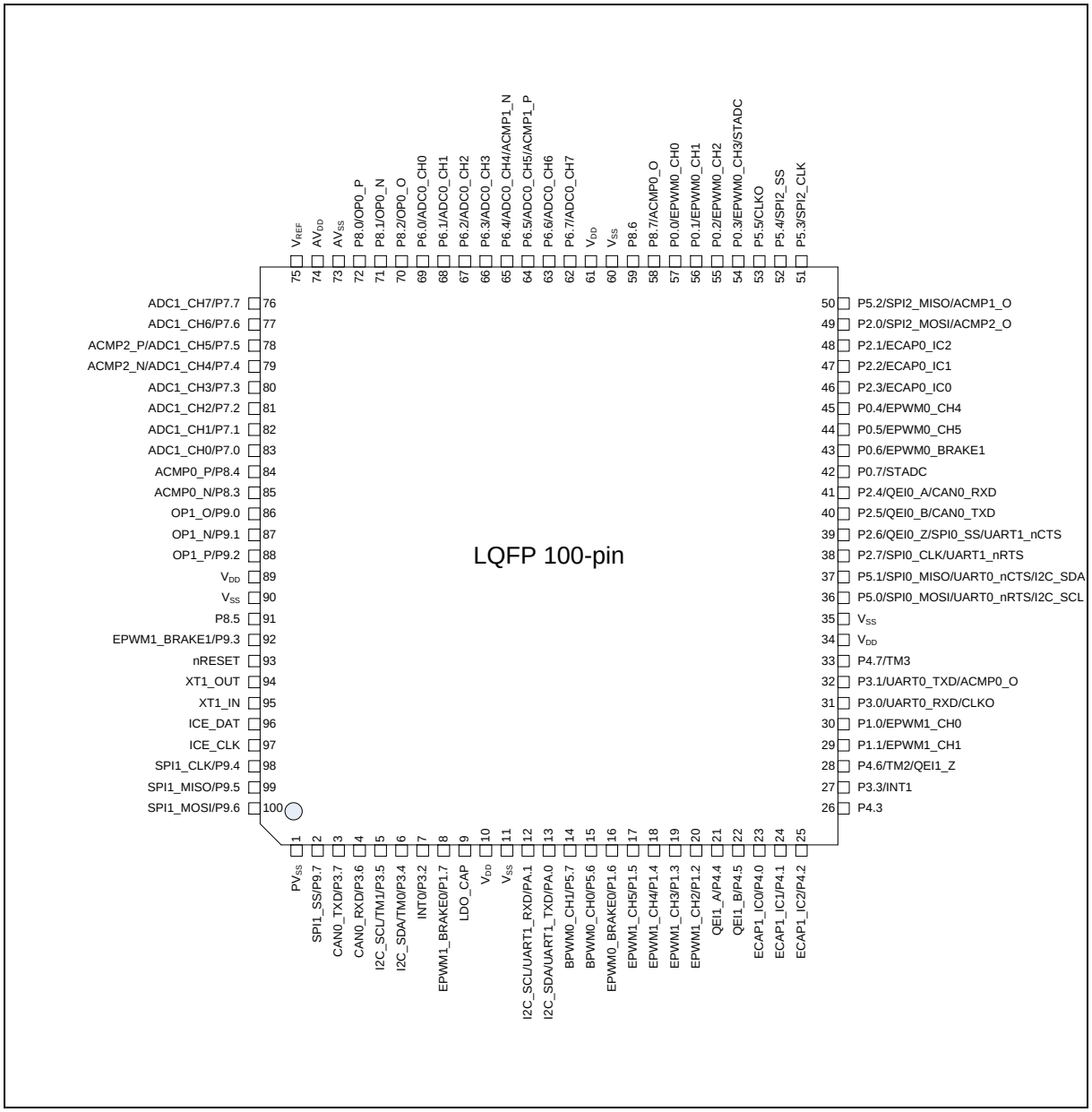


Figure 4–1 NuMicro®NM1530VxxAE Series LQFP-100 Pin Diagram

4.2.2 LQFP 64-pin

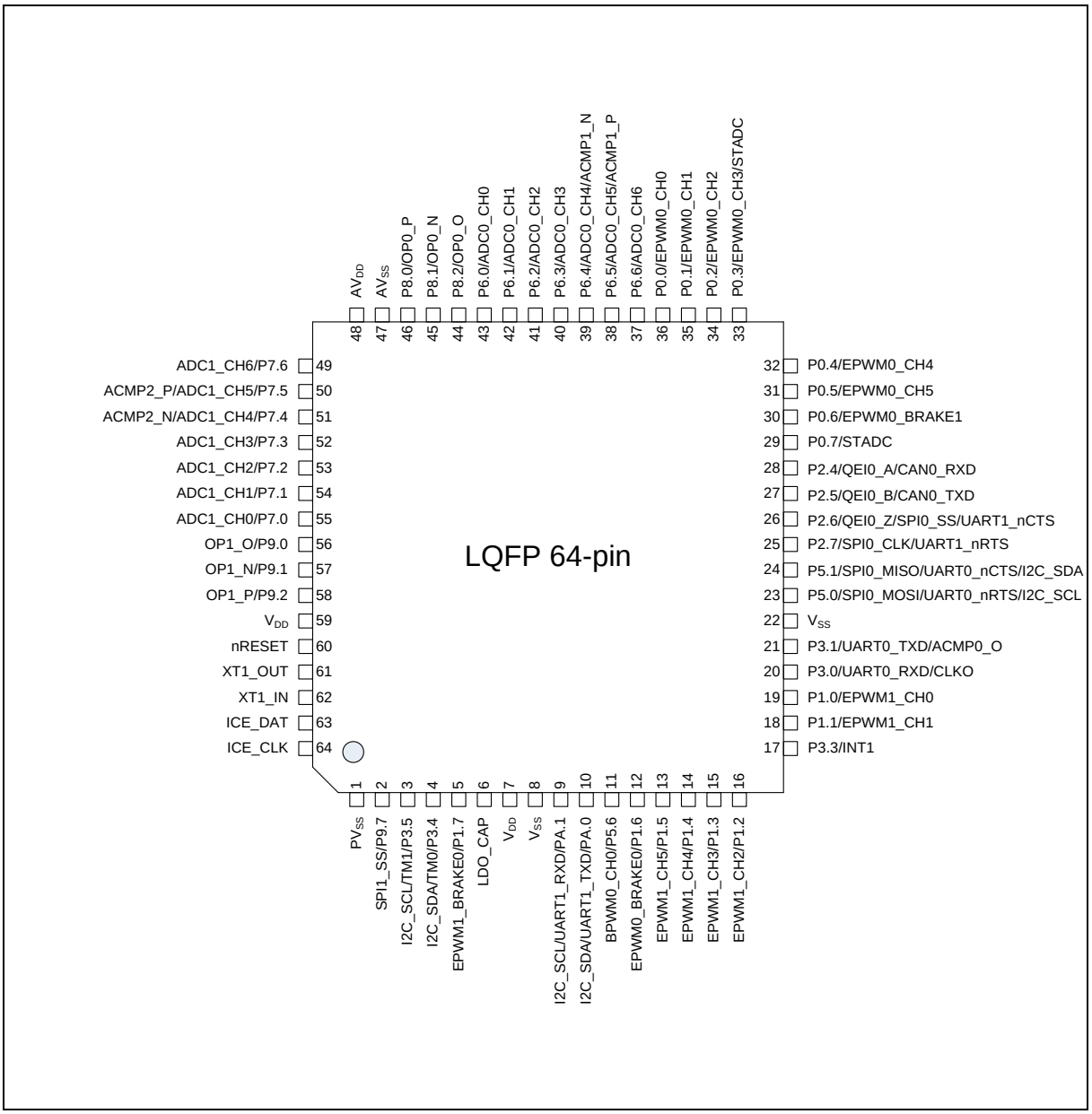


Figure 4–2 NuMicro®NM1520RxxAE Series LQFP-64 Pin Diagram

4.2.3 LQFP 48-pin

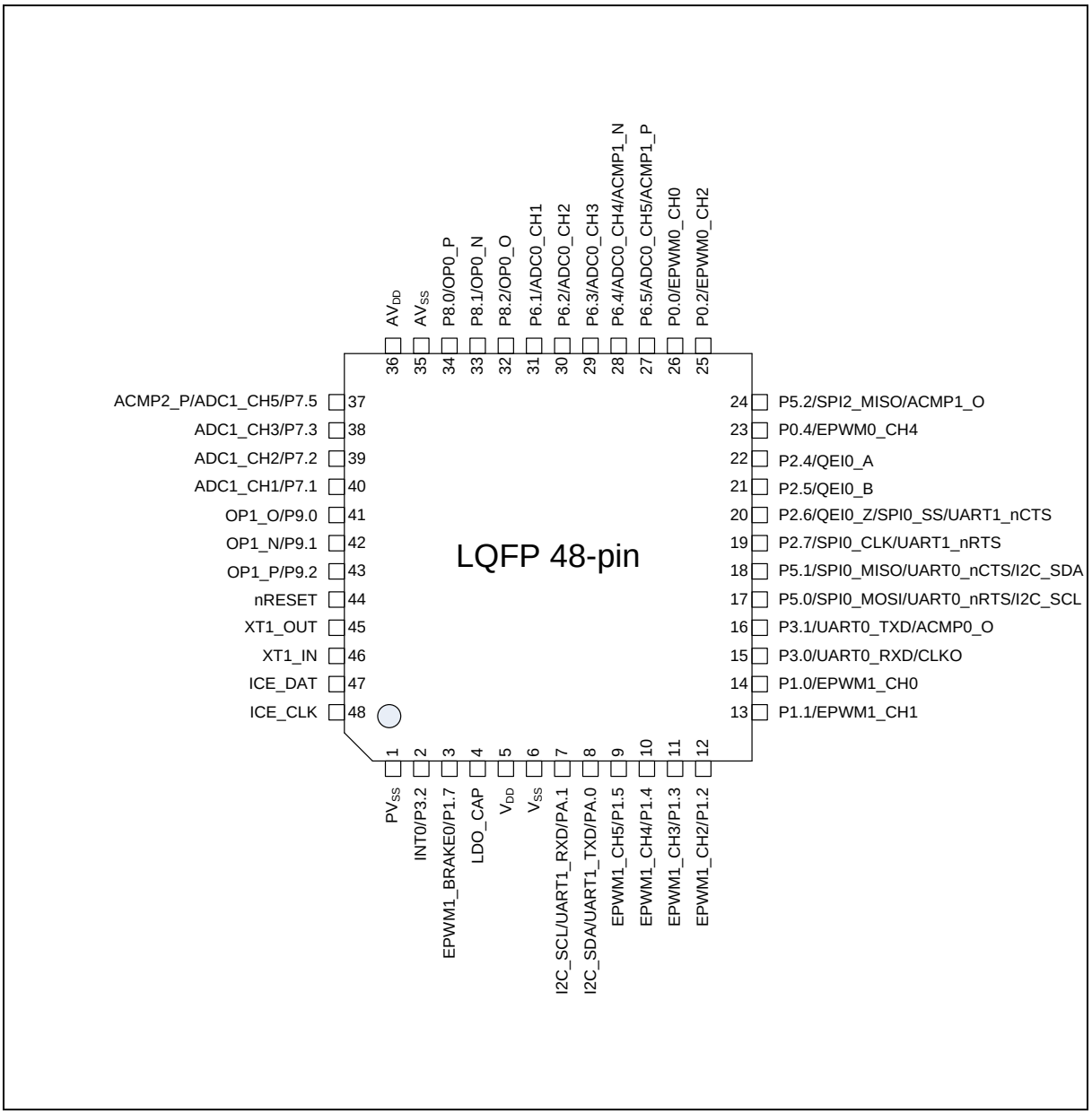


Figure 4–3 NuMicro®NM1520LxxAE/NM1510LxxAE Series LQFP-48 Pin Diagram

4.3 Pin Description

Pin Number			Pin Name	Pin Type[1]	Description
100-pin	64-pin	48-pin			
10	7	5	V_{DD}	**P**	**POWER SUPPLY:** Supply voltage **Digital**V_{DD} for operation.
34					
61	59				
89					
11	8	6	V_{SS}	**P**	**GROUND:**Digital Ground potential.
35					
60	22				
90					
9	6	4	LDO_CAP	**P**	**LDO:** LDO output pin **Note:** It needs to be connected with a 1uF capacitor.
1	1	1	PV_{SS}	**P**	**PLL GROUND: PLL** Ground potential.
74	48	36	AV_{DD}	**AP**	Power supply for internal analog circuit
73	47	35	AV_{SS}	**AP**	Ground Pin for analog circuit
75	-	-	V_{REF}	**AP**	Voltage reference input for ADC **Note:** It needs to be connected with a 1uF capacitor.
93	60	44	nRESET	**I**	**RESET:**nRESET pin is a Schmitt trigger input pin for hardware device reset. A "**Low"** on this pin for 768 clock counter of Internal RC 22.1184 MHz while the system clock is running will reset the device. nRESET pin has an internal pull-up resistor allowing power-on reset by simply connecting an external capacitor to GND.
94	61	45	XT1_OUT	**O**	**CRYSTAL OUT:** This is the output pin from the internal inverting amplifier. It emits the inverted signal of XT1_IN.
95	62	46	XT1_IN	**I**	**CRYSTAL IN:** This is the input pin to the internal inverting amplifier. The system clock is from external crystal or resonator when FOSC[1:0] (CONFIG3[1:0]) are both logic 1 by default.
96	63	47	ICE_DAT	**I/O**	Serial Wired Debugger Data pin
97	64	48	ICE_CLK	**I**	Serial Wired Debugger Clock pin
57	36	26	P0.0	**I/O**	General purpose digital I/O pin
			EPWM0_CH0	**O**	EPWM0 output of EPWM Unit 0
56	35	-	P0.1	**I/O**	General purpose digital I/O pin
			EPWM0_CH1	**O**	EPWM1 output of EPWM Unit 0
55	34	25	P0.2	**I/O**	General purpose digital I/O pin
			EPWM0_CH2	**O**	EPWM2 output of EPWM Unit 0
54	33	-	P0.3	**I/O**	General purpose digital I/O pin
			EPWM0_CH3	**O**	EPWM3 output of EPWM Unit 0

Pin Number			Pin Name	Pin Type[1]	Description
100-pin	64-pin	48-pin			
			STADC	I	ADC external trigger input
45	32	23	P0.4	I/O	General purpose digital I/O pin
			EPWM0_CH4	O	EPWM4 output of EPWM Unit 0
44	31	-	P0.5	I/O	General purpose digital I/O pin
			EPWM0_CH5	O	EPWM5 output of EPWM Unit 0
43	30	-	P0.6	I/O	General purpose digital I/O pin
			EPWM0_BRAKE1	I	Brake input pin 1 of EPWM Unit 0
42	29	-	P0.7	I/O	General purpose digital I/O pin
			STADC	I	ADC external trigger input
30	19	14	P1.0	I/O	General purpose digital I/O pin
			EPWM1_CH0	O	EPWM0 output of EPWM Unit 1
29	18	13	P1.1	I/O	General purpose digital I/O pin
			EPWM1_CH1	O	EPWM1 output of EPWM Unit 1
20	16	12	P1.2	I/O	General purpose digital I/O pin
			EPWM1_CH2	O	EPWM2 output of EPWM Unit 1
19	15	11	P1.3	I/O	General purpose digital I/O pin
			EPWM1_CH3	O	EPWM3 output of EPWM Unit 1
18	14	10	P1.4	I/O	General purpose digital I/O pin
			EPWM1_CH4	O	EPWM4 output of EPWM Unit 1
17	13	9	P1.5	I/O	General purpose digital I/O pin
			EPWM1_CH5	O	EPWM5 output of EPWM Unit 1
16	12	-	P1.6	I/O	General purpose digital I/O pin
			EPWM0_BRAKE0	I	Brake input pin 0 of EPWM Unit 0
8	5	3	P1.7	I/O	General purpose digital I/O pin
			EPWM1_BRAKE0	I	Brake input pin0 of EPWM Unit 1
49	-	-	P2.0	I/O	General purpose digital I/O pin
			SPI2_MOSI	I/O	SPI2MOSI (Master Out, Slave In) pin
			ACMP2_O	AO	Analog comparator 2 output pin
48	-	-	P2.1	I/O	General purpose digital I/O pin
			ECAP0_IC2	I	Input 2 of Enhanced Input Capture Unit 0
47	-	-	P2.2	I/O	General purpose digital I/O pin
			ECAP0_IC1	I	Input 1 of Enhanced Input Capture Unit 0

Pin Number			Pin Name	Pin Type[1]	Description
100-pin	64-pin	48-pin			
46	-	-	P2.3	I/O	General purpose digital I/O pin
			ECAP0_IC0	I	Input 0 of Enhanced Input Capture Unit 0
41	28	22	P2.4	I/O	General purpose digital I/O pin
			QEI0_A	I	Quadrature encoder phase A input of QEI Unit 10
			CAN_RXD	I	CAN Bus RX Input
40	27	21	P2.5	I/O	General purpose digital I/O pin
			QEI0_B	I	Quadrature encoder phase B input of QEI Unit 0
			CAN_TXD	O	CAN Bus TX Output
39	26	20	P2.6	I/O	General purpose digital I/O pin
			QEI0_Z	I	Quadrature Encoder Index input of QEI Unit 0
			SPI0_SS	I/O	SPI0 slave select pin
			UART1_nCTS	I	UART1 CTS pin
38	25	19	P2.7	I/O	General purpose digital I/O pin
			SPI0_CLK	I/O	SPI0serial clock pin
			UART1_nRTS	O	UART1 RTS pin
31	20	15	P3.0	I/O	General purpose digital I/O pin
			UART0_RXD	I	Data Receiver input pin for UART0
			CLKO	O	Clock output
32	21	16	P3.1	I/O	General purpose digital I/O pin
			UART0_TXD	O	Data transmitter output pin for UART0
			ACMP0_O	AO	Analog comparator 0 output
7	-	2	P3.2	I/O	General purpose digital I/O pin
			INT0	I	External Interrupt 0 input pin
27	17	-	P3.3	I/O	General purpose digital I/O pin
			INT1	I	External Interrupt 1 input pin
6	4	-	P3.4	I/O	General purpose digital I/O pin
			TM0	I/O	Timer0 external clock
			I2C0_SDA	I/O	I2C0 data input/output pin
5	3	-	P3.5	I/O	General purpose digital I/O pin
			TM1	I/O	Timer1 external clock
			I2C0_SCL	I/O	I2C0 clock output pin
4	-	-	P3.6	I/O	General purpose digital I/O pin
			CAN_RXD	I	CAN Bus RX Input

Pin Number			Pin Name	Pin Type[1]	Description
100-pin	64-pin	48-pin			
3	-	-	P3.7	I/O	General purpose digital I/O pin
			CAN_TXD	O	CAN Bus TX Output
23	-	-	P4.0	I/O	General purpose digital I/O pin
			ECAP1_IC0	I	Input 0 of Enhanced Input Capture Unit 1
24	-	-	P4.1	I/O	General purpose digital I/O pin
			ECAP1_IC1	I	Input 1 of Enhanced Input Capture Unit 1
25	-	-	P4.2	I/O	General purpose digital I/O pin
			ECAP1_IC2	I	Input 2 of Enhanced Input Capture Unit 1
26	-	-	P4.3	I/O	General purpose digital I/O pin
21	-	-	P4.4	I/O	General purpose digital I/O pin
			QEI1_A	I	Quadarature encoder phase A input of QEI Unit 1
22	-	-	P4.5	I/O	General purpose digital I/O pin
			QEI1_B	I	Quadarature encoder phase B input of QEI Unit 1
28	-	-	P4.6	I/O	General purpose digital I/O pin
			TM2	I/O	Timer2 external clock
			QEI1_Z	I	Quadrature Encoder Index input of QEI Unit 1
33	-	-	P4.7	I/O	General purpose digital I/O pin
			TM3	I/O	Timer3 external clock
36	23	17	P5.0	I/O	General purpose digital I/O pin
			SPI0_MOSI	I/O	SPI0MOSI (Master Out, Slave In) pin
			UART0_nRTS	O	UART0 RTS pin
			I2C_SCL	I/O	I2C clock output pin
37	24	18	P5.1	I/O	General purpose digital I/O pin
			SPI0_MISO	I/O	SPI0MISO (Master In, Slave Out) pin
			UART0_nCTS	I	UART0 CTS pin
			I2C_SDA	I/O	I2C data input/output pin
50	-	24	P5.2	I/O	General purpose digital I/O pin
			SPI2_MISO	I/O	SPI2MISO (Master In, Slave Out) pin
			ACMP1_O	AO	Analog comparator 1 output pin
51	-	-	P5.3	I/O	General purpose digital I/O pin
			SPI2_CLK	I/O	SPI2serial clock pin
52	-	-	P5.4	I/O	General purpose digital I/O pin
			SPI2_SS	I/O	SPI2 slave select pin

Pin Number			Pin Name	Pin Type[1]	Description
100-pin	64-pin	48-pin			
53	-	-	P5.5	I/O	General purpose digital I/O pin
			CLKO	O	Frequency Divider output pin
15	11	-	P5.6	I/O	General purpose digital I/O pin
			BPWM_CH0	I/O	BPWM0 output of BPWM
14	-	-	P5.7	I/O	General purpose digital I/O pin
			EPWM2_CH1	I/O	BPWM1 output of BEPWM
69	43	-	P6.0	I/O	General purpose digital I/O pin
			ADC0_CH0	AI	ADC analog input 0 for sample-and-hold A
68	42	31	P6.1	I/O	General purpose digital I/O pin
			ADC0_CH1	AI	ADC analog input 1 for sample-and-hold A
67	41	30	P6.2	I/O	General purpose digital I/O pin
			ADC0_CH2	AI	ADC analog input 2 for sample-and-hold A
66	40	29	P6.3	I/O	General purpose digital I/O pin
			ADC0_CH3	AI	ADC analog input 3 for sample-and-hold A
65	39	28	P6.4	I/O	General purpose digital I/O pin
			ADC0_CH4	AI	ADC analog input 4 for sample-and-hold A
			ACMP1_N	AI	Analog comparator 1 negative input
64	38	27	P6.5	I/O	General purpose digital I/O pin
			ADC0_CH5	AI	ADC analog input 5 for sample-and-hold A
			ACMP1_P	AI	Analog comparator 1 positive input
63	37	-	P6.6	I/O	General purpose digital I/O pin
			ADC0_CH6	AI	ADC analog input 6 for sample-and-hold A
62	-	-	P6.7	I/O	General purpose digital I/O pin
			ADC0_CH7	AI	ADC analog input 7 for sample-and-hold A
83	55	-	P7.0	I/O	General purpose digital I/O pin
			ADC1_CH0	AI	ADC analog input 0 for sample-and-hold B
82	54	40	P7.1	I/O	General purpose digital I/O pin
			ADC1_CH1	AI	ADC analog input 1 for sample-and-hold B
81	53	39	P7.2	I/O	General purpose digital I/O pin
			ADC1_CH2	AI	ADC analog input 2 for sample-and-hold B
80	52	38	P7.3	I/O	General purpose digital I/O pin
			ADC1_CH3	AI	ADC analog input 3 for sample-and-hold B
79	51	-	P7.4	I/O	General purpose digital I/O pin

Pin Number			Pin Name	Pin Type[1]	Description
100-pin	64-pin	48-pin			
			ADC1_CH4	AI	ADC analog input 4 for sample-and-hold B
			ACMP2_N	AI	Analog comparator 2 negative input
78	50	37	P7.5	I/O	General purpose digital I/O pin
			ADC1_CH5	AI	ADC analog input 5 for sample-and-hold B
			ACMP2_P	AI	Analog comparator 2 positive input
77	49	-	P7.6	I/O	General purpose digital I/O pin
			ADC1_CH6	AI	ADC analog input 6 for sample-and-hold B
76	-	-	P7.7	I/O	General purpose digital I/O pin
			ADC1_CH7	AI	ADC analog input 7 for sample-and-hold B
72	46	34	P8.0	I/O	General purpose digital I/O pin
			OP0_P	AI	OP Amplifier 0 positive input
71	45	33	P8.1	I/O	General purpose digital I/O pin
			OP0_N	AI	OP Amplifier 0 negative input
70	44	32	P8.2	I/O	General purpose digital I/O pin
			OP0_O	AO	OP Amplifier 0 output
85	-	-	P8.3	I/O	General purpose digital I/O pin
			ACMP0_N	AI	Analog comparator negative input pin
84	-	-	P8.4	I/O	General purpose digital I/O pin
			ACMP0_P	AI	Analog comparator positive input pin
91	-	-	P8.5	I/O	General purpose digital I/O pin
59	-	-	P8.6	I/O	General purpose digital I/O pin
58	-	-	P8.7	I/O	General purpose digital I/O pin
			ACMP0_O	O	Analog comparator output pin
86	56	41	P9.0	I/O	General purpose digital I/O pin
			OP1_O	AO	OP Amplifier 1 output
87	57	42	P9.1	I/O	General purpose digital I/O pin
			OP1_N	AI	OP Amplifier 1 negative input
88	58	43	P9.2	I/O	General purpose digital I/O pin
			OP1_P	AI	OP Amplifier 1 positive input
92	-	-	P9.3	I/O	General purpose digital I/O pin
			PWM1_BRAKE1	I	Brake input pin1 of PWM Unit 1
98	-	-	P9.4	I/O	General purpose digital I/O pin
			SPI1_CLK	I/O	SPI1serial clock pin

Pin Number			Pin Name	Pin Type[1]	Description
100-pin	64-pin	48-pin			
99	-	-	P9.5	**I/O**	General purpose digital I/O pin
			SPI1_MISO	**I/O**	SPI1MISO (Master In, Slave Out) pin
100	-	-	P9.6	**I/O**	General purpose digital I/O pin
			SPI1_MOSI	**I/O**	SPI1MOSI (Master Out, Slave In) pin
2	2	-	P9.7	**I/O**	General purpose digital I/O pin
			SPI1_SS	**I/O**	SPI1 slave select pin
13	10	8	PA.0	**I/O**	General purpose digital I/O pin
			UART1_TXD	**O**	Data transmitter output pin for UART1
			I2C0_SDA	**I/O**	I2C0 data input/output pin
12	9	7	PA.1	**I/O**	General purpose digital I/O pin
			UART1_RXD	**I**	Data Receiver input pin for UART1
			I2C0_SCL	**I/O**	I2C0 clock output pin

Note:

1. Pin Type I = Digital Input, O = Digital Output; AI = Analog Input; P = Power Pin; AP = Analog Power
2. Do not leave the pins ICE_CLK and ICE_DAT in floating when MCU is in operatoin. User may refer to one of the following methods
    a. Add external pull-up or pull-low resistors at pins.
    b. Be wired to other deivce without floating.

5 BLOCK DIAGRAM

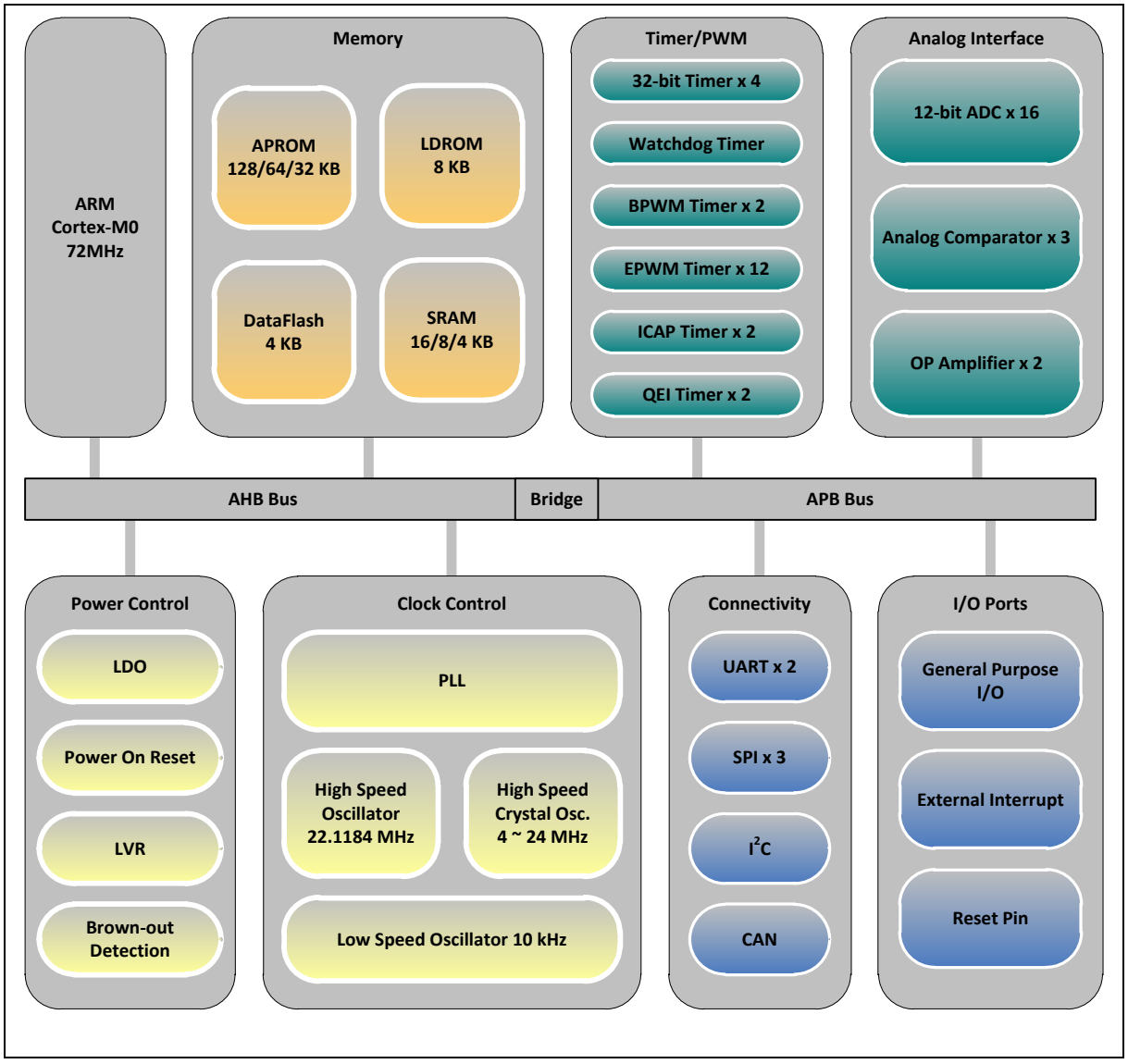


Figure 5-1Nuicro®NM1530 Series Block Diagram

6 FUNCTIONAL DESCRIPTION

6.1 ARM® Cortex®-M0 Core

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

Figure 6–1shows the functional controller of processor.

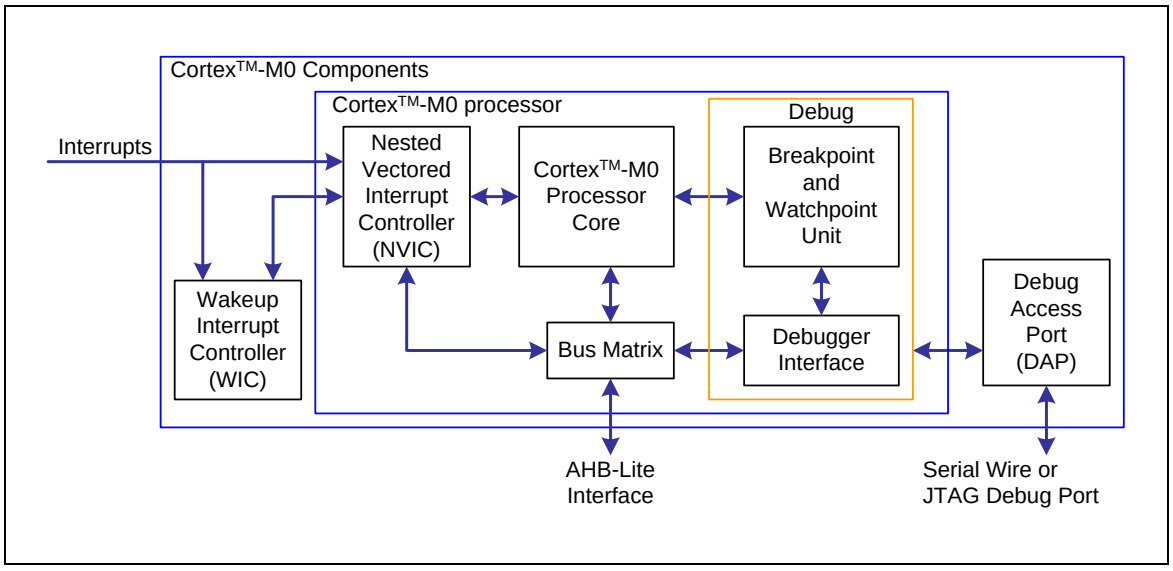


Figure 6–1 Functional Controller Diagram

The implemented device provides the following components and 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 supportedwith little-endian data accesses
  - Ability to have deterministic, fixed-latency, interrupt handling
  - Load/store-multiples and multicycle-multiplies that can be abandoned and restarted to facilitate rapid interrupt handling
  - C Application Binary Interface compliant exception model. This is the ARMv6-M, C Application Binary Interface (C-ABI) compliant exception model that enables the use of pure C functions as interrupt handlers
  - Low Power Sleepmode 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 for both level-sensitive and pulse-sensitive interrupt lines
- Supports Wake-up Interrupt Controller (WIC) and, providing 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 that provides simple integration to all system peripherals and memory
  - Single 32-bit slave port that supports the DAP (Debug Access Port)

6.2 System Manager

6.2.1 Overview

System management includes the following sections:

- System Resets
- System Power Distribution
- System Memory Map
- System management registersfor 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

6.2.2 System Reset

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

- Hardware Reset
  - Power-on Reset (POR)
  - Low level on the Reset pin (nRESET)
  - Watchdog Time-out Reset (WDT)
  - Low Voltage Reset (LVR)
  - Brown-out Detector Reset (BOD)
- Software Reset
  - SYSReset - SYSRESETREQ(AIRCR[2])
  - Cortex®-M0 Core One-shot Reset – CPU_RST(IPRSTC1[1])
  - Chip One-shot Reset – CHIP_RST(IPRSTC1[0])

Power-on Reset or CHIP_RST (IPRST1[0]) reset the whole chip including all peripherals, external crystal circuit and BS (ISPCON[1]) bit.

SYSRESETREQ (AIRCR[2]) reset the whole chip including all peripherals, but does not reset external crystal circuit and BS (ISPCON[1]) bit.

6.2.3 System Power Distribution

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

- Analog power from AV_{DD} and AV_{SS} provides the power for analog components operation.
- Digital power from V_{DD} and V_{SS} supplies the power to the I/O pins and internal regulator which provides a fixed 1.8V power for digital operation.

The output of internal voltage regulators, LDO_CAP, requires an external capacitor which should be located close to the corresponding pin. Analog power (AV_{DD}) should be the same voltage level of the digital power (V_{DD}).

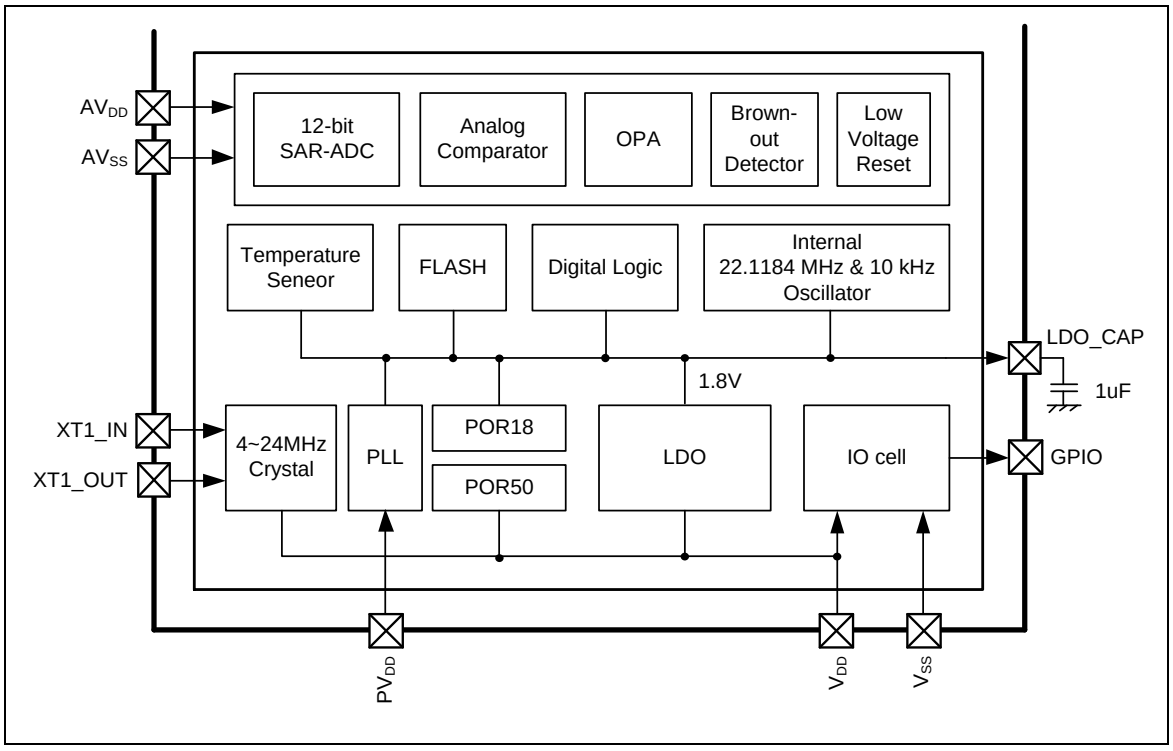


Figure 6-2 NuMicro®NM1530 Series Power Distribution Diagram

6.2.4 System Memory Map

The NuMicro®NM1530Seriesprovides 4G-byte addressing space. The memory locations assigned to each on-chip controllers are shown in theTable 6-1. The detailed register definition, memory space, and programming detailed will be described in the following sections for each on-chip peripheral. TheNuMicro®NM1530Seriesonly supports little-endian data format.

Address Space	Token	Controllers
Flash and SRAM Memory Space		
0x0000_0000 – 0x0001_FFFF	FLASH_BA	FLASH Memory Space (128KB)
0x2000_0000 – 0x2000_3FFF	SRAM_BA	SRAM Memory Space (16KB)
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_C000 – 0x5000_FFFF	FMC_BA	Flash Memory Control Registers
0x5001_4000 – 0x5001_7FFF	HDIV_BA	Hardware Divider Register
APB1 Controllers Space (0x4000_0000 ~ 0x400F_FFFF)		
0x4000_4000 – 0x4000_7FFF	WDT_BA	Watchdog Timer Control Registers
0x4000_4100 – 0x4000_7FFF	WWDT_BA	Window Watchdog Timer Control Registers
0x4001_0000 – 0x4001_3FFF	TMR01_BA	Timer0/Timer1 Control Registers
0x4002_0000 – 0x4002_3FFF	I2C0_BA	I^2C0 Interface Control Registers
0x4003_0000 – 0x4003_3FFF	SPI0_BA	SPI0 with master/slave function Control Registers
0x4003_4000 – 0x4003_7FFF	SPI1_BA	SPI1 with master/slave function Control Registers
0x4004_0000 – 0x4004_3FFF	BPWM0_BA	Basic PWM0 Control Registers
0x4005_0000 – 0x4005_3FFF	UART0_BA	UART0 Control Registers
0x400D_0000 – 0x400D_3FFF	ACMP_BA	Analog Comparator Control Registers
0x400E_0000 – 0x400E_3FFF	EADC_BA	Enhanced Analog-Digital-Converter (EADC) Control Registers
0x400F_0000 – 0x400F_3FFF	OPA_BA	Operation Amplifier Control Registers
APB2 Controllers Space (0x4010_0000 ~ 0x401F_FFFF)		
0x4011_0000 – 0x4011_3FFF	TMR23_BA	Timer2/Timer3 Control Registers
0x4013_0000 – 0x4013_3FFF	SPI2_BA	SPI2 with master/slave function Control Registers
0x4015_0000 – 0x4015_3FFF	UART1_BA	UART1 Control Registers
0x4018_0000 – 0x4018_3FFF	CAN_BA	CAN Bus Control Registers
0x4019_0000 – 0x4019_3FFF	EPWM0_BA	Enhanced PWM0 Control Registers
0x4019_4000 – 0x4019_7FFF	EPWM1_BA	Enhanced PWM1 Control Registers

Address Space	Token	Controllers
0x401B_0000 – 0x401B_3FFF	ECAP0_BA	Enhanced Input Capture 0 Control Registers
0x401B_4000 – 0x401B_7FFF	ECAP1_BA	Enhanced Input Capture 1 Control Registers
0x401C_0000 – 0x401C_3FFF	QEI0_BA	Quadrature Encoder Interface 0 Control Registers
0x401C_4000 – 0x401C_7FFF	QEI1_BA	Quadrature Encoder Interface 1 Control Registers
0x401D_0000 – 0x401D_3FFF	MDU_BA	Motor Drive Unit Control Registers
System Controllers Space (0xE000_E000 ~ 0xE000_EFFF)		
0xE000_E010 – 0xE000_E01F	SYST_BA	System Timer Control Registers
0xE000_E100 – 0xE000_E4EF	NVIC_BA	External Interrupt Controller Control Registers
0xE000_ED00 – 0xE000_ED3F	SCS_BA	System Control Registers

Table 6-1Address Space Assignments for On-Chip Controllers

6.2.5 System Manager Controller Register Map

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

Register	Offset	R/W	Description	Reset Value
GCR Base Address: **GCR_BA = 0x5000_0000**				
PDID	GCR_BA+0x00	R	Part Device Identification Number Register	0x0035_X0XX
RSTSRC	GCR_BA+0x04	R/W	System Reset Source Register	0x0000_00XX
IPRSTC1	GCR_BA+0x08	R/W	Peripheral Reset Control Register1	0x0000_0000
IPRSTC2	GCR_BA+0x0C	R/W	PeripheralReset Control Register2	0x0000_0000
BODCR	GCR_BA+0x18	R/W	Brown-out Detector Control Register	0x0000_008X
TEMPCR	GCR_BA+0x1C	R/W	Temperature Sensor Control Register	0x0000_0000
PORCR	GCR_BA+0x24	R/W	Power-onReset Controller Register	0x0000_00XX
P0_MFP	GCR_BA+0x30	R/W	P0 Multiple Function and Input Type Control Register	0x0000_0000
P1_MFP	GCR_BA+0x34	R/W	P1 Multiple Function and Input Type Control Register	0x0000_0000
P2_MFP	GCR_BA+0x38	R/W	P2 Multiple Function and Input Type Control Register	0x0000_0000
P3_MFP	GCR_BA+0x3C	R/W	P3 Multiple Function and Input Type Control Register	0x0000_0000
P4_MFP	GCR_BA+0x40	R/W	P4 Multiple Function and Input Type Control Register	0x0000_0000
P5_MFP	GCR_BA+0x44	R/W	P5 Multiple Function and Input Type Control Register	0x0000_0000
P6_MFP	GCR_BA+0x48	R/W	P6 Multiple Function and Input Type Control Register	0x0000_0000
P7_MFP	GCR_BA+0x4C	R/W	P7 Multiple Function and Input Type Control Register	0x0000_0000
P8_MFP	GCR_BA+0x50	R/W	P8 Multiple Function and Input Type Control Register	0x0000_0000
P9_MFP	GCR_BA+0x54	R/W	P9 Multiple Function and Input Type Control Register	0x0000_0000
PA_MFP	GCR_BA+0x58	R/W	PA Multiple Function and Input Type Control Register	0x0000_0000
REGWRPROT	GCR_BA+0x100	R/W	Register Write-Protection Control Register	0x0000_0000

6.2.6 Register Description

Part Device Identification Number Register (PDID)

Register	Offset	R/W	Description	Reset Value
PDID	GCR_BA+0x00	R	Part Device Identification Number Register	0x0035_X0XX[1]

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

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

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

System Reset Source Register (RSTSRC)

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

Register	Offset	R/W	Description	Reset Value
RSTSRC	GCR_BA+0x04	R/W	System Reset Source Register	0x0000_00XX

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**RSTS_CPU**	**CPU Reset Flag** The RSTS_CPU flag is set by hardware if software writes CPU_RST (IPRSTC1[1]) 1 to reset Cortex®-M0 core and Flash memory controller (FMC). 0 = No reset from CPU. 1 = Cortex®-M0 CPU kernel and FMC are reset by software setting CPU_RST(IPRSTC1[1]) to 1. **Note:**Write 1 to clear this bit to 0.
[6]	**Reserved**	Reserved.
[5]	**RSTS_SYS**	**SYS Reset Flag** The RSTS_SYS flag is set by the "Reset Signal" from the Cortex®-M0 kernel to indicate the previous reset source. 0 = No reset from Cortex®-M0. 1 = The Cortex®-M0 had issued the reset signal to reset the system by writing 1 to bit SYSRESETREQ (AIRCR[2], Application Interrupt and Reset Control Register, address = 0xE000ED0C) in system control registers of Cortex®-M0 core. **Note:** Write 1 to clear this bit to 0.
[4]	**RSTS_BOD**	**Brown-out Detector Reset Flag** The RSTS_BOD flag is set by the "Reset Signal" from the Brown-out Detector to indicate the previous reset source. 0 = No reset from BOD. 1 = The BOD had issued the reset signal to reset the system. **Note:** Write 1 to clear this bit to 0.

Bits	Description	
[3]	**RSTS_LVR**	**Low Voltage Reset Flag** The RSTS_LVR flag is set by the “Reset Signal” from the Low-Voltage-Reset controller to indicate the previous reset source. 0 = No reset from LVR. 1 = The LVR controller had issued the reset signal to reset the system. **Note:** Write 1 to clear this bit to 0.
[2]	**RSTS_WDT**	**Watchdog Timer Reset Flag** The RSTS_WDT flag is set by the “Reset Signal” from the watchdog timer or window watchdog timer to indicate the previous reset source. 0 = No reset from watchdog timer or window watchdog timer. 1 = The watchdog timer or window watchdog timerhad issued the reset signal to reset the system. **Note1:** Write 1 to clear this bit to 0. **Note2:**Watchdog Timer register WTRF(WTCR[2]) bit is set if the system has been reset by WDT time-out reset. Window Watchdog Timer register WWDTRF(WWDTSR) bit is set if the system has been reset by WWDT time-out reset.
[1]	**RSTS_RESET**	**Reset Pin Reset Flag** The RSTS_RESET flag is set by the “Reset Signal” from the nRESET pin to indicate the previous reset source. 0 = No reset from the nRESET pin. 1 = The nRESET pin had issued the reset signal to reset the system. **Note:** Write 1 to clear this bit to 0.
[0]	**RSTS_POR**	**Power-on Reset Flag** 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. 0 = No reset from POR or CHIP_RST (IPRSTC1[0]). 1 = Power-on Reset (POR) or CHIP_RST (IPRSTC1[0]) had issued the reset signal to reset the system. **Note:** Write 1 to clear this bit to 0.

Peripheral Reset Control Register1 (IPRSTC1)

Register	Offset	R/W	Description	Reset Value
IPRSTC1	GCR_BA+0x08	R/W	Peripheral Reset Control Register1	0x0000_0000

31	30	29	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			**HDIV_RST**	**Reserved**		**CPU_RST**	**CHIP_RST**

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**HDIV_RST**	**HDIV Controller Reset (Write Protect)** Setting this bit to 1 will generate a reset signal to the hardware divider. User need to set this bit to 0 to release from the reset state. 0 = Hardware divider controller normal operation. 1 = Hardware divider controllerreset. **Note:**This bit is write protected. Refer to the REGWRPROT register.
[3:2]	**Reserved**	Reserved.
[1]	**CPU_RST**	**Cortex®-M0 Core One-shot Reset (Write Protect)** Setting this bit will only reset the CPU kernel and Flash Memory Controller (FMC), and this bit will automatically return 0 after the two clock cycles. 0 = CPU normal operation. 1 = CPU one-shot reset. **Note:**This bit is write protected. Refer to the REGWRPROT register.
[0]	**CHIP_RST**	**Chip One-shot Reset (Write Protect)** 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 the same as the POR reset. All the chip controllers are reset and the chip setting from flash are also reload. 0 = Chip normal operation. 1 = Chip one-shot reset. **Note:**This bit is write protected. Refer to the REGWRPROT register.

Peripheral Reset Control Register2 (IPRSTC2)

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

Register	Offset	R/W	Description	Reset Value
IPRSTC2	GCR_BA+0x0C	R/W	Peripheral Reset Control Register2	0x0000_0000

31	30	29	28	27	26	25	24
QEI1_RST	QEI0_RST	OPA_RST	EADC_RST	ECAP1_RST	ECAP0_RST	Reserved	CAN_RST
23	22	21	20	19	18	17	16
Reserved	ACMP_RST	EPWM1_RST	EPWM0_RST	BPWM0_RST	MDU_RST	UART1_RST	UART0_RST
15	14	13	12	11	10	9	8
Reserved	SPI2_RST	SPI1_RST	SPI0_RST	Reserved			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]	QEI1_RST	**QEI1 Controller Reset** 0 = QEI1 controller normal operation. 1 = QEI1 controller reset.
[30]	QEI0_RST	**QEI0 Controller Reset** 0 = QEI0 controller normal operation. 1 = QEI0 controller reset.
[29]	OPA_RST	**OPA0 and OPA1 Controller Reset** 0 = OPA0 and OPA1 controller normal operation. 1 = OPA0 and OPA1 controller reset.
[28]	EADC_RST	**EADC Controller Reset** 0 = EADC controller normal operation. 1 = EADC controller reset.
[27]	ECAP1_RST	**Enhanced Input Capture 1Controller Reset** 0 = Enhanced input capture 1 controller normal operation. 1 = Enhanced input capture 1 controller reset.
[26]	ECAP0_RST	**Enhanced Input Capture 0Controller Reset** 0 = Enhanced input capture 0 controller normal operation. 1 = Enhanced input capture 0 controller reset.
[25]	Reserved	Reserved.
[24]	CAN_RST	**CANController Reset** 0 = CAN controller normal operation. 1 = CAN controller reset.
[23]	Reserved	Reserved.

Bits	Description	
[22]	**ACMP_RST**	**Analog Comparator Controller Reset** 0 = Analog Comparator controller normal operation. 1 = Analog Comparator controller reset.
[21]	**EPWM1_RST**	**Enhanced PWM1Controller Reset** 0 = EPWM1 controller normal operation. 1 = EPWM1 controller reset.
[20]	**EPWM0_RST**	**Enhanced PWM0 Controller Reset** 0 = EPWM0 controller normal operation. 1 = EPWM0 controller reset.
[19]	**BPWM0_RST**	**Basic PWM0Controller Reset** 0 = Basic PWM0 controller normal operation. 1 = Basic PWM0 controller reset.
[18]	**MDU_RST**	**MDUController Reset** 0 = MDU controller normal operation. 1 = MDU controller reset.
[17]	**UART1_RST**	**UART1 Controller Reset** 0 = UART1 controller normal operation. 1 = UART1 controller reset.
[16]	**UART0_RST**	**UART0 Controller Reset** 0 = UART0 controller normal operation. 1 = UART0 controller reset.
[15]	**Reserved**	Reserved.
[14]	**SPI2_RST**	**SPI2 Controller Reset** 0 = SPI2 controller normal operation. 1 = SPI2 controller reset.
[13]	**SPI1_RST**	**SPI1 Controller Reset** 0 = SPI1 controller normal operation. 1 = SPI1 controller reset.
[12]	**SPI0_RST**	**SPI0 Controller Reset** 0 = SPI0 controller normal operation. 1 = SPI0 controller reset.
[11:9]	**Reserved**	Reserved.
[8]	**I2C0_RST**	**I2C0Controller Reset** 0 = I^2C0 controller normal operation. 1 = I^2C0 controller reset.
[7:6]	**Reserved**	Reserved.
[5]	**TMR3_RST**	**Timer3 Controller Reset** 0 = Timer3 controller normal operation. 1 = Timer3 controller reset.
[4]	**TMR2_RST**	**Timer2 Controller Reset** 0 = Timer2 controller normal operation. 1 = Timer2 controller reset.

Bits	Description	
[3]	**TMR1_RST**	**Timer1 Controller Reset** 0 = Timer1 controller normal operation. 1 = Timer1 controller reset.
[2]	**TMR0_RST**	**Timer0 Controller Reset** 0 = Timer0 controller normal operation. 1 = Timer0 controller reset.
[1]	**GPIO_RST**	**GPIO Controller Reset** 0 = GPIO controller normal operation. 1 = GPIO controller reset.
[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 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 Control(Write Protect)** The LVR function reset the chip when the input power voltage is lower than LVR circuit setting. LVR function is enabled by default. 0 = Low Voltage Reset function Disabled. 1 = Low Voltage Reset function Enabled – After enabling the bit, the LVR function will be active with 100us delay for LVR output stable (default). **Note:** This bit is write protected. Refer to the REGWRPROT register.
[6]	**BOD_OUT**	**Brown-out DetectorOutput Status** 0 = Brown-out Detector output status is 0.It means the detected voltage is higher than BOD_VL setting or BOD_EN is 0. 1 = Brown-out Detector output status is 1.Itmeans the detected voltage is lower than BOD_VL setting. If the BOD_EN is 0, BOD function disabled, this bit always responds to 0.
[5]	**BOD_LPM**	**Brown-out Detector Low Power Mode (Write Protect)** 0 = BOD operated in Normal mode (default). 1 = BOD Low Power mode Enabled. **Note1:**The BOD consumes about 100 uA in Normal mode, and the low power mode can reduce the current to about 1/10 but slow the BOD response. **Note2:** This bit is write protected. Refer to the REGWRPROT register.

Bits	Description	
[4]	**BOD_INTF**	**Brown-out Detector Interrupt Flag** 0 = Brown-out Detector does not detect any voltage draft at V_{DD} down through or up through the voltage of BOD_VL setting. 1 = When Brown-out Detector detects the V_{DD} is dropped down through the voltage of BOD_VL setting or the V_{DD} 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. **Note:** Write 1 to clear this bit to 0.
[3]	**BOD_RSTEN**	**Brown-out Reset Enable Control(Write Protect)** 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). 1 = Brown-out "RESET" function Enabled. **Note1:** 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). **Note2:** The default value is set by flash controller user configuration register CBRST (Config0[20]). **Note3:** This bit is write protected. Refer to the REGWRPROT register.
[2:1]	**BOD_VL**	**Brown-out Detector Threshold Voltage Select (Write Protect)** The default value is set by flash controller user configuration register CBOV (Config0[22:21]) bits. 00 = Brown-out voltage is 2.2V. 01 = Brown-out voltage is 2.7V. 10 = Brown-out voltage is 3.7V. 11 = Brown-out voltage is 4.4V. **Note:** This bit is write protected. Refer to the REGWRPROT register.
[0]	**BOD_EN**	**Brown-out Detector Enable Control(Write Protect)** The default value is set by flash controller user configuration register CBODEN (Config0[23]) bit. 0 = Brown-out Detector function Disabled. 1 = Brown-out Detector function Enabled. **Note:** This bit is write protected. Refer to the REGWRPROT register.

Temperature Sensor Control Register (TEMPCR)

Register	Offset	R/W	Description	Reset Value
TEMPCR	GCR_BA+0x1C	R/W	Temperature Sensor Control Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**VTEMP_EN**	**Temperature Sensor Enable Control** This bit is used to enable/disable temperature sensor function. 0 = Temperature sensor function Disabled (default). 1 = Temperature sensor function Enabled. **Note:** After this bit is set to 1, the value of temperature sensor output can be obtained from the ADC conversion result. Please refer to the EADC chapter for detailed ADC conversion functional description.

Power-onReset Control Register (PORCR)

Register	Offset	R/W	Description	Reset Value
PORCR	GCR_BA+0x24	R/W	Power-onReset 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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
POR_DIS_CODE							

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

P0 Multiple Function and Input Type Control Register (P0_MFP)

Register	Offset	R/W	Description	Reset Value
P0_MFP	GCR_BA+0x30	R/W	P0Multiple Function and Input Type Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P0_TYPE							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**P0_ALT**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P0_MFP							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[m+16] m=0,1..7	**P0_TYPE[m]**	**Port 0 Schmitt Trigger Input Enable Bits** 0 = Port 0 bit m Schmitt trigger input function Disabled. 1 = Port 0 bit m Schmitt trigger input function Enabled.
[15:12]	**Reserved**	Reserved.
[11]	**P0_ALT[3]**	**P0.3 Alternative Function** See P0_MFP[3].
[10]	**P0_ALT[2]**	**P0.2 Alternative Function** See P0_MFP[2].
[9]	**P0_ALT[1]**	**P0.1 Alternative Function** See P0_MFP[1].
[8]	**P0_ALT[0]**	**P0.0 Alternative Function** See P0_MFP[0].
[7]	**P0_MFP[7]**	**P0.7Multi-function Selection** 0 = The GPIO P0.7 is selected. 1 = The STADC function is selected.
[6]	**P0_MFP[6]**	**P0.6Multi-function Selection** 0 = The GPIO P0.6 is selected. 1 = TheEPWM0_BRAKE1 function is selected.
[5]	**P0_MFP[5]**	**P0.5Multi-function Selection** 0 = The GPIO P0.5 is selected. 1 = The EPWM0_CH5 function is selected.
[4]	**P0_MFP[4]**	**P0.4Multi-function Selection** 0 = The GPIO P0.4 is selected. 1 = The EPWM0_CH4function is selected.

Bits	Description	
[3]	**P0_MFP[3]**	**P0.3Multi-function Selection** Bits P0_ALT[3] and P0_MFP[3] determine the P0.3 function. (P0_ALT[3], P0_MFP[3]) value and function mapping is as following list. (0, 0) = The GPIO P0.3 function is selected. (0, 1) = The EPWM0_CH3 function is selected. (1, 1) = The STADC function selected.
[2]	**P0_MFP[2]**	**P0.2Multi-function Selection** Bits P0_ALT[2] and P0_MFP[2] determine the P0.2 function. (P0_ALT[2], P0_MFP[2]) value and function mapping is as following list. (0, 0) = The GPIO P0.2 function is selected. (0, 1) = The EPWM0_CH2 function is selected. (1, x) = Reserved.
[1]	**P0_MFP[1]**	**P0.1Multi-function Selection** Bits P0_ALT[1] and P0_MFP[1] determine the P0.1 function. (P0_ALT[2], P0_MFP[2]) value and function mapping is as following list. (0, 0) = The GPIO P0.1 function is selected. (0, 1) = The EPWM0_CH1 function is selected. (1, x) = Reserved.
[0]	**P0_MFP[0]**	**P0.0Multi-function Selection** Bits P0_ALT[0] and P0_MFP[0] determine the P0.0 function. (P0_ALT[0], P0_MFP[0]) value and function mapping is as following list. (0, 0) = The GPIO P0.0 function is selected. (0, 1) = The EPWM0_CH0 function is selected. (1, x) = Reserved.

P1 Multiple Function and Input Type Control Register (P1_MFP)

Register	Offset	R/W	Description	Reset Value
P1_MFP	GCR_BA+0x34	R/W	P1Multiple Function and Input Type Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P1_TYPE							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P1_MFP							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[m+16] m=0,1..7	**P1_TYPE[m]**	**Port 1 Schmitt Trigger Input Enable Bits** 0 = Port 1 bit m Schmitt trigger input function Disabled. 1 = Port 1 bit m Schmitt trigger input function Enabled.
[15:8]	**Reserved**	Reserved.
[7]	**P1_MFP[7]**	**P1.7Multi-function Selection** 0 = The GPIO P1.7 is selected. 1 = The EPWM1_BRAKE0 function is selected.
[6]	**P1_MFP[6]**	**P1.6Multi-function Selection** 0 = The GPIO P1.6 is selected. 1 = TheEPWM0_BRAKE0 function is selected.
[5]	**P1_MFP[5]**	**P1.5Multi-function Selection** 0 = The GPIO P1.5 is selected. 1 = The EPWM1_CH5 function is selected.
[4]	**P1_MFP[4]**	**P1.4Multi-function Selection** 0 = The GPIO P1.4 is selected. 1 = The EPWM1_CH4 function is selected.
[3]	**P1_MFP[3]**	**P1.3Multi-function Selection** 0 = The GPIO P1.3 is selected. 1 = The EPWM1_CH3 function is selected.
[2]	**P1_MFP[2]**	**P1.2Multi-function Selection** 0 = The GPIO P1.2 is selected. 1 = The EPWM1_CH2 function is selected.

Bits	Description	
[1]	**P1_MFP[1]**	**P1.1Multi-function Selection** 0 = The GPIO P1.1 is selected. 1 = The EPWM1_CH1 function is selected.
[0]	**P1_MFP[0]**	**P1.0Multi-function Selection** 0 = The GPIO P1.0 is selected. 1 = The EPWM1_CH0 function is selected.

P2 Multiple Function and Input Type Control Register (P2_MFP)

Register	Offset	R/W	Description	Reset Value
P2_MFP	GCR_BA+0x38	R/W	P2Multiple Function and Input Type Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
P2_TYPE							
15	14	13	12	11	10	9	8
P2_ALT[7:4]				Reserved			P2_ALT[0]
7	6	5	4	3	2	1	0
P2_MFP							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[m+16] m=0,1..7	**P2_TYPE[m]**	**Port 2 Schmitt Trigger Input Enable Bits** 0 = Port 2 bit m Schmitt trigger input function Disabled. 1 = Port 2 bit m Schmitt trigger input function Enabled.
[15]	**P2_ALT[7]**	**P2.7Alternative Function** See P2_MFP[7].
[14]	**P2_ALT[6]**	**P2.6Alternative Function** See P2_MFP[6].
[13]	**P2_ALT[5]**	**P2.5 Alternative Function** See P2_MFP[5].
[12]	**P2_ALT[4]**	**P2.4 Alternative Function** See P2_MFP[4].
[11:9]	**Reserved**	Reserved.
[8]	**P2_ALT[0]**	**P2.0Alternative Function** See P2_MFP[0].
[7]	**P2_MFP[7]**	**P2.7Multi-function Selection** Bits P2_ALT[7] and P2_MFP[7] determine the P2.7 function. (P2_ALT[7], P2_MFP[7]) value and function mapping is as following list. (0, 0) = The GPIO P2.7 function is selected. (1, 0) = The SPI0_CLK function is selected. (1, 1) = The UAT1_nRTS function selected.

Bits	Description	
[6]	**P2_MFP[6]**	**P2.6Multi-function Selection** Bits P2_ALT[6] and P2_MFP[6] determine the P2.6 function. (P2_ALT[6], P2_MFP[6]) value and function mapping is as following list. (0, 0) = The GPIO P2.6 function is selected. (0, 1) = The QEI0_Z function is selected. (NM15xx series only) (0, 1) = Reserved. (1, 0) = The SPI0_SS function selected. (1, 1) = The UART1_nCTS function selected.
[5]	**P2_MFP[5]**	**P2.5Multi-function Selection** Bits P2_ALT[5] and P2_MFP[5] determine the P2.5 function. (P2_ALT[5], P2_MFP[5]) value and function mapping is as following list. (0, 0) = The GPIO P2.5 function is selected. (0, 1) = The QEI0_B function is selected. (NM15xx series only) (0, 1) = Reserved. (1, 1) = The CAN0_TXD function selected. (NM15xx series only) (1, 1) = Reserved.
[4]	**P2_MFP[4]**	**P2.4 Multi-function Selection** Bits P2_ALT[4] and P2_MFP[4] determine the P2.4 function. (P2_ALT[4], P2_MFP[4]) value and function mapping is as following list. (0, 0) = The GPIO P2.4 function is selected. (0, 1) = The QEI0_A function is selected. (NM15xx series only) (0, 1) = Reserved. (1, 1) = The CAN0_RXD function selected. (NM15xx series only) (1, 1) = Reserved.
[3]	**P2_MFP[3]**	**P2.3Multi-function Selection** 0 = The GPIO P2.3 is selected. 1 = The ECAP0_IC0 function is selected.
[2]	**P2_MFP[2]**	**P2.2Multi-function Selection** 0 = The GPIO P2.2 is selected. 1 = The ECAP0_IC1 function is selected.
[1]	**P2_MFP[1]**	**P2.1Multi-function Selection** 0 = The GPIO P2.1 is selected. 1 = TheECAP0_IC2 function is selected.
[0]	**P2_MFP[0]**	**P2.0Multi-function Selection** Bits P2_ALT[0] and P2_MFP[0] determine the P2.0 function. (P2_ALT[0], P2_MFP[0]) value and function mapping is as following list. (0, 0) = The GPIO P2.0 function is selected. (0, 1) = The SPI2_MOSI function is selected. (1, 0) = The ACMP2_O function is selected.

P3 Multiple Function and Input Type Control Register (P3_MFP)

Register	Offset	R/W	Description	Reset Value
P3_MFP	GCR_BA+0x3C	R/W	P3Multiple Function and Input Type Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
P3_TYPE							
15	14	13	12	11	10	9	8
Reserved		P3_ALT[5:4]		Reserved		P3_ALT[1:0]	
7	6	5	4	3	2	1	0
P3_MFP							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[m+16] m=0,1..7	**P3_TYPE[m]**	**Port 3 Schmitt Trigger Input Enable Bits** 0 = Port 3 bit m Schmitt trigger input function Disabled. 1 = Port 3 bit m Schmitt trigger input function Enabled.
[15:14]	**Reserved**	Reserved.
[13]	**P3_ALT[5]**	**P3.5Alternative Function** See P3_MFP[5].
[12]	**P3_ALT[4]**	**P3.4Alternative Function** See P3_MFP[4].
[11:10]	**Reserved**	Reserved.
[9]	**P3_ALT[1]**	**P3.1Alternative Function** See P3_MFP[1].
[8]	**P3_ALT[0]**	**P3.0 Alternative Function** See P3_MFP[0].
[7]	**P3_MFP[7]**	**P3.7Multi-function Selection** 0 = The GPIO P3.7 is selected. 1 = The CAN0_TXD function is selected. Should be 0 for GPIO P3.7.
[6]	**P3_MFP[6]**	**P3.6Multi-function Selection** 0 = The GPIO P3.6 is selected. 1 = The CAN0_RXD function is selected. Should be 0 for GPIO P3.6.

Bits	Description	
[5]	**P3_MFP[5]**	**P3.5Multi-function Selection** Bits P3_ALT[5] and P3_MFP[5] determine the P3.5 function. (P3_ALT[5], P3_MFP[5]) value and function mapping is as following list. (0, 0) = The GPIO P3.5 function is selected. (0, 1) = The TM1 function is selected. (1, 0) = The I2C0_SCL function is selected.
[4]	**P3_MFP[4]**	**P3.4Multi-function Selection** Bits P3_ALT[4] and P3_MFP[4] determine the P3.4 function. (P3_ALT[4], P3_MFP[4]) value and function mapping is as following list. (0, 0) = The GPIO P3.4 function is selected. (0, 1) = The TM0 function is selected. (1, 0) = The I2C0_SDA function is selected.
[3]	**P3_MFP[3]**	**P3.3Multi-function Selection** 0 = The GPIO P3.3 is selected. 1 = The INT1 function is selected.
[2]	**P3_MFP[2]**	**P3.2Multi-function Selection** 0 = The GPIO P3.2 is selected. 1 = The INT0 function is selected.
[1]	**P3_MFP[1]**	**P3.1Multi-function Selection** Bits P3_ALT[1] and P3_MFP[1] determine the P3.1 function. (P3_ALT[1], P3_MFP[1]) value and function mapping is as following list. (0, 0) = The GPIO P3.1 function is selected. (0, 1) = The UART0_TXD function is selected. (1, 0) = The ACMP0_O function is selected.
[0]	**P3_MFP[0]**	**P3.0Multi-function Selection** Bits P3_ALT[0] and P3_MFP[0] determine the P3.0 function. (P3_ALT[0], P3_MFP[0]) value and function mapping is as following list. (0, 0) = The GPIO P3.0 function is selected. (0, 1) = The UART0_RXD function is selected. (1, 0) = The CLKO function is selected.

P4 Multiple Function and Input Type Control Register (P4_MFP)

Register	Offset	R/W	Description	Reset Value
P4_MFP	GCR_BA+0x40	R/W	P4Multiple Function and Input Type Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P4_TYPE							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved	**P4_ALT[6]**	**Reserved**					
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P4_MFP[7:4]				**Reserved**	**P4_MFP[2:0]**		

Bits	Description	
[31:24]	**Reserved**	Reserved.
[m+16] m=0,1..7	**P4_TYPE[m]**	**Port 4 Schmitt Trigger Input Enable Bits** 0 = Port 4 bit m Schmitt trigger input function Disabled. 1 = Port 4 bit m Schmitt trigger input function Enabled.
[15]	**Reserved**	Reserved.
[14]	**P4_ALT[6]**	**P4.6Alternative Function** See P4_MFP[6].
[13:8]	**Reserved**	Reserved.
[7]	**P4_MFP[7]**	**P4.7Multi-function Selection** 0 = The GPIO P4.7 is selected. 1 = The TM3function is selected.
[6]	**P4_MFP[6]**	**P4.6Multi-function Selection** Bits P4_ALT[6] and P4_MFP[6] determine the P4.6 function. (P4_ALT[6], P4_MFP[6]) value and function mapping is as following list. (0, 0) = The GPIO P4.6 function is selected. (0, 1) = The TM2 function is selected. (1, 0) = The QEI1_Z function is selected. (NM15xx series only) (1, 0) = Reserved.
[5]	**P4_MFP[5]**	**P4.5Multi-function Selection** 0 = The GPIO P4.5 is selected. 1 = The QEI1_B function is selected. (NM15xx series only) 1 = Reserved.

Bits	Description	
[4]	**P4_MFP[4]**	**P4.4Multi-function Selection** 0 = The GPIO P4.4 is selected. 1 = The QEI1_A function is selected. (NM15xx series only) 1 = Reserved.
[3]	**Reserved**	Reserved.
[2]	**P4_MFP[2]**	**P4.2Multi-function Selection** 0 = The GPIO P4.2 is selected. 1 = TheECAP1_IC2 function is selected.
[1]	**P4_MFP[1]**	**P4.1Multi-function Selection** 0 = The GPIO P4.1 is selected. 1 = The ECAP1_IC1 function is selected.
[0]	**P4_MFP[0]**	**P4.0Multi-function Selection** 0 = The GPIO P4.0 is selected. 1 = The ECAP1_IC0 function is selected.

P5 Multiple Function and Input Type Control Register (P5_MFP)

Register	Offset	R/W	Description	Reset Value
P5_MFP	GCR_BA+0x44	R/W	P5Multiple Function and Input Type Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
P5_TYPE							
15	14	13	12	11	10	9	8
Reserved					P5_ALT		
7	6	5	4	3	2	1	0
P5_MFP							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[m+16] m=0,1..7	**P5_TYPE[m]**	**Port 5 Schmitt Trigger Input Enable Bits** 0 = Port 5 bit m Schmitt trigger input function Disabled. 1 = Port 5 bit m Schmitt trigger input function Enabled.
[15:11]	**Reserved**	Reserved.
[10]	**P5_ALT[2]**	**P5.2Alternative Function** See P5_MFP[2].
[9]	**P5_ALT[1]**	**P5.1Alternative Function** See P5_MFP[1].
[8]	**P5_ALT[0]**	**P5.0Alternative Function** See P5_MFP[0].
[7]	**P5_MFP[7]**	**P5.7Multi-function Selection** 0 = The GPIO P5.7 is selected. 1 = The BPWM0_CH1 function is selected.
[6]	**P5_MFP[6]**	**P5.6Multi-function Selection** 0 = The GPIO P5.6 is selected. 1 = TheBPWM0_CH0 function is selected.
[5]	**P5_MFP[5]**	**P5.5Multi-function Selection** 0 = The GPIO P5.5 is selected. 1 = The CLKO function is selected.
[4]	**P5_MFP[4]**	**P5.4Multi-function Selection** 0 = The GPIO P5.4 is selected. 1 = The SPI2_SS function is selected.

Bits	Description	
[3]	**P5_MFP[3]**	**P5.3Multi-function Selection** 0 = The GPIO P5.3 is selected. 1 = The SPI2_CLK function is selected.
[2]	**P5_MFP[2]**	**P5.2Multi-function Selection** Bits P5_ALT[2] and P5_MFP[2] determine the P5.2 function. (P5_ALT[2], P5_MFP[2]) value and function mapping is as following list. (0, 0) = The GPIO P5.2 function is selected. (0, 1) = The SPI2_MISO function is selected. (1, 0) = The ACMP1_O function is selected.
[1]	**P5_MFP[1]**	**P5.1Multi-function Selection** Bits P5_ALT[1] and P5_MFP[1] determine the P5.1 function. (P5_ALT[1], P5_MFP[1]) value and function mapping is as following list. (0, 0) = The GPIO P5.1 function is selected. (0, 1) = The SPI0_MISO function is selected. (1, 0) = The UART0_nCTSfunction is selected. (1, 1) = The I2C0_SDA function is selected.
[0]	**P5_MFP[0]**	**P5.0Multi-function Selection** This bit combined with P5_ALT[0] selects P5.0 multi-function. Bits P5_ALT[0] and P5_MFP[0] determine the P5.0 function. (P5_ALT[0], P5_MFP[0]) value and function mapping is as following list. (0, 0) = The GPIO P5.0 function is selected. (0, 1) = The SPI0_MOSI function is selected. (1, 0) = The UART0_nRTS function is selected. (1, 1) = The I2C0_SCL function is selected.

P6 Multiple Function and Input Type Control Register (P6_MFP)

Register	Offset	R/W	Description	Reset Value
P6_MFP	GCR_BA+0x48	R/W	P6Multiple Function and Input Type Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
P6_TYPE							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
P6_MFP							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[m+16] m=0,1..7	**P6_TYPE[m]**	**Port 6 Schmitt Trigger Input Enable Bits** 0 = Port 6 bit m Schmitt trigger input function Disabled. 1 = Port 6 bit m Schmitt trigger input function Enabled.
[15:8]	**Reserved**	Reserved.
[7]	**P6_MFP[7]**	**P6.7Multi-function Selection** 0 = The GPIO P6.7 is selected. 1 = The EADC0_CH7 function is selected.
[6]	**P6_MFP[6]**	**P6.6Multi-function Selection** 0 = The GPIO P6.6 is selected. 1 = The EADC0_CH6 function is selected.
[5]	**P6_MFP[5]**	**P6.5Multi-function Selection** 0 = The GPIO P6.5 is selected. 1 = The EADC0_CH5 or ACMP1_P function is selected.
[4]	**P6_MFP[4]**	**P6.4Multi-function Selection** 0 = The GPIO P6.4 is selected. 1 = The EADC0_CH4 or ACMP1_N function is selected.
[3]	**P6_MFP[3]**	**P6.3Multi-function Selection** 0 = The GPIO P6.3 is selected. 1 = The EADC0_CH3 function is selected.
[2]	**P6_MFP[2]**	**P6.2Multi-function Selection** 0 = The GPIO P6.2 is selected. 1 = The EADC0_CH2 function is selected.

Bits	Description	
[1]	**P6_MFP[1]**	**P6.1Multi-function Selection** 0 = The GPIO P6.1 is selected. 1 = The EADC0_CH1 function is selected.
[0]	**P6_MFP[0]**	**P6.0Multi-function Selection** 0 = The GPIO P6.0 is selected. 1 = The EADC0_CH0 function is selected.

P7 Multiple Function and Input TypeControl Register (P7_MFP)

Register	Offset	R/W	Description	Reset Value
P7_MFP	GCR_BA+0x4C	R/W	P7Multiple Function and Input Type Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
P7_TYPE							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
P7_MFP							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[m+16] m=0,1..7	**P7_TYPE[m]**	**Port 7 Schmitt Trigger Input Enable Bits** 0 = Port 7 bit m Schmitt trigger input function Disabled. 1 = Port 7 bit m Schmitt trigger input function Enabled.
[15:8]	**Reserved**	Reserved.
[7]	**P7_MFP[7]**	**P7.7Multi-function Selection** 0 = The GPIO P7.7 is selected. 1 = The EADC1_CH7function is selected.
[6]	**P7_MFP[6]**	**P7.6Multi-function Selection** 0 = The GPIO P7.6 is selected. 1 = TheEADC1_CH6 function is selected.
[5]	**P7_MFP[5]**	**P7.5Multi-function Selection** 0 = The GPIO P7.5 is selected. 1 = The EADC1_CH5 or ACMP2_P function is selected.
[4]	**P7_MFP[4]**	**P7.4Multi-function Selection** 0 = The GPIO P7.4 is selected. 1 = The EADC1_CH4 or ACMP2_Nfunction is selected.
[3]	**P7_MFP[3]**	**P7.3Multi-function Selection** 0 = The GPIO P7.3 is selected. 1 = The EADC1_CH3 function is selected.
[2]	**P7_MFP[2]**	**P7.2Multi-function Selection** 0 = The GPIO P7.2 is selected. 1 = The EADC1_CH2 function is selected.

Bits	Description	
[1]	**P7_MFP[1]**	**P7.1Multi-function Selection** 0 = The GPIO P7.1 is selected. 1 = The EADC1_CH1 function is selected.
[0]	**P7_MFP[0]**	**P7.0Multi-function Selection** 0 = The GPIO P7.0 is selected. 1 = The EADC1_CH0 function is selected.

P8 Multiple Function and Input Type Control Register (P8_MFP)

Register	Offset	R/W	Description	Reset Value
P8_MFP	GCR_BA+0x50	R/W	P8Multiple Function and Input Type Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
P8_TYPE							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
P8_MFP[7]	Reserved		P8_MFP[4:0]				

Bits	Description	
[31:24]	**Reserved**	Reserved.
[m+16] m=0,1..7	**P8_TYPE[m]**	**Port 8 Schmitt Trigger Input Enable Bits** 0 = Port 8 bit m Schmitt trigger input function Disabled. 1 = Port 8 bit m Schmitt trigger input function Enabled.
[15:8]	**Reserved**	Reserved.
[7]	**P8_MFP[7]**	**P8.7Multi-function Selection** 0 = The GPIO P8.7 is selected. 1 = The ACMP0_O function is selected.
[6:5]	**Reserved**	Reserved.
[4]	**P8_MFP[4]**	**P8.4Multi-function Selection** 0 = The GPIO P8.4 is selected. 1 = The ACMP0_P function is selected.
[3]	**P8_MFP[3]**	**P8.3Multi-function Selection** 0 = The GPIO P8.3 is selected. 1 = The ACMP0_N function is selected.
[2]	**P8_MFP[2]**	**P8.2Multi-function Selection** 0 = The GPIO P8.2 is selected. 1 = The OP0_O function is selected.
[1]	**P8_MFP[1]**	**P8.1Multi-function Selection** 0 = The GPIO P8.1 is selected. 1 = The OP0_N function is selected.
[0]	**P8_MFP[0]**	**P8.0Multi-function Selection** 0 = The GPIO P8.0 is selected. 1 = The OP0_P function is selected.

P9 Multiple Function and Input TypeControl Register (P9_MFP)

Register	Offset	R/W	Description	Reset Value
P9_MFP	GCR_BA+0x54	R/W	P9Multiple Function and Input Type Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
P9_TYPE							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
P9_MFP							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[m+16] m=0,1..7	**P9_TYPE[m]**	**Port 9 Schmitt Trigger Input Enable Bits** 0 = Port 9 bit m Schmitt trigger input function Disabled. 1 = Port 9 bit m Schmitt trigger input function Enabled.
[15:8]	**Reserved**	Reserved.
[7]	**P9_MFP[7]**	**P9.7Multi-function Selection** 0 = The GPIO P9.7 is selected. 1 = The SPI1_Ssfunction is selected.
[6]	**P9_MFP[6]**	**P9.6Multi-function Selection** 0 = The GPIO P9.6 is selected. 1 = TheSPI1_MOSI function is selected.
[5]	**P9_MFP[5]**	**P9.5Multi-function Selection** 0 = The GPIO P9.5 is selected. 1 = The SPI1_MISO function is selected.
[4]	**P9_MFP[4]**	**P9.4Multi-function Selection** 0 = The GPIO P9.4 is selected. 1 = The SPI1_CLKfunction is selected.
[3]	**P9_MFP[3]**	**P9.3Multi-function Selection** 0 = The GPIO P9.3 is selected. 1 = The EPWM1_BRAKE1function is selected.
[2]	**P9_MFP[2]**	**P9.2Multi-function Selection** 0 = The GPIO P9.2 is selected. 1 = The OP1_P function is selected.

Bits	Description	
[1]	**P9_MFP[1]**	**P9.1Multi-function Selection** 0 = The GPIO P9.1 is selected. 1 = The OP1_N function is selected.
[0]	**P9_MFP[0]**	**P9.0Multi-function Selection** 0 = The GPIO P9.0 is selected. 1 = The OP1_O function is selected.

PA Multiple Function and Input TypeControl Register (PA_MFP)

Register	Offset	R/W	Description	Reset Value
PA_MFP	GCR_BA+0x58	R/W	PAMultiple Function and Input Type Control Register	0x0000_0000

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

Bits	Description	
[31:18]	**Reserved**	Reserved.
[m+16] m=0,1	**PA_TYPE[m]**	**Port a Schmitt Trigger Input Enable Bits** 0 = Port A bit m Schmitt trigger input function Disabled. 1 = Port A bit m Schmitt trigger input function Enabled.
[15:10]	**Reserved**	Reserved.
[9]	**PA_ALT[1]**	**PA.1 Alternative Function** See PA_MFP[1].
[8]	**PA_ALT[0]**	**PA.0 Alternative Function** See PA_MFP[0].
[7:2]	**Reserved**	Reserved.
[1]	**PA_MFP[1]**	**PA.1Multi-function Selection** Bits PA_ALT[1] and PA_MFP[1] determine the PA.1 function. (PA_ALT[1], PA_MFP[1]) value and function mapping is as following list. (0, 0) = The GPIO PA.1 function is selected. (0, 1) = The UART1_RXD function is selected. (1, 1) = The I2C0_SCL function is selected.
[0]	**PA_MFP[0]**	**PA.0Multi-function Selection** Bits PA_ALT[0] and PA_MFP[0] determine the PA.0 function. (PA_ALT[0], PA_MFP[0]) value and function mapping is as following list. (0, 0) = The GPIO PA.0 function is selected. (0, 1) = The UART1_TXD function is selected. (1, 1) = The I2C0_SDA function is selected.

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, “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.

Writing this register to disable/enable register protection, and reading it to get 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 have to disable the protected function by writing the sequence value “59h”, “16h”, “88h” to this field. After this sequence is completed, the REGPROTDIS bit will be set to 1 and write-protection registers can be normal write.
[0]	**REGPROTDIS**	**Register Write-protection Disable Index (Read Only)** 0 = Write-protection Enabled for writing protected registers. Any write to the protectedregister is ignored. 1 = Write-protection Disabled for writing protected registers. **Note:**This bit iswrite protected bit. Refer to the REGWRPROT register.

6.2.7 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 0, and reload (wrap) to the value in the SysTick Reload Value Register (SYST_RVR) on the next clock cycle, then decrement on subsequent clocks. When the counter transitions to 0, 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 0 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 0, the timer will be maintained with a current value of 0 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”.

6.2.7.1 System Timer Control Register Map

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

Register	Offset	R/W	Description	Reset Value
SYST Base Address: **SYST_BA = 0xE000_E010**				
SYST_CSR	SYST_BA+0x00	R/W	SysTick Control and Status Register	0x0000_0000
SYST_RVR	SYST_BA+0x04	R/W	SysTick Reload Value Register	0xXXXX_XXXX
SYST_CVR	SYST_BA+0x08	R/W	SysTick Current Value Register	0xXXXX_XXXX

6.2.7.2 System Timer Control Register Description

SysTick Control and Status (SYST_CSR)

Register	Offset	R/W	Description	Reset Value
SYST_CSR	SYST_BA+0x00	R/W	SysTick Control and Status Register	0x0000_0000

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

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**COUNTFLAG**	**System Tick Counter Flag** Returns 1 if timer counted to 0 since last time this register was read. COUNTFLAG is set by a count transition from 1 to 0. COUNTFLAG is cleared on read or by a write to the Current Value register (SYST_CVR).
[15:3]	**Reserved**	Reserved.
[2]	**CLKSRC**	**System Tick Clock Source Selection** 0 = Clock source is optional, refer to STCLK_S (CLKSEL0[5:3]). 1 = Core clock used for SysTick timer.
[1]	**TICKINT**	**System Tick Interrupt Enabled** 0 = Counting down to 0 does not cause the SysTick exception to be pended. Software can use COUNTFLAG to determine if a count to 0 has occurred. 1 = Counting down to 0 will cause the SysTick exception to be pended. Clearing the SysTick Current Value register by a write in software will not cause SysTick to be pended.
[0]	**ENABLE**	**System Tick Counter Enabled** 0 = SysTick counter Disabled. 1 = SysTick counter Enabled.

SysTick Reload Value Register （SYST_RVR）

Register	Offset	R/W	Description	Reset Value
SYST_RVR	SYST_BA+0x04	R/W	SysTick Reload Value Register	0xXXXX_XXXX

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

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

SysTick Current Value Register （SYST_CVR）

Register	Offset	R/W	Description	Reset Value
SYST_CVR	SYST_BA+0x08	R/W	SysTick Current Value Register	0xXXXX_XXXX

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

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

6.2.8 Nested Vectored Interrupt Controller (NVIC)

The Cortex®-M0 provides an interrupt controller as an integral part of the exception mode, named as “Nested Vectored Interrupt Controller (NVIC)”, which is closely coupled to the processor core and provides following features:

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

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

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

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

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

6.2.8.1 *Exception Model and System Interrupt Map*

Table 6-2 lists the exception model supported by the NuMicro®NM1530Series. 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 6-2 Exception Model

Exception Number	Vector Address	Interrupt Number (Bit In Interrupt Registers)	Interrupt Name	Source Module	ExceptionDescription	Power Down Wake-Up
1 ~ 15		-	**-**	-	System exceptions	-
16	0x40	0	**BOD_INT**	Brown-out	Brown-out low voltage detectedinterrupt	Yes
17	0x44	1	**WDT_INT**	WDT	Watchdog Timer interrupt	Yes
18	0x48	2	**EINT0_INT**	P3.2	External signal interrupt from P3.2 pin	Yes
19	0x4C	3	**EINT1_INT**	P3.3	External signal interrupt from P3.3 pin	Yes
20	0x50	4	**GPG0_INT**	P0~P4 except P3.2 and P3.3	External interrupt from GPIO group 0 (P0~P4) except P3.2 and P3.3	Yes
21	0x54	5	**GPG1_INT**	P5~PA	External interrupt from GPIO group 1 (P5~PA)	Yes
22	0x58	6	**BPWM0_INT**	BPWM0	Basic PWM0interrupt	No
23	0x5C	7	**EADC0_INT**	EADC0	EADC0 interrupt	No
24	0x60	8	**TMR0_INT**	TMR0	Timer 0 interrupt	No
25	0x64	9	**TMR1_INT**	TMR1	Timer 1 interrupt	No
26	0x68	10	**TMR2_INT**	TMR2	Timer 2 interrupt	No
27	0x6C	11	**TMR3_INT**	TMR3	Timer 3 interrupt	No
28	0x70	12	**UART0_INT**	UART0	UART0interrupt	Yes

Exception Number	Vector Address	Interrupt Number (Bit In Interrupt Registers)	Interrupt Name	Source Module	ExceptionDescription	Power Down Wake-Up
29	0x74	13	**UART1_INT**	UART1	UART1interrupt	Yes
30	0x78	14	**SPI0_INT**	SPI0	SPI0 interrupt	No
31	0x7C	15	**SPI1_INT**	SPI1	SPI1 interrupt	No
32	0x80	16	**SPI2_INT**	SPI2	SPI2 interrupt	No
33	0x84	17	**MDU_INT**	MDU	Motor dive unit interrupt	No
33	0x84	17	**Reserved**	Reserved	Reserved	No
34	0x88	18	**I2C0_INT**	I^2C0	I^2C0 interrupt	Yes
35	0x8C	19	**CKD_INT**	CKD	CKD interrupt	No
36	0x90	20	**CAN_INT**	CAN	CAN interrupt	No
36	0x90	20	**Reserved**	Reserved	Reserved	-
37	0x94	21	**EPWM0_INT**	EPWM0	Enhanced PWM0 interrupt	No
38	0x98	22	**EPWM1_INT**	EPWM1	Enhanced PWM1 interrupt	No
39	0x9C	23	**ECAP0_INT**	ECAP0	Enhanced input capture 0 interrupt	No
40	0xA0	24	**ECAP1_INT**	ECAP1	Enhanced input capture 1 interrupt	No
41	0xA4	25	**ACMP_INT**	ACMP	Analog Comparator0or 1, or OP Amplifier digital output interrupt	Yes (only by analog comparator)
42	0xA8	26	**QEI0_INT**	QEI0	QEI0 interrupt	No
43	0xAC	27	**QEI1_INT**	QEI1	QEI1 interrupt	No
42	0xA8	26	**Reserved**	Reserved	Reserved	-
43	0xAC	27	**Reserved**	Reserved	Reserved	-
44	0xB0	28	**PWRWU_INT**	CLKC	Clock controller interrupt for chip wake up from power-down state	-
45	0xB4	29	**EADC1_INT**	EADC1	EADC1 interrupt	No
46	0xB8	30	**EADC2_INT**	EADC2	EADC2 interrupt	No
47	0xBC	31	**EADC3_INT**	EADC3	EADC3 interrupt	No

Table 6-3System Interrupt Map Vector Table

6.2.8.2 Vector Table

When an interrupt is accepted, the processor will automatically fetch the starting address of the interrupt service routine (ISR) from a vector table in memory. For ARMv6-M, the vector table 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(Bytes)	Description
0	SP_main – The Main stack pointer
Vector Number	Exception Entry Pointer using that Vector Number

Table 6-4 Vector Table

6.2.8.3 Operation Description

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

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

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

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

6.2.8.4 NVIC Control Registers

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

Register	Offset	R/W	Description	Reset Value
NVIC Base Address: **NVIC_BA = 0xE000_E100**				
NVIC_ISER	NVIC_BA+0x000	R/W	IRQ0 ~ IRQ31 Set-Enable Control Register	0x0000_0000
NVIC_ICER	NVIC_BA+0x080	R/W	IRQ0 ~ IRQ31 Clear-Enable Control Register	0x0000_0000
NVIC_ISPR	NVIC_BA+0x100	R/W	IRQ0 ~ IRQ31 Set-Pending Control Register	0x0000_0000
NVIC_ICPR	NVIC_BA+0x180	R/W	IRQ0 ~ IRQ31 Clear-Pending Control Register	0x0000_0000
NVIC_IPR0	NVIC_BA+0x300	R/W	IRQ0 ~ IRQ3 Interrupt Priority Control Register	0x0000_0000
NVIC_IPR1	NVIC_BA+0x304	R/W	IRQ4 ~ IRQ7 Interrupt Priority Control Register	0x0000_0000
NVIC_IPR2	NVIC_BA+0x308	R/W	IRQ8 ~ IRQ11 Interrupt Priority Control Register	0x0000_0000
NVIC_IPR3	NVIC_BA+0x30C	R/W	IRQ12 ~ IRQ15 Interrupt Priority Control Register	0x0000_0000
NVIC_IPR4	NVIC_BA+0x310	R/W	IRQ16 ~ IRQ19 Interrupt Priority Control Register	0x0000_0000
NVIC_IPR5	NVIC_BA+0x314	R/W	IRQ20 ~ IRQ23 Interrupt Priority Control Register	0x0000_0000
NVIC_IPR6	NVIC_BA+0x318	R/W	IRQ24 ~ IRQ27 Interrupt Priority Control Register	0x0000_0000
NVIC_IPR7	NVIC_BA+0x31C	R/W	IRQ28 ~ IRQ31 Interrupt Priority Control Register	0x0000_0000

IRQ0 ~ IRQ31 Set-Enable Control Register （NVIC_ISER）

Register	Offset	R/W	Description	Reset Value
NVIC_ISER	NVIC_BA+0x000	R/W	IRQ0 ~ IRQ31 Set-Enable Control Register	0x0000_0000

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

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

IRQ0 ~ IRQ31 Clear-Enable Control Register (NVIC_ICER)

Register	Offset	R/W	Description	Reset Value
NVIC_ICER	NVIC_BA+0x080	R/W	IRQ0 ~ IRQ31 Clear-Enable Control Register	0x0000_0000

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

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

IRQ0 ~ IRQ31 Set-Pending Control Register （NVIC_ISPR）

Register	Offset	R/W	Description	Reset Value
NVIC_ISPR	NVIC_BA+0x100	R/W	IRQ0 ~ IRQ31 Set-Pending Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**SETPEND**	**Interrupt Set-pending** The ISPR forces interrupts into the pending state, and shows the interrupts that arepending. Each bit represents an IRQ number from IRQ0 ~ IRQ31 (Vector number from 16 ~ 47). Write Operation: 0 = No effect. 1 = Write 1 to set pending state. Read Operation: 0 = Associated interrupt in not in pending status. 1 = Associated interrupt is in pending status. Read value indicates the current pending status.

IRQ0 ~ IRQ31 Clear-Pending Control Register（NVIC_ICPR）

Register	Offset	R/W	Description	Reset Value
NVIC_ICPR	NVIC_BA+0x180	R/W	IRQ0 ~ IRQ31 Clear-Pending Control Register	0x0000_0000

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

Bits	Description	
[31:0]	**CLRPEND**	**Interrupt Clear-pending** The ICPR removes the pending state from interrupts, and shows the interrupts that arepending. Each bit represents an IRQ number from IRQ0 ~ IRQ31 (Vectornumber from 16 ~ 47). Write Operation: 0 = No effect. 1 = Write 1 to clear pending state. Read Operation: 0 = Associated interrupt in not in pending status. 1 = Associated interrupt is in pending status. Read value indicates the current pending status.

IRQ0 ~ IRQ3 Interrupt Priority Register (NVIC_IPR0)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR0	NVIC_BA+0x300	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 lowest priority.
[29:24]	**Reserved**	Reserved.
[23:22]	**PRI_2**	**Priority of IRQ2** "0" denotes the highest priority and "3" denotes lowest priority.
[21:16]	**Reserved**	Reserved.
[15:14]	**PRI_1**	**Priority of IRQ1** "0" denotes the highest priority and "3" denotes lowest priority.
[13:8]	**Reserved**	Reserved.
[7:6]	**PRI_0**	**Priority of IRQ0** "0" denotes the highest priority and "3" denotes lowest priority.
[5:0]	**Reserved**	Reserved.

IRQ4 ~ IRQ7 Interrupt Priority Register (NVIC_IPR1)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR1	NVIC_BA+0x304	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 lowest priority.
[29:24]	**Reserved**	Reserved.
[23:22]	**PRI_6**	**Priority of IRQ6** "0" denotes the highest priority and "3" denotes lowest priority.
[21:16]	**Reserved**	Reserved.
[15:14]	**PRI_5**	**Priority of IRQ5** "0" denotes the highest priority and "3" denotes lowest priority.
[13:8]	**Reserved**	Reserved.
[7:6]	**PRI_4**	**Priority of IRQ4** "0" denotes the highest priority and "3" denotes lowest priority.
[5:0]	**Reserved**	Reserved.

IRQ8 ~ IRQ11 Interrupt Priority Register (NVIC_IPR2)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR2	NVIC_BA+0x308	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 lowest priority.
[29:24]	**Reserved**	Reserved.
[23:22]	**PRI_10**	**Priority of IRQ10** "0" denotes the highest priority and "3" denotes lowest priority.
[21:16]	**Reserved**	Reserved.
[15:14]	**PRI_9**	**Priority of IRQ9** "0" denotes the highest priority and "3" denotes lowest priority.
[13:8]	**Reserved**	Reserved.
[7:6]	**PRI_8**	**Priority of IRQ8** "0" denotes the highest priority and "3" denotes lowest priority.
[5:0]	**Reserved**	Reserved.

IRQ12 ~ IRQ15 Interrupt Priority Register (NVIC_IPR3)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR3	NVIC_BA+0x30C	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 lowest priority.
[29:24]	**Reserved**	Reserved.
[23:22]	**PRI_14**	**Priority of IRQ14** "0" denotes the highest priority and "3" denotes lowest priority.
[21:16]	**Reserved**	Reserved.
[15:14]	**PRI_13**	**Priority of IRQ13** "0" denotes the highest priority and "3" denotes lowest priority.
[13:8]	**Reserved**	Reserved.
[7:6]	**PRI_12**	**Priority of IRQ12** "0" denotes the highest priority and "3" denotes lowest priority.
[5:0]	**Reserved**	Reserved.

IRQ16 ~ IRQ19 Interrupt Priority Register (NVIC_IPR4)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR4	NVIC_BA+0x310	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 lowest priority.
[29:24]	**Reserved**	Reserved.
[23:22]	**PRI_18**	**Priority of IRQ18** "0" denotes the highest priority and "3" denotes lowest priority.
[21:16]	**Reserved**	Reserved.
[15:14]	**PRI_17**	**Priority of IRQ17** "0" denotes the highest priority and "3" denotes lowest priority.
[13:8]	**Reserved**	Reserved.
[7:6]	**PRI_16**	**Priority of IRQ16** "0" denotes the highest priority and "3" denotes lowest priority.
[5:0]	**Reserved**	Reserved.

IRQ20 ~ IRQ23 Interrupt Priority Register (NVIC_IPR5)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR5	NVIC_BA+0x314	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 lowest priority.
[29:24]	**Reserved**	Reserved.
[23:22]	**PRI_22**	**Priority of IRQ22** "0" denotes the highest priority and "3" denotes lowest priority.
[21:16]	**Reserved**	Reserved.
[15:14]	**PRI_21**	**Priority of IRQ21** "0" denotes the highest priority and "3" denotes lowest priority.
[13:8]	**Reserved**	Reserved.
[7:6]	**PRI_20**	**Priority of IRQ20** "0" denotes the highest priority and "3" denotes lowest priority.
[5:0]	**Reserved**	Reserved.

IRQ24 ~ IRQ27 Interrupt Priority Register (NVIC_IPR6)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR6	NVIC_BA+0x318	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 lowest priority.
[29:24]	**Reserved**	Reserved.
[23:22]	**PRI_26**	**Priority of IRQ26** "0" denotes the highest priority and "3" denotes lowest priority.
[21:16]	**Reserved**	Reserved.
[15:14]	**PRI_25**	**Priority of IRQ25** "0" denotes the highest priority and "3" denotes lowest priority.
[13:8]	**Reserved**	Reserved.
[7:6]	**PRI_24**	**Priority of IRQ24** "0" denotes the highest priority and "3" denotes lowest priority.
[5:0]	**Reserved**	Reserved.

IRQ28 ~ IRQ31 Interrupt Priority Register (NVIC_IPR7)

Register	Offset	R/W	Description	Reset Value
NVIC_IPR7	NVIC_BA+0x31C	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 lowest priority.
[29:24]	**Reserved**	Reserved.
[23:22]	**PRI_30**	**Priority of IRQ30** "0" denotes the highest priority and "3" denotes lowest priority.
[21:16]	**Reserved**	Reserved.
[15:14]	**PRI_29**	**Priority of IRQ29** "0" denotes the highest priority and "3" denotes lowest priority.
[13:8]	**Reserved**	Reserved.
[7:6]	**PRI_28**	**Priority of IRQ28** "0" denotes the highest priority and "3" denotes lowest priority.
[5:0]	**Reserved**	Reserved.

6.2.8.5 *Interrupt Source Control Registers*

Besides the interrupt control registers associated with the NVIC, NuMicro®NM1530Series also implement some specific control registers to facilitate the interrupt functions, including “interrupt source identification”, ”NMI source selection” and “interrupt test mode”, as described below.

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

Register	Offset	R/W	Description	Reset Value
INT Base Address: **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 (P0-P4) Interrupt Source Identity	0xXXXX_XXXX
IRQ5_SRC	INT_BA+0x14	R	IRQ5 (P5-PA) Interrupt Source Identity	0xXXXX_XXXX
IRQ6_SRC	INT_BA+0x18	R	IRQ6 (BPWM0) Interrupt Source Identity	0xXXXX_XXXX
IRQ7_SRC	INT_BA+0x1C	R	IRQ7 (EADC0) 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 (UART0) Interrupt Source Identity	0xXXXX_XXXX
IRQ13_SRC	INT_BA+0x34	R	IRQ13 (UART1) 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 (MDU) 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 (CKD) interrupt Source Identity	0xXXXX_XXXX
IRQ20_SRC	INT_BA+0x50	R	IRQ20 (CAN) Interrupt Source Identity	0xXXXX_XXXX
IRQ21_SRC	INT_BA+0x54	R	IRQ21 (EPWM0) Interrupt Source Identity	0xXXXX_XXXX
IRQ22_SRC	INT_BA+0x58	R	IRQ22 (EPWM1) Interrupt Source Identity	0xXXXX_XXXX
IRQ23_SRC	INT_BA+0x5C	R	IRQ23 (ECAP0) Interrupt Source Identity	0xXXXX_XXXX

IRQ24_SRC	INT_BA+0x60	R	IRQ24 (ECAP1) Interrupt Source Identity	0xXXXX_XXXX
IRQ25_SRC	INT_BA+0x64	R	IRQ25 (ACMP) Interrupt Source Identity	0xXXXX_XXXX
IRQ26_SRC	INT_BA+0x68	R	IRQ26 (QEI0) Interrupt Source Identity	0xXXXX_XXXX
IRQ27_SRC	INT_BA+0x6C	R	IRQ27 (QEI1) 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 (EADC1) Interrupt Source Identity	0xXXXX_XXXX
IRQ30_SRC	INT_BA+0x78	R	IRQ30 (EADC2) Interrupt Source Identity	0xXXXX_XXXX
IRQ31_SRC	INT_BA+0x7C	R	IRQ31 (EADC3) Interrupt Source Identity	0xXXXX_XXXX
NMI_SEL	INT_BA+0x80	R/W	NMI Interrupt Source Select Control Register	0x0000_0000
MCU_IRQ	INT_BA+0x84	R/W	MCU Interrupt Request Source Register	0x0000_0000
MCU_IRQCR	INT_BA+0x88	R/W	MCU Interrupt Request Control Register	0x0000_0000

IRQ0 (BOD) Interrupt Source Identity (IRQ0_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ0_SRC	INT_BA+0x00	R	IRQ0 (BOD) 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							**BOD_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**BOD_INT**	**IRQ0 Source Identity** 0 = IRQ0 source is not from BOD interrupt (BOD_INT). 1 = IRQ0 source is from BOD interrupt (BOD_INT).

IRQ1 (WDT) Interrupt Source Identity (IRQ1_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ1_SRC	INT_BA+0x04	R	IRQ1 (WDT) 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						**WWDT_INT**	**WDT_INT**

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**WWDT_INT**	**IRQ1 Source Identity** 0 = IRQ1 source is not from window-watchdog interrupt (WWDT _INT). 1 = IRQ1 source is from window-watchdog interrupt (WWDT_INT).
[0]	**WDT_INT**	**IRQ1 Source Identity** 0 = IRQ1 source is not from watchdog interrupt (WDT _INT). 1 = IRQ1 source is from watchdog interrupt (WDT_INT).

IRQ2 (EINT0) Interrupt Source Identity (IRQ2_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ2_SRC	INT_BA+0x08	R	IRQ2 (EINT0) 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							**EINT0**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**EINT0**	**IRQ2 Source Identity** 0 = IRQ2 source is not from external signal interrupt 0 from P3.2 (EINT0). 1 = IRQ2 source is from external signal interrupt 0 from P3.2 (EINT0).

IRQ3 (EINT1) Interrupt Source Identity (IRQ3_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ3_SRC	INT_BA+0x0C	R	IRQ3 (EINT1) 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							**EINT1**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**EINT1**	**IRQ3 Source Identity** 0 = IRQ3 source is not from external signal interrupt 1 from P3.3 (EINT1). 1 = IRQ3 source is from external signal interrupt 1 from P3.3 (EINT1).

IRQ4 (P0-P4) Interrupt Source Identity (IRQ4_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ4_SRC	INT_BA+0x10	R	IRQ4 (P0-P4) 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			**P4_INT**	**P3_INT**	**P2_INT**	**P1_INT**	**P0_INT**

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**P4_INT**	**IRQ4 Source Identity** 0 = IRQ4 source is not from P4 interrupt (P4_INT). 1 = IRQ4 source is from P4 interrupt (P4_INT).
[3]	**P3_INT**	**IRQ4 Source Identity** 0 = IRQ4 source is not from P3 interrupt (P3_INT). 1 = IRQ4 source is from P3 interrupt (P3_INT).
[2]	**P2_INT**	**IRQ4 Source Identity** 0 = IRQ4 source is not from P2 interrupt (P2_INT). 1 = IRQ4 source is from P2 interrupt (P2_INT).
[1]	**P1_INT**	**IRQ4 Source Identity** 0 = IRQ4 source is not from P1 interrupt (P1_INT). 1 = IRQ4 source is from P1 interrupt (P1_INT).
[0]	**P0_INT**	**IRQ4 Source Identity** 0 = IRQ4 source is not from P0 interrupt (P0_INT). 1 = IRQ4 source is from P0 interrupt (P0_INT).

IRQ5 (P5-PA) Interrupt Source Identity (IRQ5_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ5_SRC	INT_BA+0x14	R	IRQ5 (P5-PA) 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		PA_INT	P9_INT	P8_INT	P7_INT	P6_INT	P5_INT

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**PA_INT**	**IRQ5 Source Identity** 0 = IRQ5 source is not from PA interrupt (PA_INT). 1 = IRQ5 source is from PA interrupt (PA_INT).
[4]	**P9_INT**	**IRQ5 Source Identity** 0 = IRQ5 source is not from P9 interrupt (P9_INT). 1 = IRQ5 source is from P9 interrupt (P9_INT).
[3]	**P8_INT**	**IRQ5 Source Identity** 0 = IRQ5 source is not from P8 interrupt (P8_INT). 1 = IRQ5 source is from P8 interrupt (P8_INT).
[2]	**P7_INT**	**IRQ5 Source Identity** 0 = IRQ5 source is not from P7 interrupt (P7_INT). 1 = IRQ5 source is from P7 interrupt (P7_INT).
[1]	**P6_INT**	**IRQ5 Source Identity** 0 = IRQ5 source is not from P6 interrupt (P6_INT). 1 = IRQ5 source is from P6 interrupt (P6_INT).
[0]	**P5_INT**	**IRQ5 Source Identity** 0 = IRQ5 source is not from P5 interrupt (P5_INT). 1 = IRQ5 source is from P5 interrupt (P5_INT).

IRQ6 (BPWM0) Interrupt Source Identity (IRQ6_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ6_SRC	INT_BA+0x18	R	IRQ6 (BPWM0) 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						**BPCH1_INT**	**BPCH0_INT**

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**BPCH1_INT**	**IRQ6 Source Identity** 0 = IRQ6 source is not from BPWM0 channel 1interrupt (BPCH1_INT). 1 = IRQ6 source is from BPWM0 channel 1 interrupt (BPCH1_INT).
[0]	**BPCH0_INT**	**IRQ6 Source Identity** 0 = IRQ6 source is not from BPWM0 channel 0interrupt (BPCH0_INT). 1 = IRQ6 source is from BPWM0channel 0 interrupt (BPCH0_INT).

IRQ7 (EADC0) Interrupt Source Identity (IRQ7_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ7_SRC	INT_BA+0x1C	R	IRQ7 (EADC0) 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							**EADC0_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**EADC0_INT**	**IRQ7 Source Identity** 0 = IRQ7 source is not from EADC0 interrupt (EADC0_INT). 1 = IRQ7 source is from EADC0 interrupt (EADC0_INT).

IRQ8 (TMR0) Interrupt Source Identity (IRQ8_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ8_SRC	INT_BA+0x20	R	IRQ8 (TMR0) 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							**TMR0_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**TMR0_INT**	**IRQ8 Source Identity** 0 = IRQ8 source is not from Timer0 interrupt (TMR0_INT). 1 = IRQ8 source is from Timer0 interrupt (TMR0_INT).

IRQ9 (TMR1) Interrupt Source Identity (IRQ9_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ9_SRC	INT_BA+0x24	R	IRQ9 (TMR1) 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							**TMR1_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**TMR1_INT**	**IRQ9 Source Identity** 0 = IRQ9 source is not from Timer1 interrupt (TMR1_INT). 1 = IRQ9 source is from Timer1 interrupt (TMR1_INT).

IRQ10 (TMR2) Interrupt Source Identity (IRQ10_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ10_SRC	INT_BA+0x28	R	IRQ10 (TMR2) 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							**TMR2_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**TMR2_INT**	**IRQ10 Source Identity** 0 = IRQ10 source is not from Timer2 interrupt (TMR2_INT). 1 = IRQ10 source is from Timer2 interrupt (TMR2_INT).

IRQ11 (TMR3) Interrupt Source Identity (IRQ11_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ11_SRC	INT_BA+0x2C	R	IRQ11 (TMR3) 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							**TMR3_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**TMR3_INT**	**IRQ11 Source Identity** 0 = IRQ11 source is not from Timer3 interrupt (TMR3_INT). 1 = IRQ11 source is from Timer3 interrupt (TMR3_INT).

IRQ12 (UART0) Interrupt Source Identity (IRQ12_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ12_SRC	INT_BA+0x30	R	IRQ12 (UART0) 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							**UART0_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**UART0_INT**	**IRQ12 Source Identity** 0 = IRQ12 source is not from UART0 interrupt (UART0_INT). 1 = IRQ12 source is from UART0 interrupt (UART0_INT).

IRQ13 (UART1) Interrupt Source Identity (IRQ13_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ13_SRC	INT_BA+0x34	R	IRQ13 (UART1) 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							**UART1_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**UART1_INT**	**IRQ13 Source Identity** 0 = IRQ13 source is not from UART1 interrupt (UART1_INT). 1 = IRQ13 source is from UART1 interrupt (UART1_INT).

IRQ14 (SPI0) Interrupt Source Identity (IRQ14_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ14_SRC	INT_BA+0x38	R	IRQ14 (SPI0) 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							**SPI0_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**SPI0_INT**	**IRQ14 Source Identity** 0 = IRQ14 source is not from SPI0 interrupt (SPI0_INT). 1 = IRQ14 source is from SPI0 interrupt (SPI0_INT).

IRQ15 (SPI1) Interrupt Source Identity (IRQ15_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ15_SRC	INT_BA+0x3C	R	IRQ15 (SPI1) 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							**SPI1_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**SPI1_INT**	**IRQ15 Source Identity** 0 = IRQ15 source is not from SPI1 interrupt (SPI1_INT). 1 = IRQ15 source is from SPI1 interrupt (SPI1_INT).

IRQ16 (SPI2) Interrupt Source Identity (IRQ16_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ16_SRC	INT_BA+0x40	R	IRQ16 (SPI2) 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							**SPI2_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**SPI2_INT**	**IRQ16 Source Identity** 0 = IRQ16 source is not from SPI2 interrupt (SPI2_INT). 1 = IRQ16 source is from SPI2 interrupt (SPI2_INT).

IRQ17 (MDU) Interrupt Source Identity (IRQ17_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ17_SRC	INT_BA+0x44	R	IRQ17 (MDU) 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							**MDU_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**MDU_INT**	**IRQ17 Source Identity** 0 = IRQ17 source is not from MDU interrupt (MDU_INT). 1 = IRQ17 source is from MDU interrupt (MDU_INT).

IRQ18 (I2C) Interrupt Source Identity (IRQ18_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ18_SRC	INT_BA+0x48	R	IRQ18 (I2C0) 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							**I2C0_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**I2C0_INT**	**IRQ18 Source Identity** 0 = IRQ18 source is not from I^2C0 interrupt (I2C0_INT). 1 = IRQ18 source is from I^2C0interrupt (I2C0_INT).

IRQ19 (CKD) Interrupt Source Identity (IRQ19_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ19_SRC	INT_BA+0x4C	R	IRQ19 (CKD) 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							**CKD_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**CKD_INT**	**IRQ19 Source Identity** 0 = IRQ19 source is not from CKD interrupt (CKD_INT). 1 = IRQ19 source is from CKD interrupt (CKD_INT).

IRQ20 (CAN) Interrupt Source Identity (IRQ20_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ20_SRC	INT_BA+0x50	R	IRQ20 (CAN) 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							**CAN_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**CAN_INT**	**IRQ20 Source Identity** 0 = IRQ20 source is not from CAN interrupt (MDU_INT). 1 = IRQ20 source is from CAN interrupt (MDU_INT).

IRQ21 (EPWM0) Interrupt Source Identity (IRQ21_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ21_SRC	INT_BA+0x54	R	IRQ21 (EPWM0) 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							**EPWM0_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**EPWM0_INT**	**IRQ21 Source Identity** 0 = IRQ21 source is not from EPWM0 interrupt (EPWM0_INT). 1 = IRQ21 source is from EPWM0 interrupt (EPWM0_INT).

IRQ22 (EPWM1) Interrupt Source Identity (IRQ22_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ22_SRC	INT_BA+0x58	R	IRQ22 (EPWM1) 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							**EPWM1_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**EPWM1_INT**	**IRQ22 Source Identity** 0 = IRQ22 source is not from EPWM1 interrupt (EPWM1_INT). 1 = IRQ22 source is from EPWM1 interrupt (EPWM1_INT).

IRQ23 (ECAP0) Interrupt Source Identity (IRQ23_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ23_SRC	INT_BA+0x5C	R	IRQ23 (ECAP0) 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							**ECAP0_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**ECAP0_INT**	**IRQ23 Source Identity** 0 = IRQ23 source is not from ECAP0 interrupt (ECAP0_INT). 1 = IRQ23 source is from ECAP0 interrupt (ECAP0_INT).

IRQ24 (ECAP1) Interrupt Source Identity (IRQ24_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ24_SRC	INT_BA+0x60	R	IRQ24 (ECAP1) 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							**ECAP1_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**ECAP1_INT**	**IRQ24 Source Identity** 0 = IRQ24 source is not from ECAP1 interrupt (ECAP1_INT). 1 = IRQ24 source is from ECAP1 interrupt (ECAP1_INT).

IRQ25 (ACMP) Interrupt Source Identity (IRQ25_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ25_SRC	INT_BA+0x64	R	IRQ25 (ACMP) 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							**ACMP_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**ACMP_INT**	**IRQ25 Source Identity** 0 = IRQ25 source is not from ACMP interrupt (ACMP_INT). 1 = IRQ25 source is from ACMP interrupt (ACMP_INT).

IRQ26 (QEI0) Interrupt Source Identity (IRQ26_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ26_SRC	INT_BA+0x68	R	IRQ26 (QEI0) 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							**QEI0_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**QEI0_INT**	**IRQ26 Source Identity** 0 = IRQ26 source is not from QEI0 interrupt (QEI0_INT). 1 = IRQ26 source is from QEI0 interrupt (QEI0_INT).

IRQ27 (QEI1) Interrupt Source Identity (IRQ27_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ27_SRC	INT_BA+0x6C	R	IRQ27 (QEI1) 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							**QEI1_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**QEI1_INT**	**IRQ27 Source Identity** 0 = IRQ27 source is not from QEI1 interrupt (QEI1_INT). 1 = IRQ27 source is from QEI1 interrupt (QEI1_INT).

IRQ28 (PWRWU) Interrupt Source Identity (IRQ28_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ28_SRC	INT_BA+0x70	R	IRQ28 (PWRWU) 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							**PWRWU_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**PWRWU_INT**	**IRQ28 Source Identity** 0 = IRQ28 source is not from PWRWU interrupt (PWRWU_INT). 1 = IRQ28 source is from PWREU interrupt (PWRWU_INT).

IRQ29 (EADC1) InterruptSource Identity (IRQ29_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ29_SRC	INT_BA+0x74	R	IRQ29 (EADC1) InterruptSourceIdentity	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							**EADC1_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**EADC1_INT**	**IRQ29 Source Identity** 0 = IRQ29 source is not from EADC1 interrupt (EADC1_INT). 1 = IRQ29 source is from EADC1 interrupt (EADC1_INT).

IRQ30 (EADC2) InterruptSource Identity (IRQ30_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ30_SRC	INT_BA+0x78	R	IRQ30 (EADC2) InterruptSource 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							**EADC2_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**EADC2_INT**	**IRQ30 Source Identity** 0 = IRQ30 source is not from EADC2 interrupt (EADC2_INT). 1 = IRQ30 source is from EADC2 interrupt (EADC2_INT).

IRQ31 (EADC3) Interrupt Source Identity (IRQ31_SRC)

Register	Offset	R/W	Description	Reset Value
IRQ31_SRC	INT_BA+0x7C	R	IRQ31 (EADC3) 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							**EADC3_INT**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**EADC3_INT**	**IRQ31 Source Identity** 0 = IRQ31 source is not from EADC3 interrupt (EADC3_INT). 1 = IRQ31 source is from EADC3 interrupt (EADC3_INT).

NMI Interrupt Source Select Control Register (NMI_SEL)

Register	Offset	R/W	Description	Reset Value
NMI_SEL	INT_BA+0x80	R/W	NMI Interrupt Source 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**				

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**NMI_EN**	**NMI Interrupt Enable(Write Protect)** 0 = IRQ0~31 assigned toNMI Disabled. (NMI still can be software triggered by setting its pending flag.) 1 = IRQ0~31 assigned to NMI Enabled. **Note:**This bit iswrite protected bit. Refer to the REGWRPROT register.
[7:5]	**Reserved**	Reserved.
[4:0]	**NMI_SEL**	**NMI Interrupt Source Selection** The NMI interrupt to Cortex®-M0 can be selected from one of IRQ0~IRQ31 by setting NMI_SEL with IRQ number. The default NMI interrupt is assigned as IRQ0 interrupt if NMI is enabled by setting NMI_SEL[8].

MCU Interrupt Request Source Register (MCU_IRQ)

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

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

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. 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, it means no interrupt is assert. Write 1 to generate an interrupt to Cortex®-M0 NVIC[n]. When the MCU_IRQ[n] is 1, it means an interrupt is assert. Write 1 to clear the interrupt and MCU_IRQ[n].

MCU Interrupt Request Control Register (MCU_IRQCR)

Register	Offset	R/W	Description	Reset Value
MCU_IRQCR	INT_BA+0x88	R/W	MCU Interrupt Request 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							**FAST_IRQ**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**FAST_IRQ**	**Fast IRQ Latency Enable Bit** 0= MCU IRQ latency is fixed at 13 HCLK, MCU will enter IRQ handler after this fixed latency when interrupt happened. 1= MCU IRQ latency will not fixed, MCU will enter IRQ handler as soon as possible when interrupt happened.

6.2.9 System Control Block Register

Cortex®-M0 status and operating mode control are managed by System Control Block Registers including CPUID, Cortex®-M0 interrupt priority and Cortex®-M0power management.

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

Register	Offset	R/W	Description	Reset Value
SCB Base Address: **SCB_BA = 0xE000_ED00**				
CPUID	SCB_BA+0x00	R	CPUID Register	0x410C_C200
ICSR	SCB_BA+0x04	R/W	Interrupt Control State Register	0x0000_0000
AIRCR	SCB_BA+0x0C	R/W	Application Interrupt and Reset Control Register	0xFA05_0000
SCR	SCB_BA+0x10	R/W	System Control Register	0x0000_0000
SHPR2	SCB_BA+0x1C	R/W	System Handler Priority Register 2	0x0000_0000
SHPR3	SCB_BA+0x20	R/W	System Handler Priority Register 3	0x0000_0000

CPUID Register (CPUID)

Register	Offset	R/W	Description	Reset Value
CPUID	SCB_BA+0x00	R	CPUID Register	0x410C_C200

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

Bits	Description	
[31:24]	**IMPLEMENTER**	**Implementer Code Assigned by ARM®** Implementer code assigned by ARM®. (ARM® = 0x41)..
[23:20]	**Reserved**	Reserved.
[19:16]	**PART**	**Architecture of the Processor** Read as 0xC corresponding to ARMv6-M architecture.
[15:4]	**PARTNO**	**Part Number of the Processor** Reads as 0xC20 corresponding to Cortex®-M0
[3:0]	**REVISION**	**Revision Number** Reads as 0x0.

Interrupt Control State Register (ICSR)

The ICSR provides (1)a set-pending bit for the NMI exception, and (2)set-pending and clear-pending bits for the PendSV and SysTick exceptions. And the ICSR also indicates (1)the exception number of the exception being processed, (2)whether there are preempted active exceptions, (3)the exception number of the highest prority pending exception, and (4)whether any interrupts are pending.

Register	Offset	R/W	Description	Reset Value
ICSR	SCB_BA+0x04	R/W	Interrupt Control 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
Reserved	ISRPENDING	Reserved				VECTPENDING	
15	14	13	12	11	10	9	8
VECTPENDING				Reserved			
7	6	5	4	3	2	1	0
Reserved		VECTACTIVE					

Bits	Description	
[31]	**NMIPENDSET**	**NMI Set-pending Bit** Because NMI is the highest-priority exception, normally the processor enters theNMI exception handler as soon as it detects a write of 1 to this bit. Entering thehandler then clears this bit to 0. This means a read of this bit by the NMIexception handler returns 1 only if the NMI signal is reasserted while theprocessor is executing that handler. Write Operation: 0 = no effect. 1 = changes NMI exception state to pending. Read Operation: 0 = NMI exception is not pending. 1 = NMI exception is pending.
[30:29]	**Reserved**	Reserved.
[28]	**PENDSVSET**	**PendSV Set-pending Bit** Write Operation: 0 = no effect. 1 = changes PendSV exception state to pending. Read Operation: 0 = PendSV exception is not pending. 1 = PendSV exception is pending. Writing 1 to this bit is the only way to set the PendSV exception state to pending.
[27]	**PENDSVCLR**	**PendSV Clear-pending Bit(Write Only)** 0 = no effect. 1 = removes the pending state from the PendSV exception.

Bits	Description	
[26]	**PENDSTSET**	**SysTick Exception Set-pending Bit** Write Operation: 0 = no effect. 1 = changes SysTick exception state to pending. Read Operation: 0 = SysTick exception is not pending. 1 = SysTick exception is pending.
[25]	**PENDSTCLR**	**SysTick Exception Clear-pending Bit(Write Only)** 0 = no effect. 1 = removes the pending state from the SysTick exception.
[24:23]	**Reserved**	Reserved.
[22]	**ISRPENDING**	**Interrupt Pending Flag(Read Only)** Indicates if an external configurable (NVIC generated) interrupt is pending. 0 = interrupt not pending. 1 = interrupt pending.
[21:18]	**Reserved**	Reserved.
[17:12]	**VECTPENDING**	**Vector Pending Indicator (Read Only)** This field indicates the exception number of the highest priority pending enabledexception: 0 = no pending exceptions. Nonzero = the exception number of the highest priority pending enabledexception.
[11:6]	**Reserved**	Reserved.
[5:0]	**VECTACTIVE**	**Vector Active Indicator (Read Only)** This field contains the active exception number: 0 = Thread mode. Nonzero = The exception number of the currently active exception.

Application Interrupt and Reset Control Register (AIRCR)

Register	Offset	R/W	Description	Reset Value
AIRCR	SCB_BA+0x0C	R/W	Application Interrupt and Reset Control Register	0xFA05_0000

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

Bits	Description	
[31:16]	**VECTKEY**	**Register Key(Write Only)** When writing to this register, the VECTORKEY field need to be set to 0x05FA, otherwise the write operation would be ignored. The VECTORKEY filed is used to prevent accidental write to this register from resetting the system or clearing of the exception status.
[15:3]	**Reserved**	Reserved.
[2]	**SYSRESETREQ**	**System Reset Request(Write Only)** 0 = no effect. 1 = requests a system level reset.
[1:0]	**Reserved**	Reserved.

System Control Register (SCR)

Register	Offset	R/W	Description	Reset Value
SCR	SCB_BA+0x10	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 = Oonly enabled interrupts or events can wakeup the processor, disabled interrupts areexcluded. 1 = Enabled events and all interrupts, including disabled interrupts, can wake-up theprocessor. When an event or interrupt enters pending state, the event signal wakes up the processorfrom WFE. If the processor is not waiting for an event, the event is registered and affectsnext WFE. The processor also wakes up on execution of an SEV instruction or an external event.
[3]	**Reserved**	Reserved.
[2]	**SLEEPDEEP**	**Deep Sleep Mode Enable Bit** This bit controls whether the processor uses sleep or deep sleep as its low power mode: 0 = Sleep mode. 1 = Deep sleep mode.
[1]	**SLEEPONEXIT**	**Sleep-on-exit Enable Bit** This bit controls sleep-on-exit when returning from Handler mode to Thread mode: 0 = Do not sleep when returning to Thread mode. 1 = Enter Sleep, or Deep Sleep, on return from an ISR to Thread mode. Setting this bit to 1 enables an interrupt driven application to avoid returning to an emptymain application.
[0]	**Reserved**	Reserved.

System Handler Priority Register 2 (SHPR2)

Register	Offset	R/W	Description	Reset Value
SHPR2	SCB_BA+0x1C	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 ofSystem 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	SCB_BA+0x20	R/W	System Handler Priority Register 3	0x0000_0000

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

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

6.3 Clock Controller

6.3.1 Overview

The clock controller generates the clocks for the wholechip, includingsystem 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 clock divider. The chip enters Power-down mode when Cortex®-M0 core executes the WFI instruction only if the SLEEPDEEP (SCR[2]) bit is set to 1. After that, chip entersPower-down mode and wait for wake-up interrupt source triggered to leave Power-down mode. In the Power-down mode, the clock controller turns off the 4~24 MHz external high speed crystal oscillator and 22.1184 MHz internal high speed RC oscillator to reduce the overall system power consumption.The Figure 6–3 and Figure 6–4 show the clock generator and the overview of the clock source control.

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

- 4~24 MHz external high speed crystal oscillator (HXT)
- Programmable PLL output clock frequency (PLL_FOUT), PLL source can be selected from 4~24 MHz external high speed crystal oscillator (HXT) or 22.1184 MHz internal high speed RC oscillator (HIRC)
- 22.1184 MHz internal high speed RC oscillator (HIRC)
- 10 kHzinternal low speed RC oscillator (LIRC)

Each of these clock sources has certain stable time to wait for clock operating at stable frequency. When clock source is enabled, a stable counter start counting and correlated clock stable index (OSC22M_STB(CLKSTATUS[4]), OSC10K_STB(CLKSTATUS[3]), PLL_STB(CLKSTATUS[2]) and XTL12M_STB(CLKSTATUS[0])) are set to 1 after stable counter value reach a define value as Table 6-5. System and peripheral can use these clock as its operating clock only when correlate clock stable index is set to 1. The clock stable index will auto clear when user disables the clock source (OSC10K_EN(PWRCON[3]), OSC22M_EN(PWRCON[2]), XTL12M_EN(PWRCON[0]) and PD(PLLCON[16])). Besides, the clock stable index of HXT, HIRC and PLL will auto clear when chip enter power-down and clock stable counter will re-counting after chip wake-up if correlate clock is enabled.

Clock Source	Clock Stable Count Value
HXT	4096 HXT clock
PLL	6144 PLL source (PLL source is HXT if PLL_SRC(PLLCON[19]) = 0, or HIRC if PLL_SRC(PLLCON[19]) = 1)
HIRC	256 HIRC clock
LIRC	1 LIRC

Table 6-5Clock Stable Count Value Table

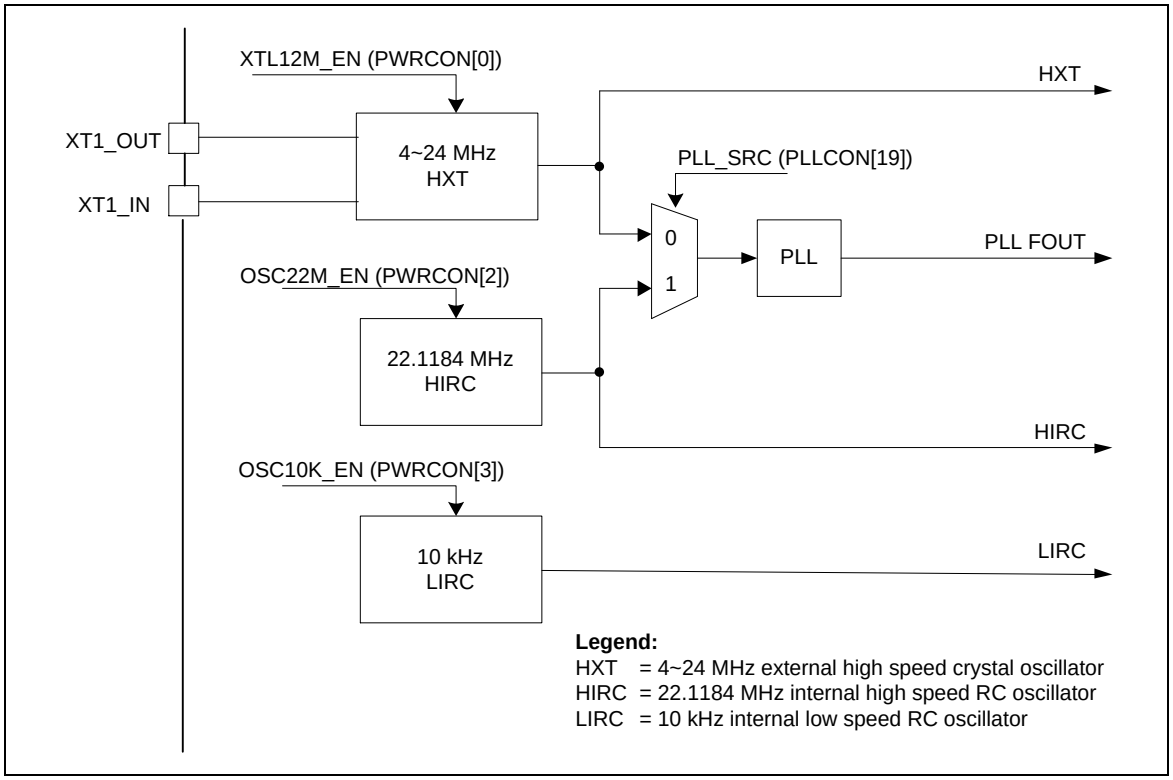
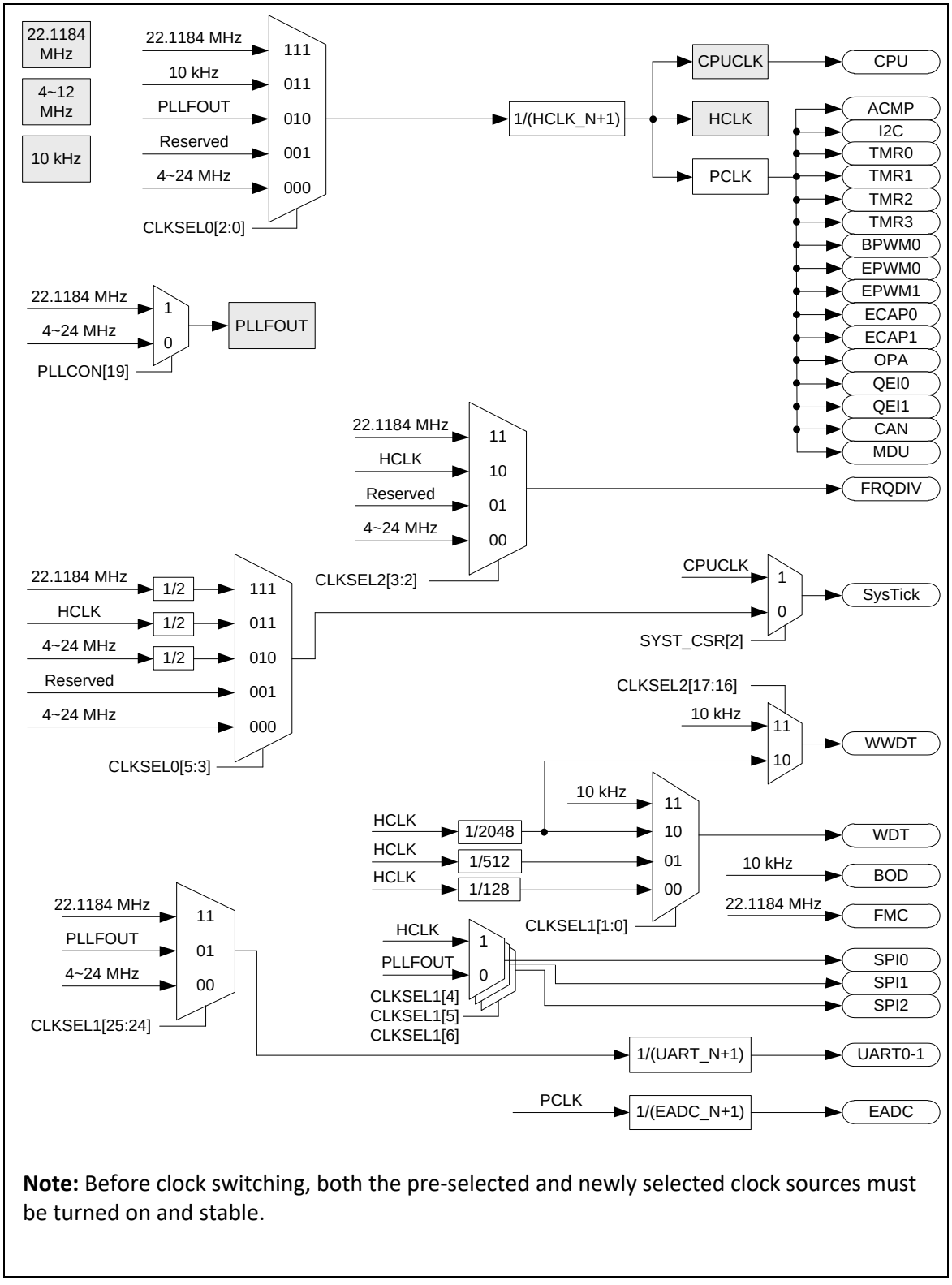


Figure 6–3 Clock Generator Block Diagram



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

Figure 6–4 Clock Generator Global View Diagram

6.3.2 System Clock & SysTick Clock

The system clock has 4 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 showedin Figure 6–5.

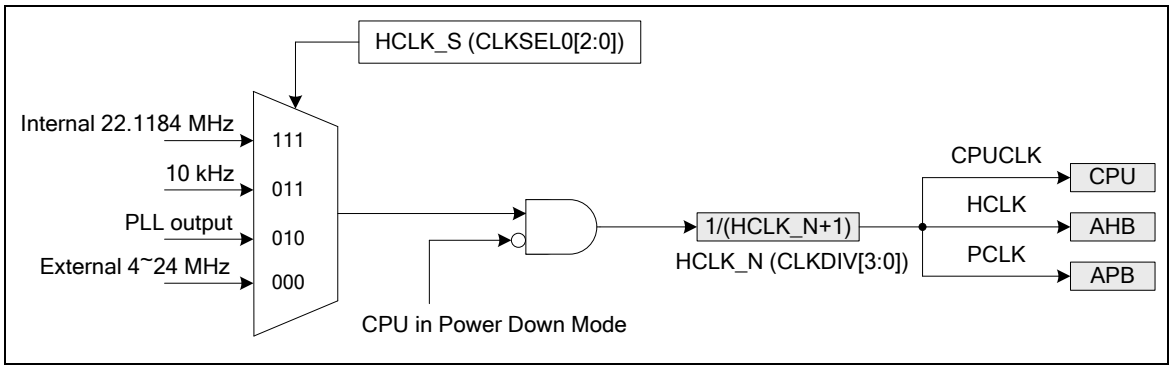


Figure 6–5 System Clock Block Diagram

The clock source of SysTick clock (STCLK) in Cortex®-M0 core comes from 4 clock sources. The clock source switch depends on the setting of the register STCLK_S (CLKSEL0[5:3]). The block diagram is showed in Figure 6–6.

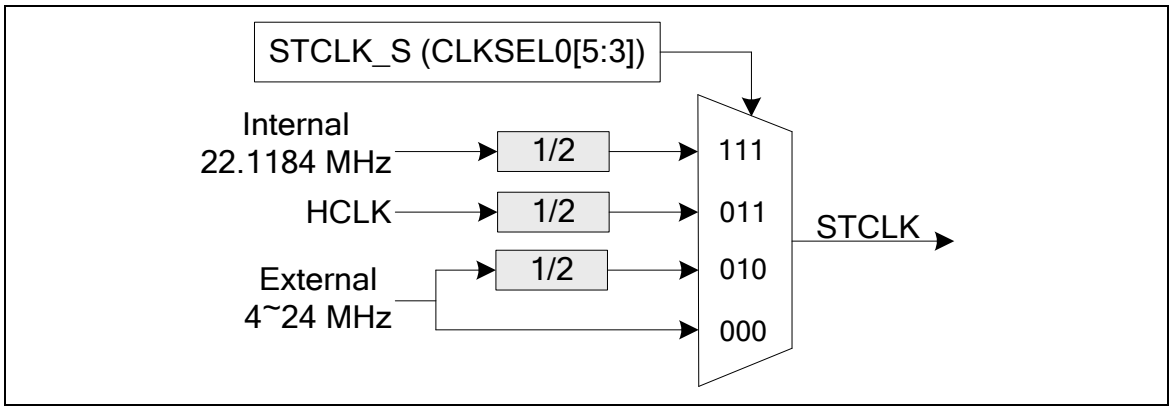


Figure 6–6 SysTick Clock Control Block Diagram

6.3.3 Peripherals Clock

The peripherals clock had different clock source switch setting which depends on the different peripheral. Please refer the CLKSEL1 and CLKSEL2 register description in 6.3.7.

6.3.4 Power-down Mode (Deep Sleep Mode) Clock

When chip enters 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.

Clocks and peripherals which still keep active:

- Clock Generator
  - Internal 10kHz oscillator clock.
- Peripherals Clock (when these peripheraladopt 10 kHz as clock source)

- Watchdog Timer.

6.3.5 Frequency Divider Output

This device is equipped a frequency divider which is composed by16 chained divide-by-2 shift registers. One of the 16 shift register outputs selected by a sixteen to one multiplexer is reflected to CLKO function pin. Therefore there are 16 options of divided clocks.

The output formula is $F_{CLKO} = F_{FRQDIV\_CLK}/2^{(N+1)}$, where F_{FRQDIV_CLK} is the input clock frequency, F_{CLKO} 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.

If DIV1(FRQDIV[5]) is set to 1, the frequency divider clock (FRQDIV_CLK) will bypass power-of-2 frequency divider. The frequency divider clock will be output to CLKO pin directly.

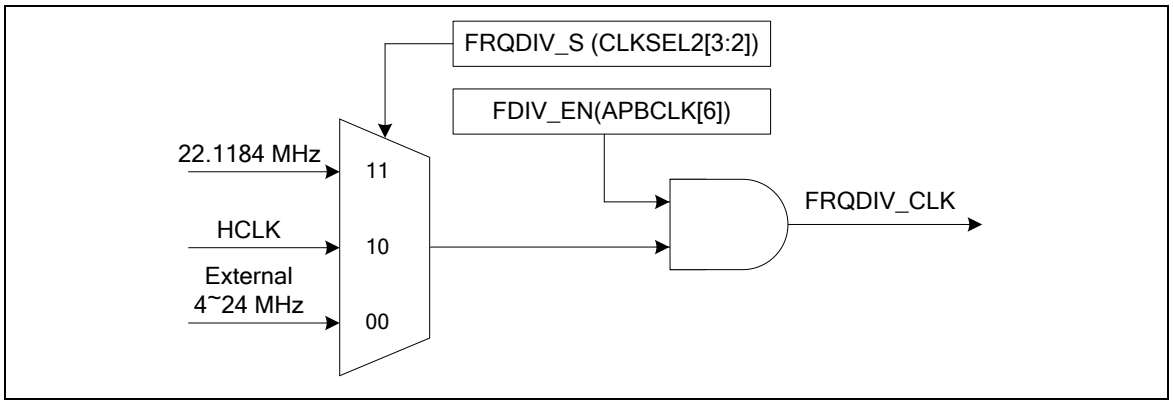


Figure 6–7 Clock Source of Frequency Divider

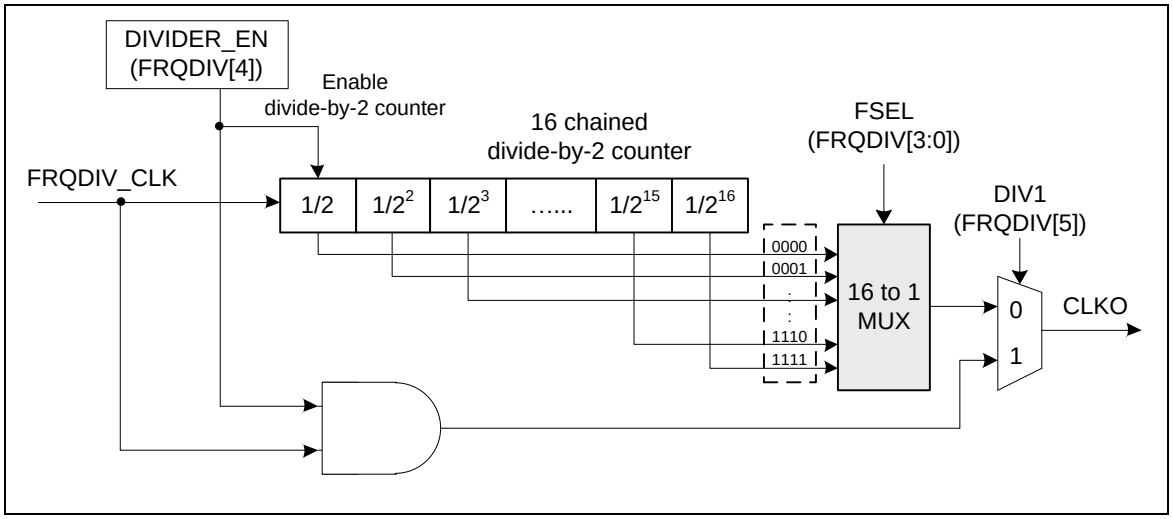


Figure 6–8 Block Diagram of Frequency Divider

6.3.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
CLK Base Address: **CLK_BA = 0x5000_0200**				
PWRCON	CLK_BA+0x00	R/W	System Power-down Control Register	0x0000_000X
AHBCLK	CLK_BA+0x04	R/W	AHB Devices Clock Enable Control Register	0x0000_0005
APBCLK	CLK_BA+0x08	R/W	APB Devices Clock Enable Control Register	0x0000_0001
CLKSTATUS	CLK_BA+0x0C	R	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
CLKDIV	CLK_BA+0x18	R/W	Clock Divider Number Register	0x0000_0000
CLKSEL2	CLK_BA+0x1C	R/W	Clock Source Select Control Register 2	0xFFFF_FFFF
PLLCON	CLK_BA+0x20	R/W	PLL Control Register	0x0005_C22E
FRQDIV	CLK_BA+0x24	R/W	Frequency Divider Control Register	0x0000_0000
CLKDCR	CLK_BA+0x28	R/W	Clock Detect Control Register	0x0000_0000
CLKDSR	CLK_BA+0x2C	R/W	Clock Detect Status Register	0x0000_0000

6.3.7 Register Description

System Power-down Control Register（PWRCON）

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

Register	Offset	R/W	Description	Reset Value
PWRCON	CLK_BA+0x00	R/W	System Power-down Control Register	0x0000_000X

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**PD_WU_STS**	**PD_WU_INT_EN**	**Reserved**	**OSC10K_EN**	**OSC22M_EN**	**Reserved**	**XTL12M_EN**

Bits	Description	
[31:7]	**Reserved**	Reserved.
[15:7]	**Reserved**	Reserved.
[6]	**PD_WU_STS**	**Power-down Mode Wake-up Interrupt Status** Set by “power down wake up event”, it indicates that resume from Power-down mode. Write 1 to clear the bit to zero. **Note:** This bit is working only if PD_WU_INT_EN (PWRCON[5]) set to 1.
[5]	**PD_WU_INT_EN**	**Power-down Mode Wake-up Interrupt Enable Bit (WriteProtect)** 0 = Power-down mode wake-up interrupt Disabled. 1 = Power-down mode wake-up interrupt Enabled. **Note1:** The interrupt will occur when both PD_WU_STS and PD_WU_INT_EN are high. **Note2:** This bit is write protected bit. Refer to the REGWRPROT register.
[4]	**Reserved**	Reserved.
[3]	**OSC10K_EN**	**Internal 10 kHz Oscillator Enable Bit (WriteProtect)** 0 = 10 kHz Oscillation Disabled. 1 = 10 kHz Oscillation Enabled. **Note:** This bit iswrite protected bit. Refer to the REGWRPROT register.
[2]	**OSC22M_EN**	**Internal 22.1184 MHz Oscillator Enable Bit (WriteProtect)** 0 = 22.1184 MHz OscillationDisabled. 1 = 22.1184 MHz OscillationEnabled. **Note:** This bit iswrite protected bit. Refer to the REGWRPROT register.

Bits	Description	
[1]	**Reserved**	Reserved.
[0]	**XTL12M_EN**	**External 4~24 MHz Crystal Enable Bit (WriteProtect)** The bit default value is set by flash controller user configuration register CFOSC (Config0[26:24]). When the default clock source is from external 4~24 MHz crystal, this bit is set to 1 automatically. 0 = External 4~24 MHz crystalDisabled. 1 = External 4~24 MHz crystalEnabled. **Note:** This bit iswrite protected bit. Refer to the REGWRPROT register.

Chip Operating Mode	SLEEPDEEP (SCR[2])	CPU Run WFI Instruction	Clock Disable
Normal Running Mode	0	NO	All Clock are disabled by control register.
IDLE Mode (CPU entry Sleep Mode)	0	YES	Only CPU clock is disabled.
Power-down Mode (CPU entry deep sleep mode)	1	YES	Most Clock are disabled except 10K WDT clock still enabled.

Table 6-6 Power-down mode Control Table

AHB Devices Clock Enable Control Register （AHBCLK）

These bits for this register are used to enable/disable clock for AHB devicesclock.

Register	Offset	R/W	Description	Reset Value
AHBCLK	CLK_BA+0x04	R/W	AHB Devices Clock Enable Control Register	0x0000_0005

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

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**HDIV_EN**	**Hardware Divider Controller Clock Enable Control** 0 = Hardware Divider engine clock Disabled. 1 = Hardware Divider engine clock Enabled.
[3:0]	**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_0001

31	30	29	28	27	26	25	24
QEI1_EN	QEI0_EN	OPA_EN	EADC_EN	ECAP1_EN	ECAP0_EN	Reserved	CAN_EN
23	22	21	20	19	18	17	16
Reserved	ACMP_EN	EPWM1_EN	EPWM0_EN	BPWM0_EN	MDU_EN	UART1_EN	UART0_EN
15	14	13	12	11	10	9	8
Reserved	SPI2_EN	SPI1_EN	SPI0_EN	Reserved			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]	QEI1_EN	**QEI1 Clock Enable Bit** 0 = QEI1clock Disabled. 1 = QEI1clock Enabled.
[30]	QEI0_EN	**QEI0 Clock Enable Bit** 0 = QEI0clock Disabled. 1 = QEI0clock Enabled.
[29]	OPA_EN	**OPA0 and OPA1 Clock Enable Bit** 0 = OPA0 and OPA1clock Disabled. 1 = OPA0 and OPA1clock Enabled.
[28]	EADC_EN	**EADC Clock Enable Bit** 0 = EADC clock Disabled. 1 = EADC clock Enabled.
[27]	ECAP1_EN	**Enhanced Input Capture 1Clock Enable Bit** 0 = Enhanced input capture 1 clock Disabled. 1 = Enhanced input capture 1 clock Enabled.
[26]	ECAP0_EN	**Enhanced Input Capture 0Clock Enable Bit** 0 = Enhanced input capture 0 clock Disabled. 1 = Enhanced input capture 0 clock Enabled.
[25]	Reserved	Reserved.
[24]	CAN_EN	**CAN Bus Controller Clock Enable Bit** 0 = CAN clock Disabled. 1 = CAN clock Enabled.
[23]	Reserved	Reserved.

Bits	Description	
[22]	**ACMP_EN**	**Analog Comparator Clock Enable Bit** 0 = Analog comparator clock Disabled. 1 = Analog comparator clock Enabled.
[21]	**EPWM1_EN**	**Enhanced PWM1Clock Enable Bit** 0 = Enhanced PWM1 clock Disabled. 1 = Enhanced PWM1 clock Enabled.
[20]	**EPWM0_EN**	**Enhanced PWM0Clock Enable Bit** 0 = Enhanced PWM0 clock Disabled. 1 = Enhanced PWM0 clock Enabled.
[19]	**BPWM0_EN**	**Basic PWM0Clock Enable Bit** 0 = Basic PWM0 clock Disabled. 1 = Basic PWM0 clock Enabled.
[18]	**MDU_EN**	**MDU Clock Enable Bit** 0 = MDU clock Disabled. 1 = MDU clock Enabled.
[17]	**UART1_EN**	**UART1 Clock Enable Bit** 0 = UART1 clock Disabled. 1 = UART1 clock Enabled.
[16]	**UART0_EN**	**UART0 Clock Enable Bit** 0 = UART0 clock Disabled. 1 = UART0 clock Enabled.
[15]	**Reserved**	Reserved.
[14]	**SPI2_EN**	**SPI2Clock Enable Bit** 0 = SPI2clock Disabled. 1 = SPI2clock Enabled.
[13]	**SPI1_EN**	**SPI1Clock Enable Bit** 0 = SPI1clock Disabled. 1 = SPI1clock Enabled.
[12]	**SPI0_EN**	**SPI0Clock Enable Bit** 0 = SPI0clock Disabled. 1 = SPI0clock Enabled.
[11:9]	**Reserved**	Reserved.
[8]	**I2C0_EN**	**I^2C0Clock Enable Bit** 0 = I^2C0clock Disabled. 1 = I^2C0clock Enabled.
[7]	**Reserved**	Reserved.
[6]	**FDIV_EN**	**Frequency Divider Output Clock Enable Bit** 0 = Frequency divider output clock Disabled. 1 = Frequency divider output clock Enabled.
[5]	**TMR3_EN**	**Timer3 Clock Enable Bit** 0 = Timer3 clock Disabled. 1 = Timer3 clock Enabled.

Bits	Description	
[4]	**TMR2_EN**	**Timer2 Clock Enable Bit** 0 = Timer2 clock Disabled. 1 = Timer2 clock Enabled.
[3]	**TMR1_EN**	**Timer1 Clock Enable Bit** 0 = Timer1 clock Disabled. 1 = Timer1 clock Enabled.
[2]	**TMR0_EN**	**Timer0 Clock Enable Bit** 0 = Timer0 clock Disabled. 1 = Timer0 clock Enabled.
[1]	**Reserved**	Reserved.
[0]	**WDT_EN**	**Watchdog Timer Clock EnableBit (Write Protect)** 0 = Watchdog Timer clock Disabled. 1 = Watchdog Timer clock Enabled. **Note:** This bit iswrite protected bit. Refer to the REGWRPROT register.

Clock Status Monitor Register（CLKSTATUS）

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

Register	Offset	R/W	Description	Reset Value
CLKSTATUS	CLK_BA+0x0C	R	Clock Status MonitorRegister	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(Read Only)** 0 = Clock switching success. 1 = Clock switching failure. This bit is an index that if current system clock source is match as user defined at HCLK_S (CLKSEL0[2:0]). When user switch system clock, the system clock source will keep old clock until the new clock is stable. During the period that waiting new clock stable, this bit will be an index shows system clock source is not match as user wanted.
[6:5]	**Reserved**	Reserved.
[4]	**OSC22M_STB**	**Internal 22.1184M HzOscillator Clock Source Stable Flag(Read Only)** 0 = Internal 22.1184MHz oscillator clock is not stable or disabled. 1 = Internal 22.1184MHz oscillator clock is stable and enabled.
[3]	**OSC10K_STB**	**Internal 10k HzClock Source Stable Flag(Read Only)** 0 = Internal 10k Hz oscillator clock is not stable or disabled. 1 = Internal 10k Hz oscillator clock is stable and enabled.
[2]	**PLL_STB**	**PLL Clock Source Stable Flag(Read Only)** 0 = PLL clock is not stable or disabled. 1 = PLL clock is stable in normal mode.
[1]	**Reserved**	Reserved.
[0]	**XTL12M_STB**	**External 4~24 MHz CrystalClock Source Stable Flag(Read Only)** 0 = External 4~24MHz crystal clock is not stable or disabled. 1 = External 4~24MHz crystal clock is stable and enabled.

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(WriteProtect)** If SYST_CSR[2]=0, SysTick uses listed clock source below. 000 =Clock source from external 4~24 MHz crystal clock. 001 = Reserved. 010 = Clock source from external 4~24 MHz crystal clock/2. 011 = Clock source from HCLK/2. 111 = Clock source from internal 22.1184 MHz oscillator clock/2. Others = Reserved. **Note1:** These bitsarewrite protected bits. Refer to the REGWRPROT register. **Note2:** if SysTick clock source is not from HCLK(i.e. SYST_CSR[2]=0), SysTick clock source must less than or equal to HCLK/2.
[2:0]	**HCLK_S**	**HCLK Clock Source Selection(WriteProtect)** 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. 000 = Clock source from external 4~24 MHz crystal clock. 001 = Reserved. 010 = Clock source fromPLL clock. 011 = Clock source frominternal 10 kHz oscillator clock. 111 = Clock source from internal 22.1184 MHz oscillator clock. Others = Reserved. **Note:** These bitsarewrite protected bits. Refer to the REGWRPROT register.

Clock Source Select Control Register 1（CLKSEL1）

Before clock switching, the related clock sources (pre-selected and new-selected) 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
Reserved						**UART_S**	
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	**SPI2_S**	**SPI1_S**	**SPI0_S**	**Reserved**		**WDT_S**	

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25:24]	**UART_S**	**UART Clock Source Selection** 00 = Clock source from external 4~24 MHz crystal clock. 01 = Clock source fromPLL clock. 10 = Reserved. 11 = Clock source from internal 22.1184 MHz oscillator clock.
[23:7]	**Reserved**	Reserved.
[6]	**SPI2_S**	**SPI2Clock Source Selection** 0 = Clock source from PLL clock. 1 = Clock source from HCLK.
[5]	**SPI1_S**	**SPI1Clock Source Selection** 0 = Clock source from PLL clock. 1 = Clock source from HCLK.
[4]	**SPI0_S**	**SPI0Clock Source Selection** 0 = Clock source from PLL clock. 1 = Clock source from HCLK.
[3:2]	**Reserved**	Reserved.
[1:0]	**WDT_S**	**Watchdog Timer Clock Source Selection (WriteProtect)** 00 = Clock source from HCLK/128 clock. 01 = Clock source from HCLK/512 clock. 10 = Clock source from HCLK/2048 clock. 11 = Clock source from internal 10 kHz oscillator clock. **Note:**These bitsarewrite protected bits. Refer to the REGWRPROT register.

Clock Source Select Control Register 2（CLKSEL2）

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

Register	Offset	R/W	Description	Reset Value
CLKSEL2	CLK_BA+0x1C	R/W	Clock Source Select Control Register 2	0xFFFF_FFFF

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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**FRQDIV_S**		**Reserved**	

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17:16]	**WWDT_S**	**Window Watchdog Timer Clock Source Selection** 00 = Reserved. 01 = Reserved. 10 = Clock source from HCLK/2048 clock. 11 = Clock source from internal 10 kHz low speed oscillator clock.
[15:4]	**Reserved**	Reserved.
[3:2]	**FRQDIV_S**	**Clock Divider Clock Source Selection** 00 = Clock source from external 4~24 MHz crystal clock. 01 = Reserved. 10 = Clock source from HCLK. 11 = Clock source from internal 22.1184 MHz oscillator clock.
[1:0]	**Reserved**	Reserved.

Clock Divider Number 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**
EADC_N							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**UART_N**			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**HCLK_N**			

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:16]	**EADC_N**	**EADC Clock Divider** The EADC clock frequency = (EADC clock source frequency) / (EADC_N + 1).
[15:12]	**Reserved**	Reserved.
[11:8]	**UART_N**	**UART Clock Divider** The UART clock frequency = (UART clock source frequency) / (UART_N + 1).
[7:4]	**Reserved**	Reserved.
[3:0]	**HCLK_N**	**HCLK Clock Divider** The HCLK clock frequency = (HCLK clock source frequency) / (HCLK_N + 1).

PLL Control Register（PLLCON）

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

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

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

Bits	Description	
[31:20]	**Reserved**	Reserved.
[20]	**FCO_SEL**	**PLL FCO Selection** 0 = When the FCO frequency range between 100MHz and 200MHz, this bit should be set as 0. 1 = When the FCO frequency range between 200MHz to 500MHz, this bit should be set as 1.
[19]	**PLL_SRC**	**PLL Source Clock Selection** 0 = PLL source clock from external 4~24 MHz crystal. 1 = PLL source clock from internal 22.1184 MHz oscillator.
[18]	**OE**	**PLL OE (FOUT Enable)Control** 0 =PLL FOUT enable. 1 =PLL FOUT is fixed low.
[17]	**BP**	**PLL Bypass Control** 0 =PLL is in normal mode (default). 1 =PLL clock output is same as clock input.
[16]	**PD**	**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 Bits** Refer to the formulas below the table.
[13:9]	**IN_DV**	**PLL Input Divider Control Bits** Refer to the formulas below the table.

Bits	Description	
[8:0]	**FB_DV**	**PLL Feedback Divider Control Bits** Refer to the formulas below the table.

Output Clock Frequency Setting

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

Constrain:

1. $3.2MHz < FIN < 150\,MHz$
2. $800\,KHz < Fref = \frac{FIN}{2 \times NR} < 8MHz$
3. $200\,MHz < FCO = Fref \times 2 \times NF = FIN \times \frac{NF}{NR} < 500\,MHz, (\text{FCO\_SEL} = 1)$

   $100\,MHz < FCO = Fref \times 2 \times NF = FIN \times \frac{NF}{NR} < 200\,MHz, (\text{FCO\_SEL} = 0)$

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 FrequencySetting

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 = 48MHz

Frequency Divider Control Register (FRQDIV)

Register	Offset	R/W	Description	Reset Value
FRQDIV	CLK_BA+0x24	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		**DIV1**	**DIVIDER_EN**	**FSEL**			

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**DIV1**	**Frequency Divider Divide 1Enable Bit** 0 = Frequency divider will output clock with source frequency divide by FSEL. 1 = Frequency divider will output clock with source frequency.
[4]	**DIVIDER_EN**	**Frequency Divider Enable Bit** 0 = Frequency divider Disabled. 1 = Frequency divider Enabled.
[3:0]	**FSEL**	**Frequency Divider Output Selection Bits** The output formula is: $F_{CLKO} = F_{FRQDIV\_CLK}/2^{(N+1)}$,. Where F_{FRQDIV_CLK} is the input clock frequency, F_{CLKO} is the clock divider output frequency and N is the 4-bit value in FSEL[3:0].

Clock Detect Control Register (CLKDCR)

Register	Offset	R/W	Description	Reset Value
CLKDCR	CLK_BA+0x28	R/W	Clock Detect 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						CKD_IEN	CKD_EN

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**CKD_IEN**	**Clock Detect Interrupt Enable Bit** 0 = Clock detectinterrupt Disabled. 1 = Clock detect interrupt Enabled.
[0]	**CKD_EN**	**Clock Detect Enable Bit** 0 = Clock detect Disabled. 1 = Clock detect Enabled.

Clock Detect Status Register (CLKDSR)

Register	Offset	R/W	Description	Reset Value
CLKDSR	CLK_BA+0x2C	R/W	Clock Detect 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						CKSTOP	CKSTOP_IF

Bits	Description	
[31:2]	Reserved	Reserved.
[1]	CKSTOP	Clock Stop Status This bit reflect clock detect circuit result.
[0]	CKSTOP_IF	Clock Stop Interrupt Flag When system clock stop is detected, this bit will be set. Software can write 1 to clear this bit.

6.4 Flash Memory Controller (FMC)

6.4.1 Overview

The NuMicro®NM1530 Series is equipped with 128/64/32KB on-chip embedded flash for application program memory (APROM) and data flash, and with 8K bytes for ISP loader program memory (LDROM) that could be programmed boot loader to updateAPROM and dataflash through In-System-Programming (ISP) procedure. ISP function enables user to update 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 (Config0[7:6]). By the way, the NuMicro®NM1530 Series also provides data flash for user to store some application dependent data before chip power off. For 128KB APROM device, the data flash is shared with original 128Kbprogram memory and its start address is configurable in Config1. For 64/32KB APROM device, the data flash is fixed at 4K bytes.

6.4.2 Features

- Runs up to 72 MHz and optional up to 50MHz with zero wait state for continuous address read access
- Supports 512 bytes page erase for all embedded flash
- Supports 128/64/32Kbytes application program ROM (APROM)
- Supports 8 KB loader ROM (LDROM)
- Supports 4KB data flash for 64/32Kbytes APROM device
- Supports configurable data flash size for 128KB APROM device
- Supports 8 bytes User Configuration block to control system initiation
- Support In-System-Programming (ISP) /In-Application-Programming (IAP) to update embedded flash memory

6.4.3 Block Diagram

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

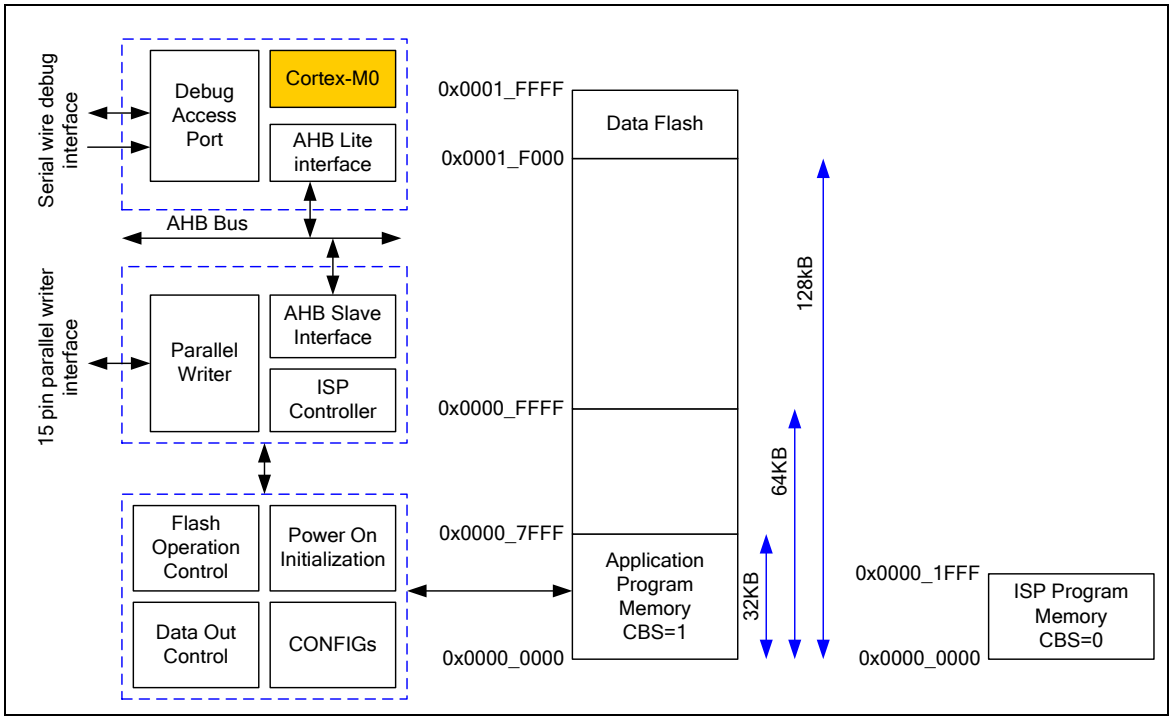


Figure 6–9 Flash Memory Control Block Diagram

6.4.4 Functional Description

FMC functions include the memory organization, boot selection, IAP, ISP, andISP procedure.

6.4.4.1 Memory Organization

The NuMicro®NM1530 Series flash memory consists of program memory (APROM), data flash, ISP loader program memory (LDROM), and user configuration.

Program memory is main memory for user applications and called APROM. User can write their application to APROM and set system to boot from APROM.

ISP loader program memory is designed for a loader to implement In-System-Programming function. LDROM is independent to APROM and system can also be set to boot from LDROM. Therefore, user can use LDROM to avoid system boot fail when code of APROM was corrupted.

Data flash is used for user to store data. It can be read by ISP read or memory read and programmed through ISP procedure. Because the size of each erase unit is 512 bytes, data flash should use 512 bytes as minimum unit. For 128 KB APROM device, the data flash and application program share the same 128 KB memory, if Data Flash Enable bit DFEN (Config0[0]) is enabled, the data flash base address is defined by DFBADR (Config1[19:0]) and its size is (0x20000 – DFBADR), At the same time, the APROM size will be (128 KB – data flash size). For 64/32 KB APROM devices, data flash size is always 4 KB and start address is fixed at 0x0001_F000.

User configuration provides several bytes to control system logic, such as flash security lock, boot select, Brown-out voltage level, data flash base address, etc.… User configuration works like a fuse for power on setting and loaded from flash memory to its corresponding control registers during chip powered on.

In NuMicro®NM1530 Series, the flash memory organization is different to system memory map. Flash memory organization is used when user using ISP command to read, program or erase flash memory. System memory map is used when CPU access flash memory to fetch code or data. For example, When system is set to boot from LDROM by CBS = 01b, CPU will be able to fetch code of LDROM from 0x0 ~ 0x1FFF. However, if user wants to read LDROM by ISP, they still need to read the address of LDROM as 0x0010_0000 ~ 0x0010_1FFF.

Table 6-7 shows the address mapping information of APROM, LDROM, data flash and user configuration for 32/64and 128 KB devices.

Block Name	Device Type	Size		Start Address	End Address
APROM	32 KB	32 KB		0x0000_0000	0x0000_7FFF
	64 KB	64 KB		0x0000_0000	0x0000_FFFF
	128 KB	Data Flash Enable	128 KB – Data Flash Size	0x0000_0000	0x20000 – Data Flash Size – 1
		Data Flash Disable	128 KB	0x0000_0000	0x0001_FFFF
Data Flash	32 KB	4 KB		0x0001_F000	0x0001_FFFF
	64 KB	4 KB		0x0001_F000	
	128 KB	Data Flash Enable	0x20000-DFBADR	DFBADR	
		Data Flash Disable	0 KB	N/A	N/A
LDROM	32/64/128 KB	8 KB		0x0010_0000	0x0010_1FFF
User Configuration	32/64/128 KB	2 words		0x0030_0000	0x0030_0004

Note: The erase page is 512 bytes; the bit 0 to bit8 of DFBADR should always keep 0

Table 6-7 Memory Address Map

The Flash memory organization is shown as Figure 6–10:

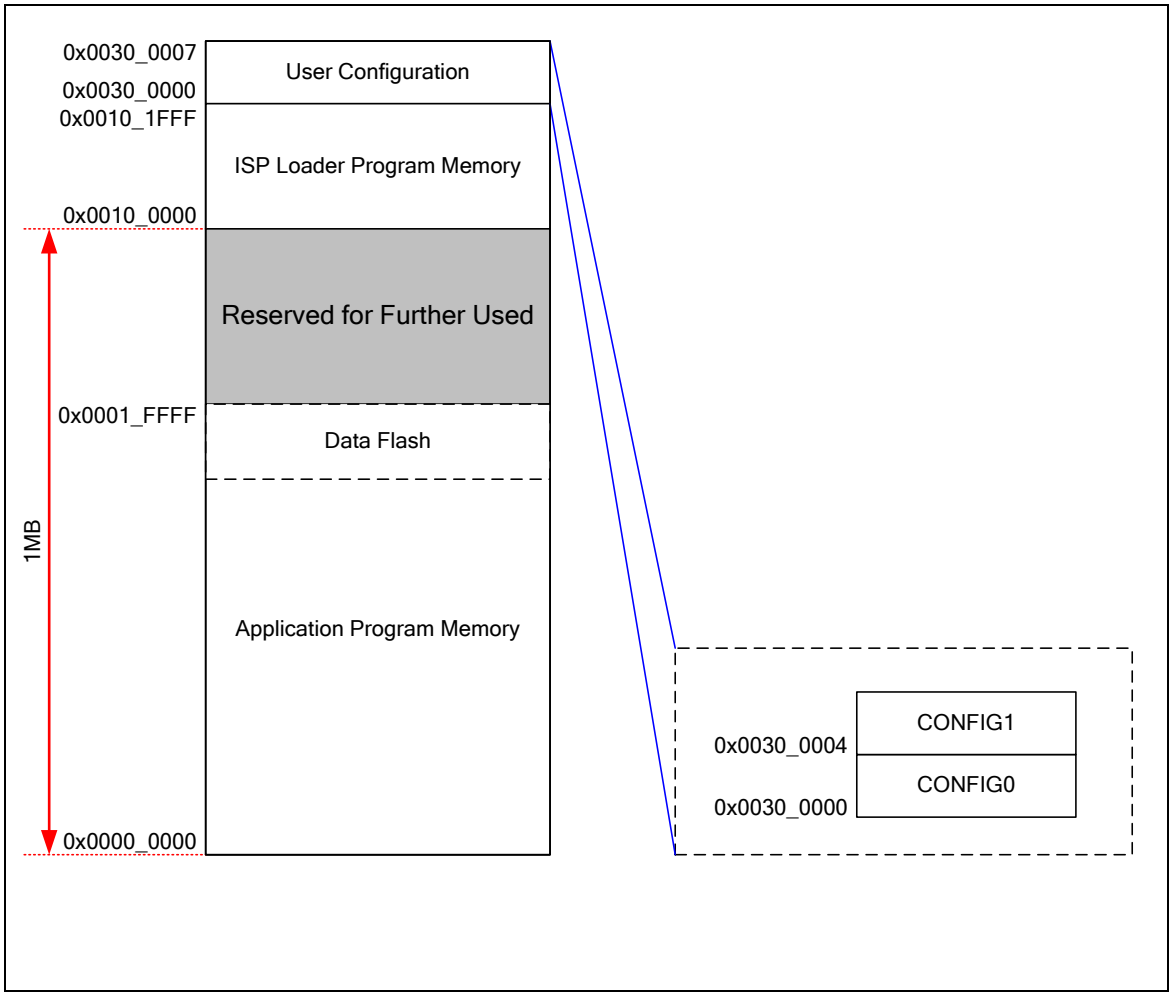


Figure 6–10 Flash Memory Organization

6.4.4.1 Boot Selection

The NuMicro®NM1530 Seriesprovides four booting sources for user to select, including LDROM with IAP, LDROM without IAP, APROM with IAP, and APROM without IAP. In any time, the booting source and system memory map are set by CBS (CONFIG0[7:6]).

CBS	Boot Selection/System Memory Map	Vector Mapping Supporting
00b	LDROM with IAP mode.	Yes
01b	LDROM without IAP mode.	No
10b	APROM with IAP mode.	Yes
11b	APROM without IAP mode.	No

Table 6-8Boot Selection Define Table

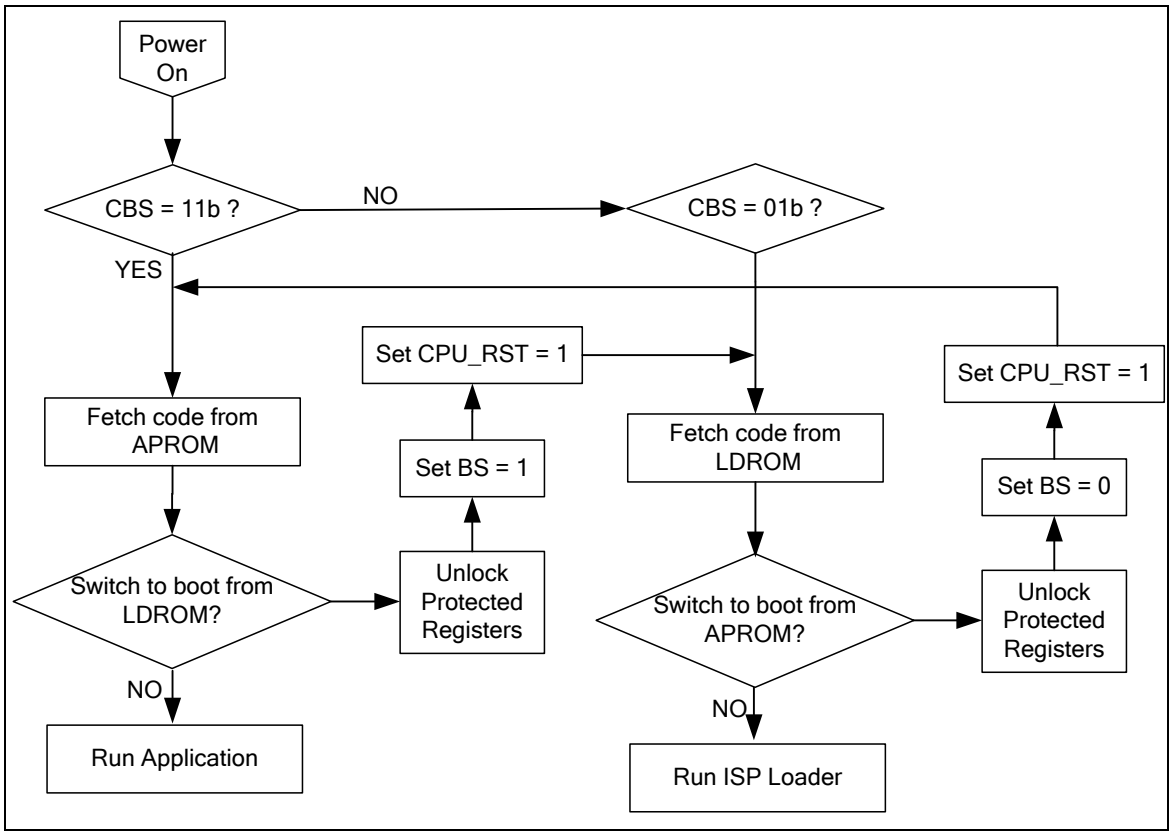


Figure 6-11 Example Flow of Boot Selection by BS Bit

6.4.4.2 In-Application-Programming (IAP)

The NuMicro®NM1530Series provides 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 setting the chip boot selection bits in Config0 (CBS[1:0]) as 10b or 00b and re-booting chip.

In the case that the chip 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 8 KB LDROM is mapped to 0x0010_0000~ 0x0010_1FFF.

In the case that the chip 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 8 KB LDROM is mapped to 0x0010_0000~0x0010_1FFF.

Please refer to Figure 6-12for the address map while IAP is activating.

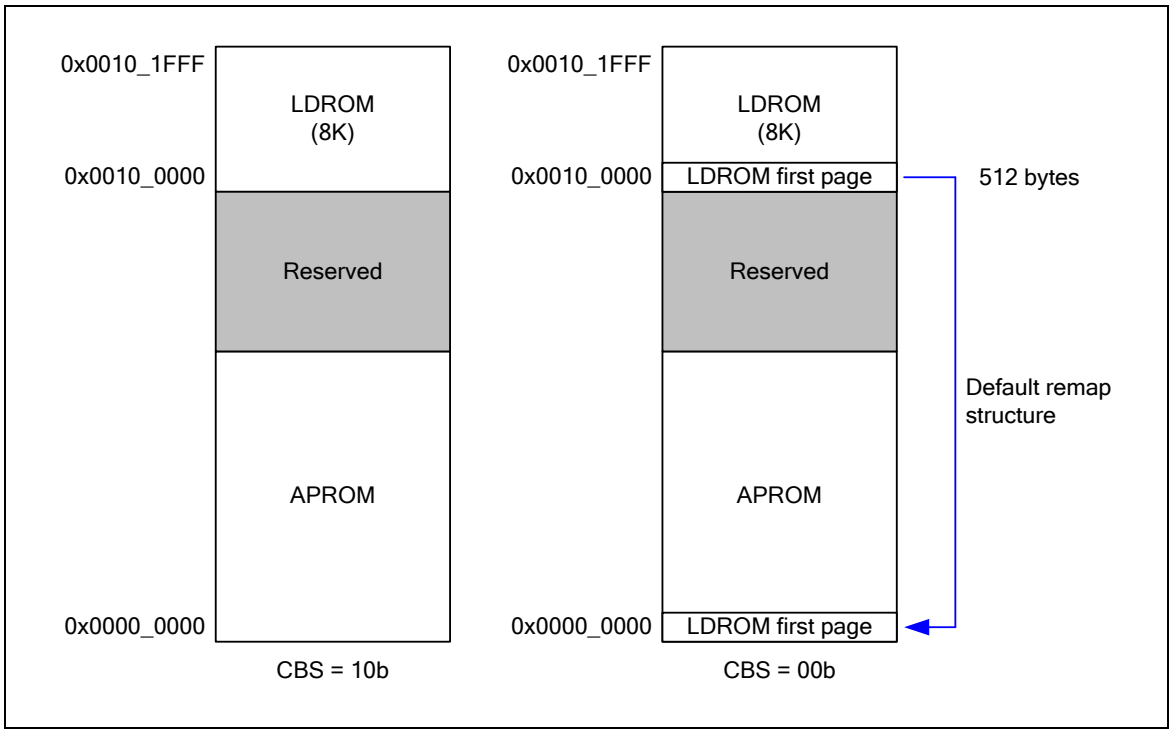


Figure 6-12 Executable Range of Code with IAP Function Enabled

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 changing the remap address, user can check if the change is successful by reading the VECMAP (ISPSTA[20:9]) field in the ISPSTA register.

6.4.4.3 In System Program (ISP)

The NuMicro®NM1530 Seriesprovides In-System-Programming (ISP) feature to enable user to update program memory by a stand-along ISP firmware. A dedicated 8 KB program memory (LDROM) is used to store ISP firmware. Users can select to start program fetch from APROM or LDROM by CBS[1](Config0[7]).

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 CPU 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 CPU read. Figure 6-13shows the memory map when boot from APROM and LDROM.

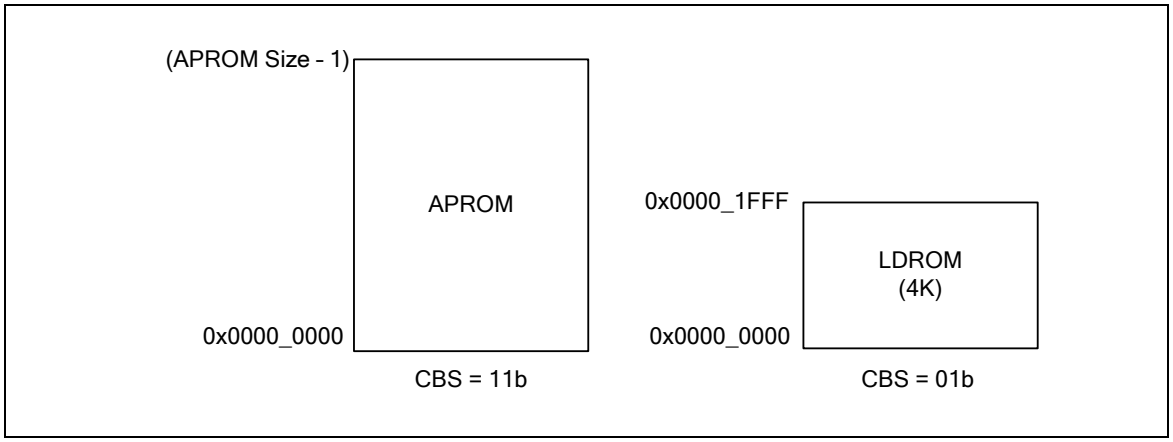


Figure 6-13 Executing Range for boot from APROM and boot from LDROM

NuMicro®NM1530 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®NM1530 Series enables user to easily perform ISP through Nuvoton ISP tool.

ISP Mode	ISPCMD	ISPADR	ISPDAT
FLASH Page Erase	0x22	Valid address of flash memory organization. It must be 512 bytes page alignment.	N/A
FLASH Program	0x21	Valid address of flash memory organization	Programming Data
FLASH Read	0x00	Valid address of flash memory organization	Return Data
Read Company ID	0x0B	0x0000_0000	0x0000_00DA
Read Unique ID	0x04	0x0000_0000	Unique ID Word 0
		0x0000_0004	Unique ID Word 1
		0x0000_0008	Unique ID Word 2
Vector Remap	0x2E	Valid address in APROM,LDROM or boot loader It must be 512 bytes alignment	N/A

Table 6-9 ISP Mode Command

6.4.4.4 ISP Procedure

The NuMicro®NM1530 Series supports booting from APROM or LDROM initially defined by user configuration bits (CBS). If user wants to update application program in APROM without IAP, he can write BS (ISPCON[1]) =1 and uses 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.

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

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

Several error conditions are checked after software sets ISPGO(ISPTRG[0]) to 1. If error condition occurs, ISP operation is not started and ISPFF (ISPSTA[6]) will be set. ISPFF (ISPSTA[6]) is cleared by software, it will not be over written in next ISP operation. The next ISP procedure can be started even ISPFF (ISPSTA[6]) keeps at 1. It is recommended that software to check ISPFF (ISPSTA[6]) and clear it after each ISP operation.

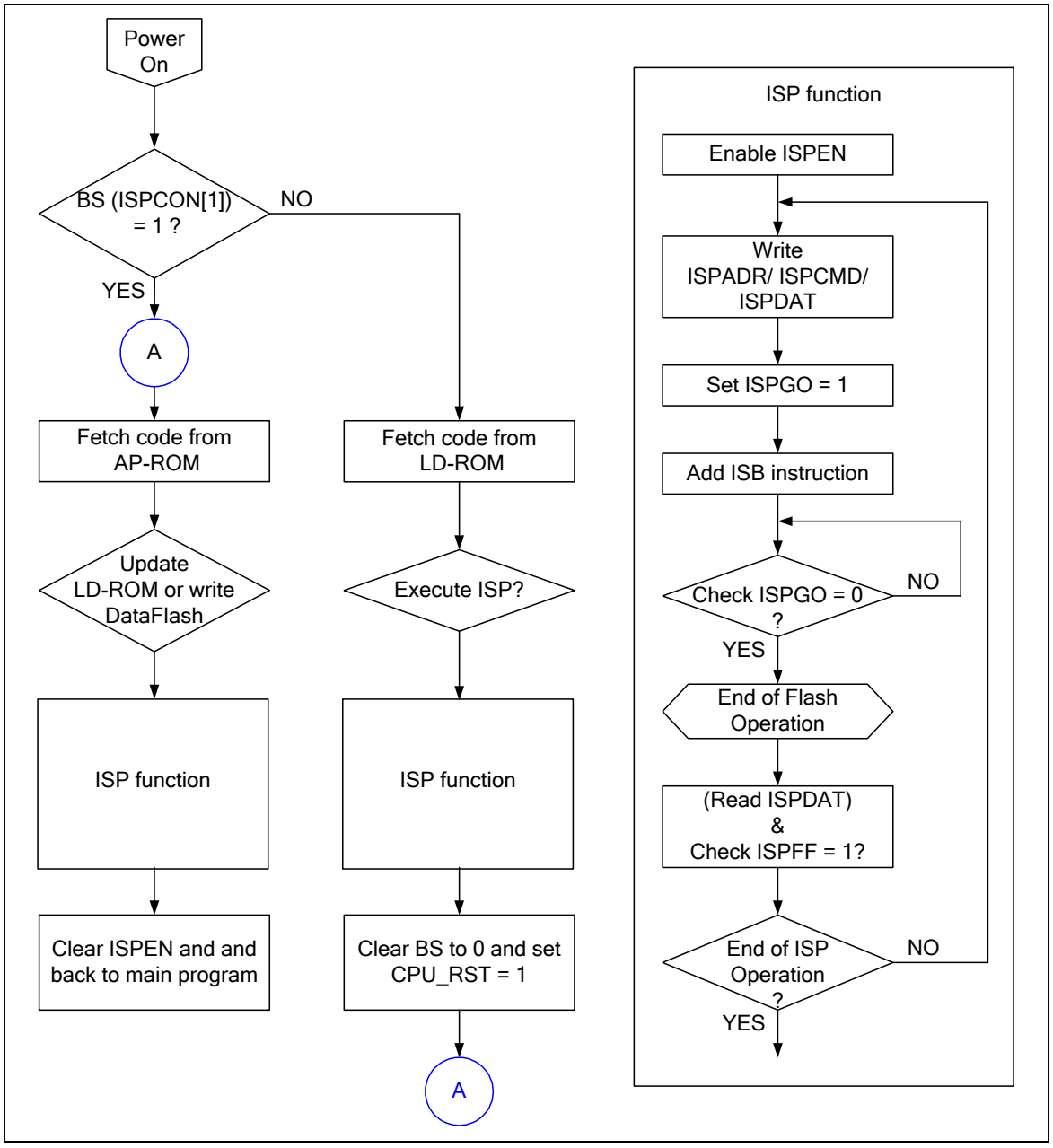


Figure 6-14ISP Procedure Example

6.4.4.1 Data Flash

The NuMicro®NM1530provides data flash for user to store data, which is read or written through ISP procedure. The size of each erase unit is 512 bytes. If a word need be changed, all 128 words have to be copied to another page or SRAM in advance based on 512 bytes limitation.

The data flash size is fixed 4 K bytes for the 64K bytes application program memory (APROM) device. The start address of data flash is fixed at 0x0001_F000. The data flash size is shared with APROM with variable size defined by user for the 128K bytes application program memory (APROM) device. If DFEN (Config0[0]) is set to 1, there is no data flash and all 128K bytes size is used for APROM. If DFEN (Config0[0]) is set to 0, the data flash share with APROM and its base

address is defined by DFBADR (Config1[19:0]). Under this setting, the application program memory size is (128-0.5*N)KB and data flash size is 0.5*N KB.

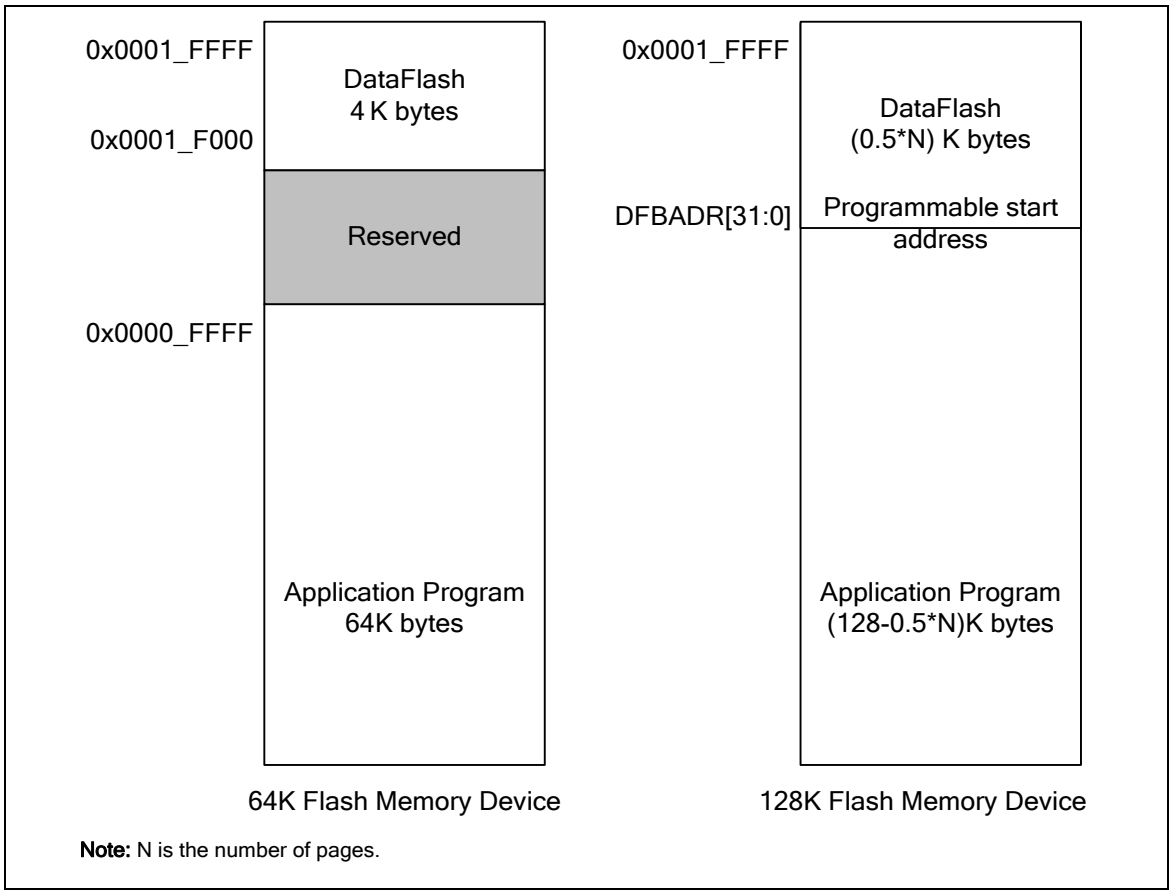


Figure 6-15 Flash Memory Structure

6.4.4.2 User Configuration

User configuration is internal programmable configuration area for boot options. The user configuration is located at 0x300000 of Flash Memory Organization and they are two 32 bits words. Any change on user configuration will take effect after system reboot.

Config0 (Address = 0x0030_0000)

31	30	29	28	27	26	25	24
CWDTEN	CWDTPDEN	Reserved			CFOSC		
23	22	21	20	19	18	17	16
CBODEN	CBOV1	CBOV0	CBORST	Reserved			
15	14	13	12	11	10	9	8
Reserved			CHZ_BPWM	CHZ_Odd1	CHZ_Even1	CHZ_Odd0	CHZ_Even0
7	6	5	4	3	2	1	0
CBS		Reserved				LOCK	DFEN

Config0 Bits	Description	
[31]	**CWDTEN**	**Watchdog EnableBit** 0 = Watchdog Timer Enabled and force Watchdog Timer clock source as OSC10K after chip powered on. 1 = Watchdog Timer Disabled after chip powered on.
[30]	**CWDTPDEN**	**Watchdog Clock Power Down EnableBit** 0 = OSC10K Watchdog Timer clock source is forced to be always enabled. 1 = OSC10K Watchdog Timer clock source is controlled by OSC10K_EN (PWRCON[3]) when chip enters power down. **Note:** This bit only works at CWDTEN is set to 0
[29:27]	**Reserved**	Reserved
[26:24]	**CFOSC**	**CPU Clock Source Selection After Reset** The value of CFOSC will be loaded to HCLK_S (CLKSEL0 [2:0]) in system register after any reset occurs. 000 = 4~24 MHz external high speed crystal oscillator (HXT) 111 = 22.1184 MHz internal high speed RC oscillator (HIRC) Others = Reserved
[23]	**CBODEN**	**Brown-Out Detector EnableBit** 0= Brown-out detect Enabled after powered on. 1= Brown-out detect Disabled after powered on.
[22:21]	**CBOV1-0**	**Brown-Out Voltage Selection** 00 = Brown-out voltage is 2.2V. 01 = Brown-out voltage is 2.7V. 10 = Brown-out voltage is 3.7V. 11 = Brown-out voltage is 4.4V.
[20]	**CBORST**	**Brown-out Reset EnableBit** 0 = Brown-out reset Enabled after powered on 1 = Brown-out reset Disabled after powered on
[19:13]	**Reserved**	Reserved

[12]	**CHZ_BPWM**	**Basic PWM0 Ports Tri-state Driving Control** 0 = Basic PWM0 ports driving mode is controlled by GPIO mode registers (GPIO_PMD). 1 = Basic PWM0 ports driving mode is forced in tri-state all the time. This bit will be load to GPIO control bit HZ_BPWM (PWMPOEN[4]) after any reset.
[11]	**CHZ_Odd1**	**PWM Unit1 odd Ports Tri-state Driving Control** 0 = PWM unit1 odd ports driving mode is controlled by GPIO mode registers (GPIO_PMD). 1 = PWM unit1 odd ports driving mode is forced in tri-state all the time. This bit will be load to GPIO control bit HZ_Odd1 (PWMPOEN[3]) after any reset.
[10]	**CHZ_Even1**	**PWM Unit0 Even Ports Tri-state Driving Control** 0 = PWM unit1 even ports driving mode is controlled by GPIO mode registers (GPIO_PMD). 1 = PWM unit1 even ports driving mode is forced in tri-state all the time. This bit will be load to GPIO control bit HZ_Even1 (PWMPOEN[2]) after any reset.
[9]	**CHZ_Odd0**	**PWM Unit0 Odd Ports Tri-state Driving Control** 0 = PWM unit0 odd ports driving mode is controlled by GPIO mode registers (GPIO_PMD). 1 = PWM unit0 odd ports driving mode is forced in tri-state all the time. This bit will be load to GPIO control bit HZ_Odd0 (PWMPOEN[1]) after any reset.
[8]	**CHZ_Even0**	**PWM Unit0 Even Ports Tri-state Driving Control** 0 = PWM unit0 even ports driving mode is controlled by GPIO mode registers (GPIO_PMD). 1 = PWM unit0 even ports driving mode is forced in tri-state all the time. This bit will be load to GPIO control bit HZ_Even0 (PWMPOEN[0]) after any reset.
[7:6]	**CBS**	**Chip Boot Selection** When CBS[0] = 0, IAP mode enabled, the LDROM base address is mapping to 0x100000 and APROM base address is mapping to 0x0. User could access both APROM and LDROM without boot switching. In other words, if IAP mode is supported, the code in LDROM and APROM can be called by each other. 00 = Boot from LDROM with IAP mode. 01 = Boot from LDROM without IAP mode. 10 = Boot from APROM with IAP mode. 11 = Boot from APROM without IAP mode. **Note:** BS (ISPCON[1]) is onlybe used to control boot switching when IAP mode disabled VECMAP (ISPSTA[20:9]) is only be used to remap 0x0~0x1ff when IAP mode enabled.
[5:2]	**Reserved**	Reserved
[1]	**LOCK**	**Security Lock** 0 = Flash memory contentis locked 1 = Flash memory contentis 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 0xFFFF_FFFF. ISP can read data anywhere regardless of LOCK bit value.
[0]	**DFEN**	**Data Flash EnableBit (Only for 128 KB APROM Device)** 0 = Data Flash Enabled. 1 = Data Flash Disabled.

Config1 (Address = 0x0030_0004)

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

Config1 Bits	Description	
[31:20]	**Reserved**	Reserved
[19:0]	**DFBADR**	**Data Flash Base Address (Only for 128 KB APROM Device)** For 128 KB APROM device, its data flash base address is defined by user. Since on-chip flash erase unit is 512 bytes, it is mandatory to keep bit 8-0 as 0. This configuration is only valid for 128 KB flash device.

6.4.5 Register Map

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

Register	Offset	R/W	Description	Reset Value
FMC Base Address: **FMC_BA = 0x5000_C000**				
ISPCON	FMC_BA+0x00	R/W	ISP Control Register	0x0000_000X
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 Control Register	0x0000_0000
DFBADR	FMC_BA+0x14	R	Data Flash Base Address	0x000X_XXXX
FATCON	FMC_BA+0x18	R/W	Flash Access Time Control Register	0x0000_0000
ISPSTA	FMC_BA+0x40	R/W	ISP Status Register	0x00XX_XX0X

6.4.6 Register Description

ISP Control Register (ISPCON)

Register	Offset	R/W	Description	Reset Value
ISPCON	FMC_BA+0x00	R/W	ISP Control Register	0x0000_000X

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

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**ISPFF**	**ISP Fail Flag (Write Protect)** This bit is set by hardware when a triggered ISP meets any of the following conditions: (1) APROM writes to itself if APUEN is set to 0. (2) LDROM writes to itself if LDUEN is set to 0. (3) CONFIG is erased/programmed if CFGUEN is set to 0. (4) Page Erase command at LOCK mode with ICE connection (5) Erase or Program command at brown-out detected (6) Destination address is illegal, such as over an available range. (7) Invalid ISP commands Write 1 to clear this bit.
[5]	**LDUEN**	**LDROM Update EnableBit (WriteProtect)** LDROM update enable bit. 0 = LDROM cannot be updated. 1 = LDROM can be updated when chip runs in APROM.
[4]	**CFGUEN**	**Config-bits Update Enable Bit (WriteProtect)** 0 = User Configuration cannot be updated. 1 = User Configuration can be updated.
[3]	**APUEN**	**APROM Update Enable Bit (Write Protect)** 0 = APROM cannot be updated when chip runs in APROM. 1 = APROM can be updated when chip runs in APROM.
[2]	**Reserved**	Reserved.

[1]	**BS**	**Boot Select (WriteProtect)** 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 0 = Boot from APROM. 1 = Boot from LDROM.
[0]	**ISPEN**	**ISP Enable Bit (WriteProtect)** ISP function enable bit. Set this bit to enable ISP function. 0 = ISP function Disabled. 1 = ISP function Enabled.

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							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ISPADR							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ISPADR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ISPADR							

Bits	Description	
[31:0]	**ISPADR**	**ISP Address** The NuMicro®NM1530 Serieshas a maximum 32Kx32 (128 KB) of embedded Flash, which 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							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
ISPDAT							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
ISPDAT							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
ISPDAT							

Bits	Description	
[31:0]	**ISPDAT**	**ISP Data** Write data to this register before ISP program operation Read data from this register after ISP read operation

ISPCommand (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		**CMD**					

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**CMD**	**ISP Command** ISP command table is shown below: 0x00= FLASH Read. 0x04= Read Unique ID. 0x0B= Read Company ID. 0x21= FLASH Program. 0x22= FLASH Page Erase. 0x2E= Vector Remap. The other commands are invalid.

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 Protect)** Write 1 to start ISP operation and this bit will be cleared to 0 by hardware automatically when ISP operation is finished. 0 = ISP operation finished. 1 = ISP progressed. 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 **Note:**To make sure ISP function has been finished before CPU goes ahead, ISB (Instruction Synchronization Barrier) instruction is used right after ISPGO (ISPTRG[0]) setting.

Data Flash Base Address Register (DFBADR)

Register	Offset	R/W	Description	Reset Value
DFBADR	FMC_BA+0x14	R	Data Flash Base Address	0x000X_XXXX

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

Bits	Description	
[31:0]	**DFBADR**	**Data Flash Base Address** This register indicates data flash start address. It is read only. For 128 KB flash memory device, the data flash size is defined by user configuration, register content is loaded from Config1 when chip is powered on but for 64/32 KB device, it is fixed at 0x0001_F000.

Flash Access Time Control Register (FATCON)

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

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

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**FOM_SEL[1]**	**Chip Frequency Optimization Mode Select (Write Protect)** When chip operation frequency is lower than 25 MHz, chip can work more efficiently by setting FOM_SEL[1:0]= 01. When chip operation frequency is lower than 50 MHz, chip can work more efficiently by setting FOM_SEL[1:0]= 00. When chip operation frequency is over than 50 MHz, chip only can work by setting FOM_SEL[1:0]= 11. 00 = Middle frequency optimization mode Enabled. 01 = Low frequency optimization mode Enabled. 10 = Reserved. 11 = High frequency optimization mode Enabled.
[5]	**Reserved**	Reserved.
[4]	**FOM_SEL[0]**	**Chip Frequency Optimization Mode Select (Write Protect)** When chip operation frequency is lower than 25 MHz, chip can work more efficiently by setting FOM_SEL[1:0]= 01. When chip operation frequency is lower than 50 MHz, chip can work more efficiently by setting FOM_SEL[1:0]= 00. When chip operation frequency is over than 50 MHz, chip only can work by setting FOM_SEL[1:0]= 11. 00 = Middle frequency optimization mode Enabled. 01 = Low frequency optimization mode Enabled. 10 = Reserved. 11 = High frequency optimization mode Enabled.
[3:0]	**Reserved**	Reserved.

ISP Status Register (ISPSTA)

Register	Offset	R/W	Description	Reset Value
ISPSTA	FMC_BA+0x40	R/W	ISP Status Register	0x00XX_XX0X

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

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20:9]	**VECMAP**	**Vector Page Mapping Address (Read Only)** The current flash address space 0x0000_0000~0x0000_01FF is mapping to address {VECMAP[11:0], 9'h000} ~ {VECMAP[11:0], 9'h1FF} **Note:** vector map function only workable when IAP mode enabled
[8:7]	**Reserved**	Reserved.
[6]	**ISPFF**	**ISP Fail Flag (Write Protect)** This bit is the mirror of ISPFF (ISPCON[6]), it needs to be cleared by writing 1 to ISPCON[6] or FMC_ISPSTA[6]. This bit is set by hardware when a triggered ISP meets any of the following conditions: (1) APROM writes to itself if APUEN is set to 0. (2) LDROM writes to itself if LDUEN is set to 0. (3) CONFIG is erased/programmed if CFGUEN is set to 0. (4) Page Erase command at LOCK mode with ICE connection (5) Erase or Program command at brown-out detected (6) Destination address is illegal, such as over an available range. (7) Invalid ISP commands Write 1 to clear this bit.
[5:3]	**Reserved**	Reserved.

[2:1]	**CBS**	**Chip Boot Selectionof CONFIG (Read Only)** When CBS[0] = 0, the LDROM base address is mapping to 0x100000 and APROM base address is mapping to 0x0. User could access both APROM and LDROM without boot switching. In other words, if IAP mode is supported, the code in LDROM and APROM can be called by each other. 00 = Boot from LDROM with IAP mode. 01 = Boot from LDROM without IAP mode. 10 = Boot from APROM with IAP mode. 11 = Boot from APROM without IAP mode. **Note:** BS (ISPCON[1]) is onlybe used to control boot switching when CBS[0] = 1. VECMAP (ISPSTA[20:9]) is only be used to remap 0x0~0x1ff when CBS[0] = 0.
[0]	**ISPGO**	**ISP Start Trigger (Read Only)** Write 1 to start ISP operation and this bit will be cleared to 0 by hardware automatically when ISP operation is finished. 0 = ISP operation finished. 1 = ISP operation progressed. **Note:** This bit is the same with ISPTRG bit0

6.5 General Purpose I/O (GPIO)

6.5.1 Overview

The NuMicro®NM1530 Series has up to 82 General Purpose I/O pins to be shared with other function pins depending on the chip configuration. These 82 pins are arranged in 10 ports named as P0, P1, P2, P3, P4, P5, P6, P7, P8, P9 and PA. The P0/1/2/3/4/5/6/7/8/9 port has the maximum of 8 pins and PA port has the maximum of 2 pins. Each of the 82 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 mode of all pins are stay at input mode. In Quasi-bidirectional mode, I/O pin has a very weak individual pull-up resistor which is about 110~300 KΩ for V_{DD} is from 5.0 V to 2.5 V.

6.5.2 Features

- Four I/O modes:
  - Quasi-bidirectional
  - Push-Pull output
  - Open-Drain output
  - Input only with high impendence
- TTL/Schmitt trigger input selectable by Px_TYPE[7:0] in Px_MFP[23:16]
- I/O pin configured as interrupt source with edge/level setting
- I/O pin internal pull-up resistor enabled only in Quasi-bidirectional I/O mode
- Enabling pin interrupt function will also enable the pin wake-up function

6.5.3 Block Diagram

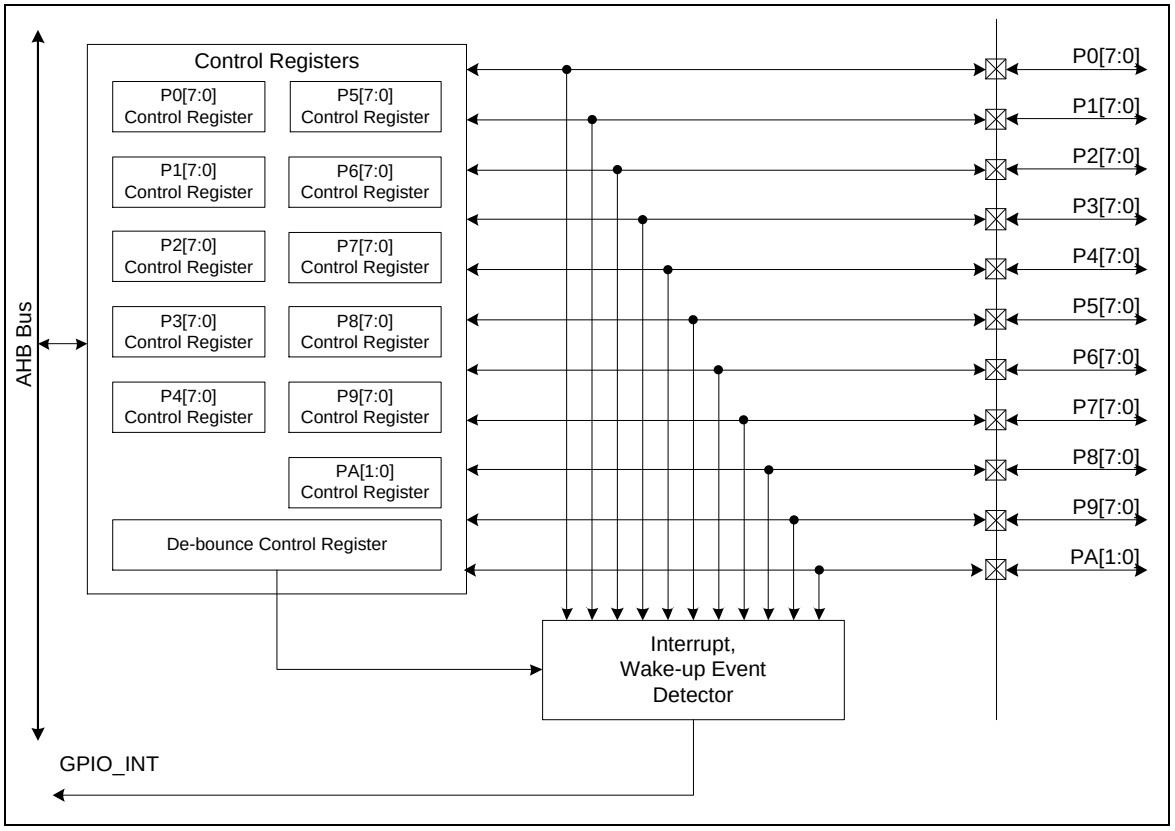


Figure 6–16GPIO Controller Block Diagram

Basic Configuration

The GPIO pin functions are configured in P0_MFP, P1_MFP, P2_MFP, P3_MFP, P4_MFP, P5_MFP, P6_MFP, P7_MFP, P8_MFP, P9_MFP and PA_MFP registers.

6.5.4 Functional Description

6.5.4.1 Input Only Mode

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

6.5.4.2 Push-Pull Output Mode

SettingPx_PMD (PMDn[1:0]) to 01b as the Px port [n]pin is in Push-pull Output mode and the I/O pin supports digital output function with source/sink current capability. The bit value in the corresponding bit [n] of Px_DOUT is driven on the pin.

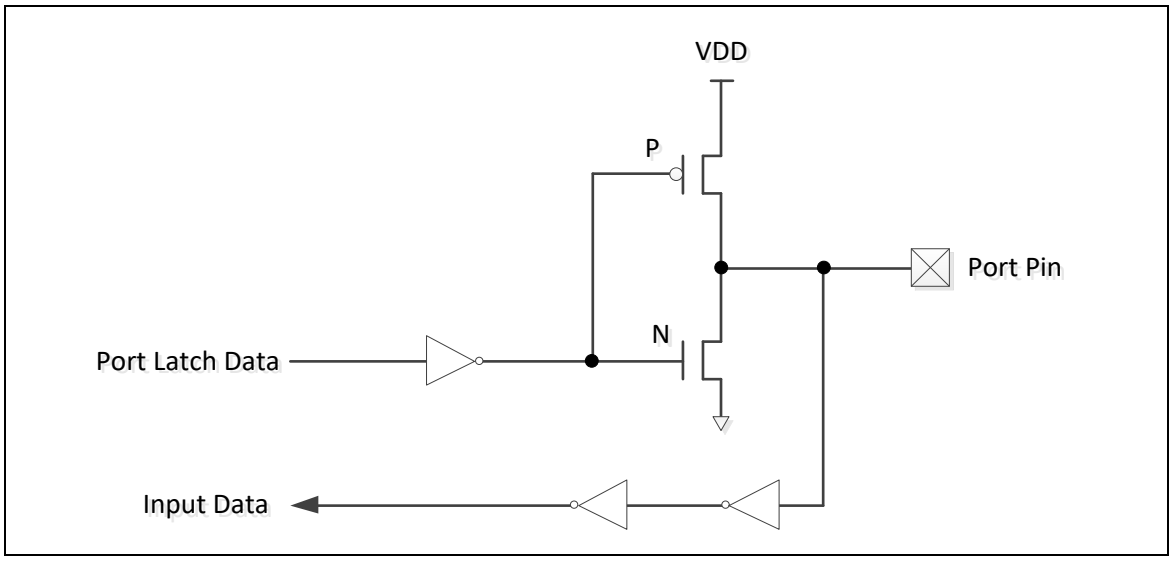


Figure 6–17 Push-Pull Output

6.5.4.3 Open-Drain Mode

SettingPx_PMD (PMDn[1:0]) to 10b as the Px 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 resister is needed for driving high state. If the bit value in the corresponding bit [n] of Px_DOUT is 0, the pin drive a “low” output on the pin. If the bit value in the corresponding bit [n] of Px_DOUT is 1, the pin output drives high that is controlled by external pull high resistor.

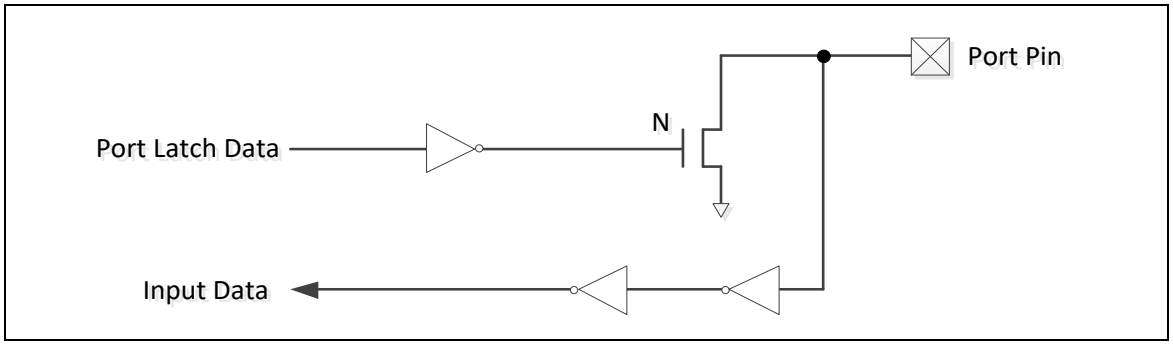


Figure 6–18 Open-Drain Output

6.5.4.4 Quasi Bi-directional Mode

SettingPx_PMD (PMDn[1:0]) to 11b as the Px 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 performedthe corresponding bit in

Px_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 Px_DOUT is 0, the pin drive a “low” output on the pin. If the bit value in the corresponding bit [n] of Px_DOUT is 1, the pin will check the pin value. If pin value is high, no action takes. If pin state is low, then pin will drive strong high with 2 clock cycles on the pin and then disable the strong output drive and then the pin status is control by internal pull-up resistor. Note that the source current capability in quasi-bidirectional mode is only about 200uA to 30uA if VDD is from 5.0V to 2.5V.

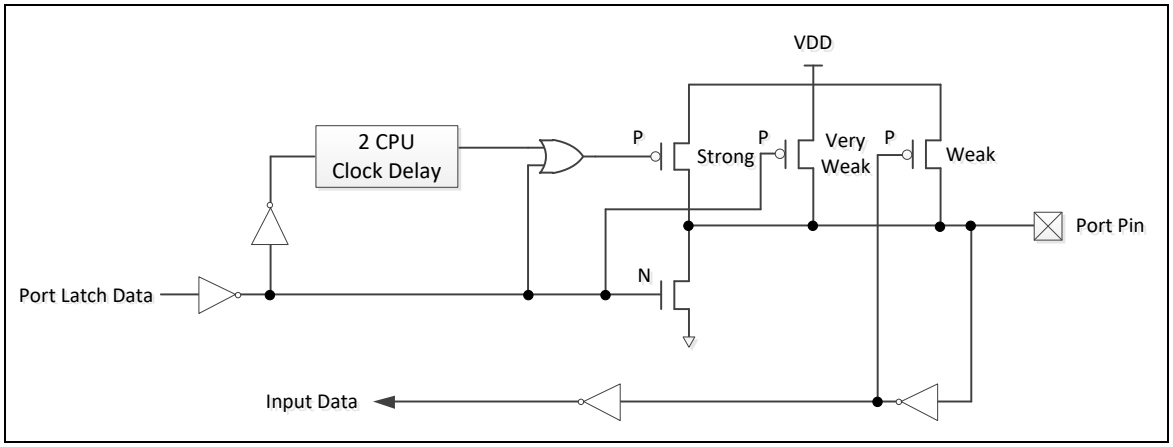


Figure 6–19 Quasi bi-directional I/O Mode

6.5.4.5 GPIO Interrupt and Wake-up Function

Each GPIO pin can be set as chip interrupt source by setting correlativePx_IEN bit and Px_IMD. There are five types of interrupt condition can be selected: low level trigger, high level trigger, falling edge trigger and rising edge trigger and both rising and falling edge trigger. For edge trigger condition, user can enable input signal de-bounce function to prevent unexpected interrupt happened which caused by noise. The de-bounce clock source and sampling cycle can be set through DEBOUNCE register.

The GPIO can also be the chip wake-up source when chip enters Idle mode or Power-down mode. The setting of wake-up trigger condition is the same as GPIO interrupt trigger, but there isone thing need to be noticed if using GPIO as chip wake-up source.

- **To ensure the I/O status before enter into Power-down mode**

When using toggle GPIO to wake-up system, user must make sure the I/O status before entering Idle mode or Power-down mode according to the relative wake-up settings.

For example, if configuring the wake-up event occurred by I/O rising edge/high level trigger, user must make sure the I/O status of specified pin is at low level before entering to Idle/Power-down mode; and if configure I/O falling edge/low level trigger to trigger a wake-up event, user must make sure the I/O status of specified pin is at high level before entering to Power-down mode.

6.5.5 PWM Port Output Driving Control

There are two enhanced PWM units each unit has six output pins in this device. The Enhanced PWM port outputs are P0.0~P0.5 and P1.0~P1.5 for unit 0 and unit 1, respectively.There are oneBasic PWM, It is has two output pins in this device.The BasicPWM port outputs are P5.6~P5.7.

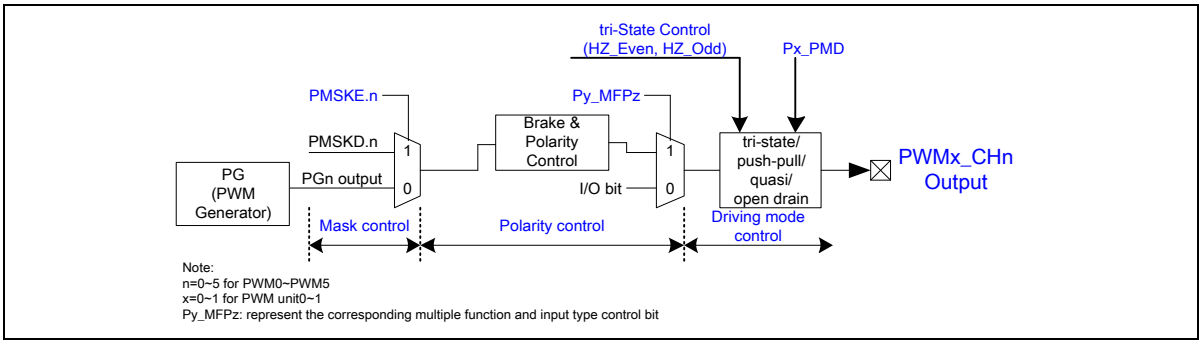


Figure 6–20Enhanced PWM Output Driving Control

The driving mode of PWM output ports can be initialized as tri-state type or other type dependent with Px_PMD (PMDn[1:0]) register setting after any reset. Figure 6-20 show enhanced PWM outputdriving control diagram, for HZ_Even0 (PWMPOEN[0])/HZ_Odd0 (PWMPOEN[1])/HZ_Even1 (PWMPOEN[2])/HZ_Odd1 (PWMPOEN[3])/HZ_BPWM (PWMPOEN[3]), the initial value is defined by CHZ_Even0 (Config0[8])/ CHZ_Odd0 (Config0[9])/ CHZ_Even1 (Config0[10])/ CHZ_Odd1 (Config0[11])/CHZ_BPWM (Config0[12]), respectively..

If each of register bit set to 0,the driving mode ofPWM ports are controlled by GPIO mode registeror multi-function register, If each of register bit set to 1,the driving mode of PWM ports are forced in tri-state all the time.

HZ_Even/HZ_Odd/HZ_BPWM(In PWMPOEN Register)	PWM Outputs Drive Mode
0	Depend on Px_PMD or Px_MFP
1	Force in tri-state

Table 6-8PWM Outputs Drive Mode Setting Table

6.5.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
GPIO Base Address: **GPIO_BA = 0x5000_4000**				
P0_PMD	GPIO_BA+0x000	R/W	GPIO Port 0Pin I/O Mode Control	0x0000_0000
P0_OFFD	GPIO_BA+0x004	R/W	GPIO Port 0 Pin Digital Input Path Disable Control	0x0000_0000
P0_DOUT	GPIO_BA+0x008	R/W	GPIO Port 0 Data Output Value	0x0000_00FF
P0_DMASK	GPIO_BA+0x00C	R/W	GPIO Port 0 Data Output Write Mask	0x0000_0000
P0_PIN	GPIO_BA+0x010	R	GPIO Port 0 Pin Value	0x0000_00XX
P0_DBEN	GPIO_BA+0x014	R/W	GPIO Port 0 De-bounce Enable	0x0000_0000
P0_IMD	GPIO_BA+0x018	R/W	GPIO Port 0 Interrupt Mode Control	0x0000_0000
P0_IEN	GPIO_BA+0x01C	R/W	GPIO Port 0 Interrupt Enable	0x0000_0000
P0_ISF	GPIO_BA+0x020	R/W	GPIO Port 0 Interrupt Source Flag	0xXXXX_XXXX
P1_PMD	GPIO_BA+0x040	R/W	GPIO Port 1Pin I/O Mode Control	0x0000_0000
P1_OFFD	GPIO_BA+0x044	R/W	GPIO Port 1 Pin Digital Input Path Disable Control	0x0000_0000
P1_DOUT	GPIO_BA+0x048	R/W	GPIO Port 1 Data Output Value	0x0000_00FF
P1_DMASK	GPIO_BA+0x04C	R/W	GPIO Port 1 Data Output Write Mask	0x0000_0000
P1_PIN	GPIO_BA+0x050	R	GPIO Port 1 Pin Value	0x0000_00XX
P1_DBEN	GPIO_BA+0x054	R/W	GPIO Port 1 De-bounce Enable	0x0000_0000
P1_IMD	GPIO_BA+0x058	R/W	GPIO Port 1 Interrupt Mode Control	0x0000_0000
P1_IEN	GPIO_BA+0x05C	R/W	GPIO Port 1 Interrupt Enable	0x0000_0000
P1_ISF	GPIO_BA+0x060	R/W	GPIO Port 1 Interrupt Source Flag	0xXXXX_XXXX
P2_PMD	GPIO_BA+0x080	R/W	GPIO Port 2Pin I/O Mode Control	0x0000_0000
P2_OFFD	GPIO_BA+0x084	R/W	GPIO Port 2 Pin Digital Input Path Disable Control	0x0000_0000
P2_DOUT	GPIO_BA+0x088	R/W	GPIO Port 2 Data Output Value	0x0000_00FF
P2_DMASK	GPIO_BA+0x08C	R/W	GPIO Port 2 Data Output Write Mask	0x0000_0000
P2_PIN	GPIO_BA+0x090	R	GPIO Port 2 Pin Value	0x0000_00XX
P2_DBEN	GPIO_BA+0x094	R/W	GPIO Port 2 De-bounce Enable	0x0000_0000
P2_IMD	GPIO_BA+0x098	R/W	GPIO Port 2 Interrupt Mode Control	0x0000_0000
P2_IEN	GPIO_BA+0x09C	R/W	GPIO Port 2 Interrupt Enable	0x0000_0000

P2_ISF	GPIO_BA+0x0A0	R/W	GPIO Port 2 Interrupt Source Flag	0xXXXX_XXXX
P3_PMD	GPIO_BA+0x0C0	R/W	GPIO Port 3Pin I/O Mode Control	0x0000_0000
P3_OFFD	GPIO_BA+0x0C4	R/W	GPIO Port 3 Pin Digital Input Path Disable Control	0x0000_0000
P3_DOUT	GPIO_BA+0x0C8	R/W	GPIO Port 3 Data Output Value	0x0000_00FF
P3_DMASK	GPIO_BA+0x0CC	R/W	GPIO Port 3 Data Output Write Mask	0x0000_0000
P3_PIN	GPIO_BA+0x0D0	R	GPIO Port 3 Pin Value	0x0000_00XX
P3_DBEN	GPIO_BA+0x0D4	R/W	GPIO Port 3 De-bounce Enable	0x0000_0000
P3_IMD	GPIO_BA+0x0D8	R/W	GPIO Port 3 Interrupt Mode Control	0x0000_0000
P3_IEN	GPIO_BA+0x0DC	R/W	GPIO Port 3 Interrupt Enable	0x0000_0000
P3_ISF	GPIO_BA+0x0E0	R/W	GPIO Port 3 Interrupt Source Flag	0xXXXX_XXXX
P4_PMD	GPIO_BA+0x100	R/W	GPIO Port 4Pin I/O Mode Control	0x0000_0000
P4_OFFD	GPIO_BA+0x104	R/W	GPIO Port 4 Pin Digital Input Path Disable Control	0x0000_0000
P4_DOUT	GPIO_BA+0x108	R/W	GPIO Port 4 Data Output Value	0x0000_00FF
P4_DMASK	GPIO_BA+0x10C	R/W	GPIO Port 4 Data Output Write Mask	0x0000_0000
P4_PIN	GPIO_BA+0x110	R	GPIO Port 4 Pin Value	0x0000_00XX
P4_DBEN	GPIO_BA+0x114	R/W	GPIO Port 4 De-bounce Enable	0x0000_0000
P4_IMD	GPIO_BA+0x118	R/W	GPIO Port 4 Interrupt Mode Control	0x0000_0000
P4_IEN	GPIO_BA+0x11C	R/W	GPIO Port 4 Interrupt Enable	0x0000_0000
P4_ISF	GPIO_BA+0x120	R/W	GPIO Port 4 Interrupt Source Flag	0xXXXX_XXXX
P5_PMD	GPIO_BA+0x140	R/W	GPIO Port 5Pin I/O Mode Control	0x0000_0000
P5_OFFD	GPIO_BA+0x144	R/W	GPIO Port 5 Pin Digital Input Path Disable Control	0x0000_0000
P5_DOUT	GPIO_BA+0x148	R/W	GPIO Port 5 Data Output Value	0x0000_00FF
P5_DMASK	GPIO_BA+0x14C	R/W	GPIO Port 5 Data Output Write Mask	0x0000_0000
P5_PIN	GPIO_BA+0x150	R	GPIO Port 5 Pin Value	0x0000_00XX
P5_DBEN	GPIO_BA+0x154	R/W	GPIO Port 5 De-bounce Enable	0x0000_0000
P5_IMD	GPIO_BA+0x158	R/W	GPIO Port 5 Interrupt Mode Control	0x0000_0000
P5_IEN	GPIO_BA+0x15C	R/W	GPIO Port 5 Interrupt Enable	0x0000_0000
P5_ISF	GPIO_BA+0x160	R/W	GPIO Port 5 Interrupt Source Flag	0xXXXX_XXXX
P6_PMD	GPIO_BA+0x180	R/W	GPIO Port 6Pin I/O Mode Control	0x0000_0000
P6_OFFD	GPIO_BA+0x184	R/W	GPIO Port 6 Pin Digital Input Path Disable Control	0x0000_0000
P6_DOUT	GPIO_BA+0x188	R/W	GPIO Port 6 Data Output Value	0x0000_00FF

P6_DMASK	GPIO_BA+0x18C	R/W	GPIO Port 6 Data Output Write Mask	0x0000_0000
P6_PIN	GPIO_BA+0x190	R	GPIO Port 6Pin Value	0x0000_00XX
P6_DBEN	GPIO_BA+0x194	R/W	GPIO Port 6De-bounce Enable	0x0000_0000
P6_IMD	GPIO_BA+0x198	R/W	GPIO Port 6 Interrupt Mode Control	0x0000_0000
P6_IEN	GPIO_BA+0x19C	R/W	GPIO Port 6Interrupt Enable	0x0000_0000
P6_ISF	GPIO_BA+0x200	R/W	GPIO Port 6Interrupt Source Flag	0xXXXX_XXXX
P7_PMD	GPIO_BA+0x1C0	R/W	GPIO Port 7Pin I/O Mode Control	0x0000_0000
P7_OFFD	GPIO_BA+0x1C4	R/W	GPIO Port 7 Pin Digital Input Path Disable Control	0x0000_0000
P7_DOUT	GPIO_BA+0x1C8	R/W	GPIO Port 7 Data Output Value	0x0000_00FF
P7_DMASK	GPIO_BA+0x1CC	R/W	GPIO Port 7 Data Output Write Mask	0x0000_0000
P7_PIN	GPIO_BA+0x1D0	R	GPIO Port 7 Pin Value	0x0000_00XX
P7_DBEN	GPIO_BA+0x1D4	R/W	GPIO Port 7 De-bounce Enable	0x0000_0000
P7_IMD	GPIO_BA+0x1D8	R/W	GPIO Port 7 Interrupt Mode Control	0x0000_0000
P7_IEN	GPIO_BA+0x1DC	R/W	GPIO Port 7 Interrupt Enable	0x0000_0000
P7_ISF	GPIO_BA+0x1E0	R/W	GPIO Port 7 Interrupt Source Flag	0xXXXX_XXXX
P8_PMD	GPIO_BA+0x200	R/W	GPIO Port 8Pin I/O Mode Control	0x0000_0000
P8_OFFD	GPIO_BA+0x204	R/W	GPIO Port 8 Pin Digital Input Path Disable Control	0x0000_0000
P8_DOUT	GPIO_BA+0x208	R/W	GPIO Port 8 Data Output Value	0x0000_00FF
P8_DMASK	GPIO_BA+0x20C	R/W	GPIO Port 8 Data Output Write Mask	0x0000_0000
P8_PIN	GPIO_BA+0x210	R	GPIO Port 8 Pin Value	0x0000_00XX
P8_DBEN	GPIO_BA+0x214	R/W	GPIO Port 8 De-bounce Enable	0x0000_0000
P8_IMD	GPIO_BA+0x218	R/W	GPIO Port 8 Interrupt Mode Control	0x0000_0000
P8_IEN	GPIO_BA+0x21C	R/W	GPIO Port 8 Interrupt Enable	0x0000_0000
P8_ISF	GPIO_BA+0x220	R/W	GPIO Port 8 Interrupt Source Flag	0xXXXX_XXXX
P9_PMD	GPIO_BA+0x240	R/W	GPIO Port 9Pin I/O Mode Control	0x0000_0000
P9_OFFD	GPIO_BA+0x244	R/W	GPIO Port 9 Pin Digital Input Path Disable Control	0x0000_0000
P9_DOUT	GPIO_BA+0x248	R/W	GPIO Port 9 Data Output Value	0x0000_00FF
P9_DMASK	GPIO_BA+0x24C	R/W	GPIO Port 9 Data Output Write Mask	0x0000_0000
P9_PIN	GPIO_BA+0x250	R	GPIO Port 9 Pin Value	0x0000_00XX
P9_DBEN	GPIO_BA+0x254	R/W	GPIO Port 9 De-bounce Enable	0x0000_0000
P9_IMD	GPIO_BA+0x258	R/W	GPIO Port 9 Interrupt Mode Control	0x0000_0000

P9_IEN	GPIO_BA+0x25C	R/W	GPIO Port 9 Interrupt Enable	0x0000_0000
P9_ISF	GPIO_BA+0x260	R/W	GPIO Port 9 Interrupt Source Flag	0xXXXX_XXXX
PA_PMD	GPIO_BA+0x280	R/W	GPIO Port Apin I/O Mode Control	0x0000_0000
PA_OFFD	GPIO_BA+0x284	R/W	GPIO Port A Pin Digital Input Path Disable Control	0x0000_0000
PA_DOUT	GPIO_BA+0x288	R/W	GPIO Port A Data Output Value	0x0000_0003
PA_DMASK	GPIO_BA+0x28C	R/W	GPIO Port A Data Output Write Mask	0x0000_0000
PA_PIN	GPIO_BA+0x290	R	GPIO Port Apin Value	0x0000_000X
PA_DBEN	GPIO_BA+0x294	R/W	GPIO Port Ade-bounce Enable	0x0000_0000
PA_IMD	GPIO_BA+0x298	R/W	GPIO Port A Interrupt Mode Control	0x0000_0000
PA_IEN	GPIO_BA+0x29C	R/W	GPIO Port Ainterrupt Enable	0x0000_0000
PA_ISF	GPIO_BA+0x2A0	R/W	GPIO Port Ainterrupt Source Flag	0xXXXX_XXXX
DBNCECON	GPIO_BA+0x2E0	R/W	External Interrupt De-bounce Control	0x0000_0000
PWMPOEN	GPIO_BA+0x2E4	R/W	PWM Port Output Enable	0x0000_00XX
P0_N **n=0,1..7**	GPIO_BA+0x300 + 0x04 * n	R/W	GPIO P0.n PinData Input/Output	0x0000_000X
P1_N **n=0,1..7**	GPIO_BA+0x320 + 0x04 * n	R/W	GPIO P1.n PinData Input/Output	0x0000_000X
P2_N **n=0,1..7**	GPIO_BA+0x340 + 0x04 * n	R/W	GPIO P2.n PinData Input/Output	0x0000_000X
P3_N **n=0,1..7**	GPIO_BA+0x360 + 0x04 * n	R/W	GPIO P3.n PinData Input/Output	0x0000_000X
P4_N **n=0,1..7**	GPIO_BA+0x380 + 0x04 * n	R/W	GPIO P4.n PinData Input/Output	0x0000_000X
P5_N **n=0,1..7**	GPIO_BA+0x3A0 + 0x04 * n	R/W	GPIO P5.n PinData Input/Output	0x0000_000X
P6_N **n=0,1..7**	GPIO_BA+0x3C0 + 0x04 * n	R/W	GPIO P6.n PinData Input/Output	0x0000_000X
P7_N **n=0,1..7**	GPIO_BA+0x3E0 + 0x04 * n	R/W	GPIO P7.n PinData Input/Output	0x0000_000X
P8_N **n=0,1..7**	GPIO_BA+0x400 + 0x04 * n	R/W	GPIO P8.n PinData Input/Output	0x0000_000X
P9_N **n=0,1..7**	GPIO_BA+0x420 + 0x04 * n	R/W	GPIO P9.n PinData Input/Output	0x0000_000X
PA_N **n=0,1**	GPIO_BA+0x440 + 0x04 * n	R/W	GPIO PA.n PinData Input/Output	0x0000_000X

6.5.7 Register Description

GPIO Port [0/1/2/3/4/5/6/7/8/9/A] Pin I/O Mode Control (Px_PMD)

Register	Offset	R/W	Description	Reset Value
P0_PMD	GPIO_BA+0x000	R/W	GPIO Port 0 Pin I/O Mode Control	0x0000_0000
P1_PMD	GPIO_BA+0x040	R/W	GPIO Port 1 Pin I/O Mode Control	0x0000_0000
P2_PMD	GPIO_BA+0x080	R/W	GPIO Port 2 Pin I/O Mode Control	0x0000_0000
P3_PMD	GPIO_BA+0x0C0	R/W	GPIO Port 3 Pin I/O Mode Control	0x0000_0000
P4_PMD	GPIO_BA+0x100	R/W	GPIO Port 4 Pin I/O Mode Control	0x0000_0000
P5_PMD	GPIO_BA+0x140	R/W	GPIO Port 5 Pin I/O Mode Control	0x0000_0000
P6_PMD	GPIO_BA+0x180	R/W	GPIO Port 6 Pin I/O Mode Control	0x0000_0000
P7_PMD	GPIO_BA+0x1C0	R/W	GPIO Port 7 Pin I/O Mode Control	0x0000_0000
P8_PMD	GPIO_BA+0x200	R/W	GPIO Port 8 Pin I/O Mode Control	0x0000_0000
P9_PMD	GPIO_BA+0x240	R/W	GPIO Port 9 Pin I/O Mode Control	0x0000_0000
PA_PMD	GPIO_BA+0x280	R/W	GPIO Port A Pin I/O 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**
PMD7		**PMD6**		**PMD5**		**PMD4**	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PMD3		**PMD2**		**PMD1**		**PMD0**	

Bits	Description	
[31:16]	**Reserved**	Reserved.
[2n+1:2n] n=0,1..7	**PMDn**	**Port I/O Pin[N] Mode Control** Determine each I/O mode of Px pins. 00 = GPIO port [n] pin is in Input mode. 01 = GPIO port [n] pin is in Push-pull Output mode. 10 = GPIO port [n] pin is in Open-drain Output mode. 11 = GPIO port [n] pin is in Quasi-bidirectional mode. **Note:** Max. n = 1 for PA; Max. n = 7 for P0/P1/P2/P3/P4/P5/P6/P7/P8/P9.

GPIO Port [0/1/2/3/4/5/6/7/8/9/A] Pin Digital Input Path Disable Control (Px_OFFD)

Register	Offset	R/W	Description	Reset Value
P0_OFFD	GPIO_BA+0x004	R/W	GPIO Port 0 Pin Digital Input Path Disable Control	0x0000_0000
P1_OFFD	GPIO_BA+0x044	R/W	GPIO Port 1 Pin Digital Input Path Disable Control	0x0000_0000
P2_OFFD	GPIO_BA+0x084	R/W	GPIO Port 2 Pin Digital Input Path Disable Control	0x0000_0000
P3_OFFD	GPIO_BA+0x0C4	R/W	GPIO Port 3 Pin Digital Input Path Disable Control	0x0000_0000
P4_OFFD	GPIO_BA+0x104	R/W	GPIO Port 4 Pin Digital Input Path Disable Control	0x0000_0000
P5_OFFD	GPIO_BA+0x144	R/W	GPIO Port 5 Pin Digital Input Path Disable Control	0x0000_0000
P6_OFFD	GPIO_BA+0x184	R/W	GPIO Port 6 Pin Digital Input Path Disable Control	0x0000_0000
P7_OFFD	GPIO_BA+0x1C4	R/W	GPIO Port 7 Pin Digital Input Path Disable Control	0x0000_0000
P8_OFFD	GPIO_BA+0x204	R/W	GPIO Port 8 Pin Digital Input Path Disable Control	0x0000_0000
P9_OFFD	GPIO_BA+0x244	R/W	GPIO Port 9 Pin Digital Input Path Disable Control	0x0000_0000
PA_OFFD	GPIO_BA+0x284	R/W	GPIO Port A Pin Digital Input Path Disable Control	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
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	
[31:24]	**Reserved**	Reserved.
[23:16]	**OFFD**	**Port Pin[N] Digital Input Path Disable Control** Each of these bits is used to control if the digital input path of corresponding GPIO pin is disabled. If input is analog signal, users can disable GPIO digital input path to avoid input current leakage. 0 = I/O digital input path Enabled. 1 = I/O digital input path Disabled (digital input tied to low). **Note:** Max. n = 1 for PA; Max. n = 7 for P0/P1/P2/P3/P4/P5/P6/P7/P8/P9.
[15:0]	**Reserved**	Reserved.

GPIO Port [0/1/2/3/4/5/6/7/8/9/A] Data Output Value (Px_DOUT)

Register	Offset	R/W	Description	Reset Value
P0_DOUT	GPIO_BA+0x008	R/W	GPIO Port 0 Data Output Value	0x0000_00FF
P1_DOUT	GPIO_BA+0x048	R/W	GPIO Port 1 Data Output Value	0x0000_00FF
P2_DOUT	GPIO_BA+0x088	R/W	GPIO Port 2 Data Output Value	0x0000_00FF
P3_DOUT	GPIO_BA+0x0C8	R/W	GPIO Port 3 Data Output Value	0x0000_00FF
P4_DOUT	GPIO_BA+0x108	R/W	GPIO Port 4 Data Output Value	0x0000_00FF
P5_DOUT	GPIO_BA+0x148	R/W	GPIO Port 5 Data Output Value	0x0000_00FF
P6_DOUT	GPIO_BA+0x188	R/W	GPIO Port 6 Data Output Value	0x0000_00FF
P7_DOUT	GPIO_BA+0x1C8	R/W	GPIO Port 7 Data Output Value	0x0000_00FF
P8_DOUT	GPIO_BA+0x208	R/W	GPIO Port 8 Data Output Value	0x0000_00FF
P9_DOUT	GPIO_BA+0x248	R/W	GPIO Port 9 Data Output Value	0x0000_00FF
PA_DOUT	GPIO_BA+0x288	R/W	GPIO Port A Data Output Value	0x0000_0003

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**DOUT**	**Port Pin[N] Output Value** Each of these bits controls the status of a GPIO pin when the GPIO is configured as Push-pull output, Open-drain output or Quasi-bidirectional mode. 0 = GPIO port [0/1/2/3/4/5/6/7/8/9/A] Pin[n] will drive Low if the GPIO pin is configuredas Push-pull output, Open-drain output orQuasi-bidirectional mode. 1 = GPIO port [0/1/2/3/4/5/6/7/8/9/A] Pin[n] will drive High if the GPIO pin is configured as Push-pull output orQuasi-bidirectional mode. **Note:** Max. n = 1 for PA; Max. n = 7 for P0/P1/P2/P3/P4/P5/P6/P7/P8/P9.

GPIO Port [0/1/2/3/4/5/6/7/8/9/A] Data Output Write Mask (Px _DMASK)

Register	Offset	R/W	Description	Reset Value
P0_DMASK	GPIO_BA+0x00C	R/W	GPIO Port 0 Data Output Write Mask	0x0000_0000
P1_DMASK	GPIO_BA+0x04C	R/W	GPIO Port 1 Data Output Write Mask	0x0000_0000
P2_DMASK	GPIO_BA+0x08C	R/W	GPIO Port 2 Data Output Write Mask	0x0000_0000
P3_DMASK	GPIO_BA+0x0CC	R/W	GPIO Port 3 Data Output Write Mask	0x0000_0000
P4_DMASK	GPIO_BA+0x10C	R/W	GPIO Port 4 Data Output Write Mask	0x0000_0000
P5_DMASK	GPIO_BA+0x14C	R/W	GPIO Port 5 Data Output Write Mask	0x0000_0000
P6_DMASK	GPIO_BA+0x18C	R/W	GPIO Port 6 Data Output Write Mask	0x0000_0000
P7_DMASK	GPIO_BA+0x1CC	R/W	GPIO Port 7 Data Output Write Mask	0x0000_0000
P8_DMASK	GPIO_BA+0x20C	R/W	GPIO Port 8 Data Output Write Mask	0x0000_0000
P9_DMASK	GPIO_BA+0x24C	R/W	GPIO Port 9 Data Output Write Mask	0x0000_0000
PA_DMASK	GPIO_BA+0x28C	R/W	GPIO Port A 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**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DMASK							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**DMASK**	**Port Data Output Write Mask** These bits are used to protect the corresponding register of Px_DOUT bit[n]. When set the DMASK bit[n] is set to 1, the corresponding Px_DOUT[n] bit is protected. If the write signal is masked, write data to the protect bit is ignored. 0 = Corresponding Px_DOUT[n] bit can be updated. 1 = Corresponding Px_DOUT[n] bit protected. **Note1:**This function only protects the corresponding Px_DOUT[n] bit, and will not protect the corresponding bit control register (P0_n, P1_n, P2_n, P3_n, P4_n, P5_n, P6_n, P7_n, P8_n, P9_n and PA_n). **Note2:** Max. n = 1 for PA; Max. n = 7 for P0/P1/P2/P3/P4/P5/P6/P7/P8/P9.

GPIO Port [0/1/2/3/4/5/6/7/8/9/A] Pin Value (Px _PIN)

Register	Offset	R/W	Description	Reset Value
P0_PIN	GPIO_BA+0x010	R	GPIO Port 0 Pin Value	0x0000_00XX
P1_PIN	GPIO_BA+0x050	R	GPIO Port 1 Pin Value	0x0000_00XX
P2_PIN	GPIO_BA+0x090	R	GPIO Port 2 Pin Value	0x0000_00XX
P3_PIN	GPIO_BA+0x0D0	R	GPIO Port 3 Pin Value	0x0000_00XX
P4_PIN	GPIO_BA+0x110	R	GPIO Port 4 Pin Value	0x0000_00XX
P5_PIN	GPIO_BA+0x150	R	GPIO Port 5 Pin Value	0x0000_00XX
P6_PIN	GPIO_BA+0x190	R	GPIO Port 6 Pin Value	0x0000_00XX
P7_PIN	GPIO_BA+0x1D0	R	GPIO Port 7 Pin Value	0x0000_00XX
P8_PIN	GPIO_BA+0x210	R	GPIO Port 8 Pin Value	0x0000_00XX
P9_PIN	GPIO_BA+0x250	R	GPIO Port 9 Pin Value	0x0000_00XX
PA_PIN	GPIO_BA+0x290	R	GPIO Port A 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**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PIN							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**PIN**	**Port Pin Values** Each bit of the register reflects the actual status of the respective GPIO pin. If the bit is 1, it indicates the corresponding pin status is high, else the pin status is low. **Note:** Max. n = 1 for PA; Max. n = 7 for P0/P1/P2/P3/P4/P5/P6/P7/P8/P9.

GPIO Port [0/1/2/3/4/5/6/7/8/9/A] De-bounce Enable (Px _DBEN)

Register	Offset	R/W	Description	Reset Value
P0_DBEN	GPIO_BA+0x014	R/W	GPIO Port 0 De-bounce Enable	0x0000_0000
P1_DBEN	GPIO_BA+0x054	R/W	GPIO Port 1 De-bounce Enable	0x0000_0000
P2_DBEN	GPIO_BA+0x094	R/W	GPIO Port 2 De-bounce Enable	0x0000_0000
P3_DBEN	GPIO_BA+0x0D4	R/W	GPIO Port 3 De-bounce Enable	0x0000_0000
P4_DBEN	GPIO_BA+0x114	R/W	GPIO Port 4 De-bounce Enable	0x0000_0000
P5_DBEN	GPIO_BA+0x154	R/W	GPIO Port 5 De-bounce Enable	0x0000_0000
P6_DBEN	GPIO_BA+0x194	R/W	GPIO Port 6 De-bounce Enable	0x0000_0000
P7_DBEN	GPIO_BA+0x1D4	R/W	GPIO Port 7 De-bounce Enable	0x0000_0000
P8_DBEN	GPIO_BA+0x214	R/W	GPIO Port 8 De-bounce Enable	0x0000_0000
P9_DBEN	GPIO_BA+0x254	R/W	GPIO Port 9 De-bounce Enable	0x0000_0000
PA_DBEN	GPIO_BA+0x294	R/W	GPIO Port A 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**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DBEN							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**DBEN**	**Port Input Signal De-bounce Enable Bits** DBEN[n] is used to enable the de-bounce function for each corresponding bit. If the input signal pulse width cannot be sampled by continuous two de-bounce sample cycle,the input signal transition is seen as the signal bounce and will not trigger the interrupt. The de-bounce clock source is controlled by DBCLKSRC (DBNCECON[4]), one de-bounce sample cycle period is controlled by DBCLKSEL (DBNCECON[3:0]). 0 = Bit[n] de-bounce function Disabled. 1 = Bit[n] de-bounce function Enabled. **Note:** Max. n = 1 for PA; Max. n = 7 for P0/P1/P2/P3/P4/P5/P6/P7/P8/P9.

GPIO Port [0/1/2/3/4/5/6/7/8/9/A] Interrupt Mode Control (Px _IMD)

Register	Offset	R/W	Description	Reset Value
P0_IMD	GPIO_BA+0x018	R/W	GPIO Port 0 Interrupt Mode Control	0x0000_0000
P1_IMD	GPIO_BA+0x058	R/W	GPIO Port 1 Interrupt Mode Control	0x0000_0000
P2_IMD	GPIO_BA+0x098	R/W	GPIO Port 2 Interrupt Mode Control	0x0000_0000
P3_IMD	GPIO_BA+0x0D8	R/W	GPIO Port 3 Interrupt Mode Control	0x0000_0000
P4_IMD	GPIO_BA+0x118	R/W	GPIO Port 4 Interrupt Mode Control	0x0000_0000
P5_IMD	GPIO_BA+0x158	R/W	GPIO Port 5 Interrupt Mode Control	0x0000_0000
P6_IMD	GPIO_BA+0x198	R/W	GPIO Port 6 Interrupt Mode Control	0x0000_0000
P7_IMD	GPIO_BA+0x1D8	R/W	GPIO Port 7 Interrupt Mode Control	0x0000_0000
P8_IMD	GPIO_BA+0x218	R/W	GPIO Port 8 Interrupt Mode Control	0x0000_0000
P9_IMD	GPIO_BA+0x258	R/W	GPIO Port 9 Interrupt Mode Control	0x0000_0000
PA_IMD	GPIO_BA+0x298	R/W	GPIO Port A 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**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
IMD							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**IMD**	**Port Edge or Level Detection Interrupt Control** IMD[n] decides the pin interrupt triggered by level or edge. If the interrupt is by edge trigger, the trigger source can be controlled by de-bounce. If the interrupt is by level trigger, the input source is sampled by one HCLK clock and generates the interrupt. 0 = Edge triggered interrupt. 1 = Level triggered interrupt. If set pin as the level trigger interrupt, then only one level can be set on the registers Px_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:** Max. n = 1 for PA; Max. n = 7 for P0/P1/P2/P3/P4/P5/P6/P7/P8/P9.

GPIO Port [0/1/2/3/4/5/6/7/8/9/A] Interrupt Enable Control (Px _IEN)

Register	Offset	R/W	Description	Reset Value
P0_IEN	GPIO_BA+0x01C	R/W	GPIO Port 0 Interrupt Enable	0x0000_0000
P1_IEN	GPIO_BA+0x05C	R/W	GPIO Port 1 Interrupt Enable	0x0000_0000
P2_IEN	GPIO_BA+0x09C	R/W	GPIO Port 2 Interrupt Enable	0x0000_0000
P3_IEN	GPIO_BA+0x0DC	R/W	GPIO Port 3 Interrupt Enable	0x0000_0000
P4_IEN	GPIO_BA+0x11C	R/W	GPIO Port 4 Interrupt Enable	0x0000_0000
P5_IEN	GPIO_BA+0x15C	R/W	GPIO Port 5 Interrupt Enable	0x0000_0000
P6_IEN	GPIO_BA+0x19C	R/W	GPIO Port 6 Interrupt Enable	0x0000_0000
P7_IEN	GPIO_BA+0x1DC	R/W	GPIO Port 7 Interrupt Enable	0x0000_0000
P8_IEN	GPIO_BA+0x21C	R/W	GPIO Port 8 Interrupt Enable	0x0000_0000
P9_IEN	GPIO_BA+0x25C	R/W	GPIO Port 9 Interrupt Enable	0x0000_0000
PA_IEN	GPIO_BA+0x29C	R/W	GPIO Port A Interrupt Enable	0x0000_0000

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

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:16]	**IR_EN**	**Port Interrupt Enable by Input Rising Edge or Input Level High** IR_EN[n] used to enable the interrupt for each of the corresponding input Px_PIN[n]. Set bit to 1 also enable the pin wake-up function. When setting the IR_EN[n] bit to 1: If the interrupt is level trigger, the input PIN[n] state at level “high” will generate the interrupt. If the interrupt is edge trigger, the input PIN[n] state change from “low-to-high” will generate the interrupt. 0 = PIN[n] level-high or low-to-high interruptDisabled. 1 = PIN[n] level-high or low-to-high interruptEnabled. **Note:** Max. n = 1 for PA; Max. n = 7 for P0/P1/P2/P3/P4/P5/P6/P7/P8/P9.
[15:8]	**Reserved**	Reserved.

[7:0]	**IF_EN**	**Port 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 Px_PIN[n]. Set bit to 1 also enable the pin wake-up function. When setting the IF_EN[n] bit to 1: If the interrupt is level trigger, the input PIN[n] state at level “low” will generate the interrupt. If the interrupt is edge trigger, the input PIN[n] state change from “high-to-low” will generate the interrupt. 0 = PIN[n] state low-level or high-to-low change interrupt Disabled. 1 = PIN[n] state low-level or high-to-low change interrupt Enabled. **Note:** Max. n = 1 for PA; Max. n = 7 for P0/P1/P2/P3/P4/P5/P6/P7/P8/P9.

GPIO Port [0/1/2/3/4/5/6/7/8/9/A] Interrupt Source Flag (Px _ISF)

Register	Offset	R/W	Description	Reset Value
P0_ISF	GPIO_BA+0x020	R/W	GPIO Port 0 Interrupt Source Flag	0xXXXX_XXXX
P1_ISF	GPIO_BA+0x060	R/W	GPIO Port 1 Interrupt Source Flag	0xXXXX_XXXX
P2_ISF	GPIO_BA+0x0A0	R/W	GPIO Port 2 Interrupt Source Flag	0xXXXX_XXXX
P3_ISF	GPIO_BA+0x0E0	R/W	GPIO Port 3 Interrupt Source Flag	0xXXXX_XXXX
P4_ISF	GPIO_BA+0x120	R/W	GPIO Port 4 Interrupt Source Flag	0xXXXX_XXXX
P5_ISF	GPIO_BA+0x160	R/W	GPIO Port 5 Interrupt Source Flag	0xXXXX_XXXX
P6_ISF	GPIO_BA+0x1A0	R/W	GPIO Port 6 Interrupt Source Flag	0xXXXX_XXXX
P7_ISF	GPIO_BA+0x1E0	R/W	GPIO Port 7 Interrupt Source Flag	0xXXXX_XXXX
P8_ISF	GPIO_BA+0x220	R/W	GPIO Port 8 Interrupt Source Flag	0xXXXX_XXXX
P9_ISF	GPIO_BA+0x260	R/W	GPIO Port 9 Interrupt Source Flag	0xXXXX_XXXX
PA_ISF	GPIO_BA+0x2A0	R/W	GPIO Port A Interrupt Source Flag	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**
IF_ISF							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**IF_ISF**	**Port Interrupt Source Flag** These bits are edge/level trigger event indicators of each pin of Port x, if GPG0_INT or GPG1_INT enable bit is set, the corresponding interrupt service routine will be served. Note that P3.2 and P3.3 will only vector to INT0_INT and INT1_INT ISRs if its own interrupt is enabled. Read : 0 = No interrupt at Px[n]. 1 = Px[n] generates an interrupt. Write : 0= No action. 1= Clear the corresponding pending interrupt. **Note:** Max. n = 1 for PA; Max. n = 7 for P0/P1/P2/P3/P4/P5/P6/P7/P8/P9.

External Interrupt De-bounce Control (DBNCECON)

Register	Offset	R/W	Description	Reset Value
DBNCECON	GPIO_BA+0x2E0	R/W	External Interrupt De-bounce 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
Reserved							
7	6	5	4	3	2	1	0
Reserved		ICLK_ON	DBCLKSRC	DBCLKSEL			

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**ICLK_ON**	**Interrupt Clock on Mode** 0 = Edge detection circuit is active only if I/O pin corresponding Px_IEN bit is set to 1. 1 = All I/O pins edge detection circuit is always active after reset. It is recommended to turn off this bit to save system power if no special application concern.
[4]	**DBCLKSRC**	**De-bounce Counter Clock Source Selection** 0 = De-bounce counter clock source is the HCLK. 1 = De-bounce counter clock source is the internal 10 kHz low speed oscillator.
[3:0]	**DBCLKSEL**	**De-bounce Sampling Cycle Selection** 0000 = Sample interrupt input once per 1 clocks. 0001 = Sample interrupt input once per 2 clocks. 0010 = Sample interrupt input once per 4 clocks. 0011 = Sample interrupt input once per 8 clocks. 0100 = Sample interrupt input once per 16 clocks. 0101 = Sample interrupt input once per 32 clocks. 0110 = Sample interrupt input once per 64 clocks. 0111 = Sample interrupt input once per 128 clocks. 1000 = Sample interrupt input once per 256 clocks. 1001 = Sample interrupt input once per 2*256 clocks. 1010 = Sample interrupt input once per 4*256 clocks. 1011 = Sample interrupt input once per 8*256 clocks. 1100 = Sample interrupt input once per 16*256 clocks. 1101 = Sample interrupt input once per 32*256 clocks. 1110 = Sample interrupt input once per 64*256 clocks. 1111 = Sample interrupt input once per 128*256 clocks.

PWM Port Output Enable (PWMPOEN)

Register	Offset	R/W	Description	Reset Value
PWMPOEN	GPIO_BA+0x2E4	R/W	PWM Port Output Enable	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**
Reserved			**HZ_BPWM**	**HZ_Odd1**	**HZ_Even1**	**HZ_Odd0**	**HZ_Even0**

Bits	Description	
[31:5]	**Reserved**	Reserved.
[4]	**HZ_BPWM**	**Basic PWM0 Ports Output Control** 0 = The driving mode of Basic PWM ports are controlled by GPIO mode register. (Px_PMD) or multi-function register (Px_MFP). 1 = The driving mode of Basic PWM ports are forced in tri-state (Hi-Z) all the time. **Note:**The initial value is loaded from CHZ_BPWM(Config0[12]) after any reset.
[3]	**HZ_Odd1**	**Enhanced PWM Unit1 Odd Ports Output Control** 0 = The driving mode of Enhanced PWM unit1 odd ports are controlled by GPIO mode register (Px_PMD) or multi-function register (Px_MFP). 1 = The driving mode of Enhanced PWM unit1 odd ports are forced in tri-state (Hi-Z) all the time. **Note:**The initial value is loaded from CHZ_Odd1 (Config0[11]) after any reset.
[2]	**HZ_Even1**	**Enhanced PWM Unit1 Even Ports Output Control** 0 = The driving mode of Enhanced PWM unit1 even ports are controlled by GPIO mode register (Px_PMD) or multi-function register (Px_MFP). 1 = The driving mode of Enhanced PWM unit1 even ports are forced in tri-state (Hi-Z) all the time. **Note:**The initial value is loaded from CHZ_Even1 (Config0[10]) after any reset.
[1]	**HZ_Odd0**	**Enhanced PWM Unit0 Odd Ports Output Control** 0 = The driving mode of Enhanced PWM unit0 odd ports are controlled by GPIO mode register (Px_PMD) or multi-function register (Px_MFP). 1 = The driving mode of Enhanced PWM unit0 odd ports are forced in tri-state (Hi-Z) all the time. **Note:**The initial value is loaded from CHZ_Odd0 (Config0[9]) after any reset.

Bits	Description	
[0]	**HZ_Even0**	**Enhanced PWM Unit0 Even Ports Output Control** 0 = The driving mode of Enhanced PWM unit0 even ports are controlled by GPIO mode register (Px_PMD) or multi-function register (Px_MFP). 1 = The driving mode of Enhanced PWM unit0 even ports are forced in tri-state (Hi-Z) all the time. **Note:**The initial value is loaded from CHZ_Even0 (Config0[8]) after any reset.

GPIO Px.n Pin Data Input/Output (Px_n)

Register	Offset	R/W	Description	Reset Value
P0_N n=0,1..7	GPIO_BA+0x300+ 0x04 * n	R/W	GPIO P0.n PinData Input/Output	0x0000_000X
P1_N n=0,1..7	GPIO_BA+0x320+ 0x04 * n	R/W	GPIO P1.n PinData Input/Output	0x0000_000X
P2_N n=0,1..7	GPIO_BA+0x340+ 0x04 * n	R/W	GPIO P2.n PinData Input/Output	0x0000_000X
P3_N n=0,1..7	GPIO_BA+0x360+ 0x04 * n	R/W	GPIO P3.n PinData Input/Output	0x0000_000X
P4_N n=0,1..7	GPIO_BA+0x380+ 0x04 * n	R/W	GPIO P4.n PinData Input/Output	0x0000_000X
P5_N n=0,1..7	GPIO_BA+0x3A0+ 0x04 * n	R/W	GPIO P5.n PinData Input/Output	0x0000_000X
P6_N n=0,1..7	GPIO_BA+0x3C0+ 0x04 * n	R/W	GPIO P6.n PinData Input/Output	0x0000_000X
P7_N n=0,1..7	GPIO_BA+0x3E0+ 0x04 * n	R/W	GPIO P7.n PinData Input/Output	0x0000_000X
P8_N n=0,1..7	GPIO_BA+0x400+ 0x04 * n	R/W	GPIO P8.n PinData Input/Output	0x0000_000X
P9_N n=0,1..7	GPIO_BA+0x420+ 0x04 * n	R/W	GPIO P9.n PinData Input/Output	0x0000_000X
PA_N n=0,1	GPIO_BA+0x440+ 0x04 * n	R/W	GPIO PA.n PinData Input/Output	0x0000_000X

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**Pxn**	**GPIO Px.N Pin Data Input/Output** Write this bit can control one GPIO pin output value.

		0 = Corresponding GPIO pin set to low. 1 = Corresponding GPIO pin set to high. Read this register to get GPIO pin status. For example: writing P0_0 will reflect the written value to bit P0_DOUT[0], read P0_0 will return the value of P0_PIN[0]. **Note:** The write operation will not be affected by register Px_DMASK.

6.6 Timer Controller (TIMER)

6.6.1 Overview

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

6.6.2 Features

- Four sets of 32-bit timers with 24-bit up counter and one 8-bit prescale counter
- Independent clock source for each timer
- Provides one-shot, periodic, toggle-output and continuous counting operation modes
- 24-bitup counter value is readable through TDR (TDR[23:0])
- Supports event counting function
- Supports external capture pin event for interval measurement
- Supports external capture pinevent to reset 24-bit up counter

6.6.3 Block Diagram

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

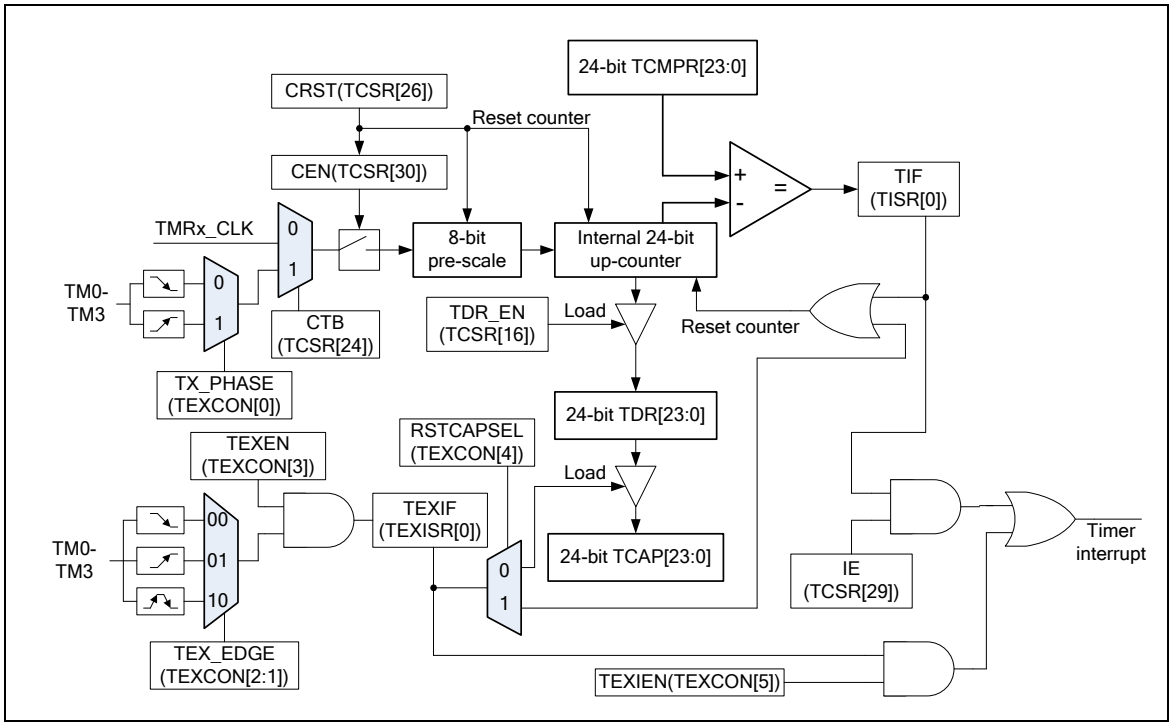


Figure 6-21 Timer Controller Block Diagram

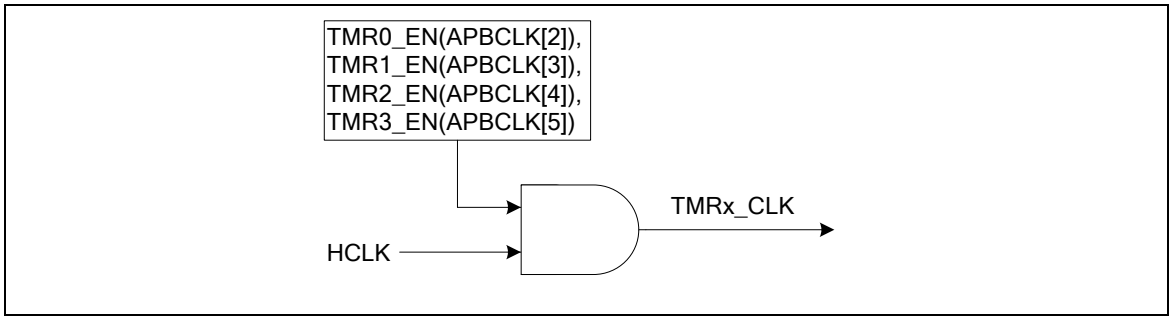


Figure 6-22 Clock Source of Timer Controller

6.6.4 Basic Configuration

The HCLK clock source of Timer0 ~ Timer3 can be enabled in TMRx_EN (APBCLK[5:2]).

6.6.5 Function Description

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

6.6.5.1 Timer Interrupt Flag

Timer controller supports two interrupt flags; one is TIF (TISR[0]) and its set while timer counter value TDR(TDR[23:0]) matches the timer compared value TCMP (TCMPR[23:0]), the other is TEXIF (TEXISR[0]) and its set when the transition on the TMx pin associated TEX_EDGE (TEXCON[2:1])setting.

6.6.5.2 Timer Counting Mode

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

6.6.5.3 One-shot Mode

If timer controller is configured at one-shot mode (TCSR[28:27] is 00) and CEN (TCSR[30]) is set, the timer counter starts up counting. Once the TDR (TDR[23:0]) value reaches TCMP (TCMPR[23:0]) value, the TIF (TISR[0])will be set to 1, TDR value and CEN bit is cleared automatically by timer controller then timer counting operation stops. In the meantime, if the IE (TCSR[29]) is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU also.

6.6.5.4 Periodic mode

If timer controller is configured at periodic mode (TCSR[28:27] is 01) and CEN(TCSR[30]) is set, the timer counter starts up counting. Once the TDR(TDR[23:0]) value reaches TCMP(TCMPR[23:0]) value, the TIF (TISR[0]) will be set to 1, TDR value will be cleared automatically by timer controller and timer counter operates counting again. In the meantime, if the IE(TCSR[29]) bit is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU also. In this mode, timer controller operates counting and compares with TCMP value periodically until the CEN bit is cleared by user.

6.6.5.5 Toggle-Output mode

If timer controller is configured at toggle-output mode (TCSR[28:27] is 10) and CEN(TCSR[30])is set, the timer counter starts up counting. The counting operation of toggle-output mode is almost the same as periodic mode, except toggle-output mode has associated TM0~TM3 pin to output signal while specify TIF(TISR[0]) is set. Thus, the toggle-output signal on TM0~TM3 pinis high and changing back and forth with 50% duty cycle.

6.6.5.6 *Continuous Counting Mode*

If timer controller is configured at continuous counting mode (TCSR[28:27] is 11) and CEN(TCSR[30])is set, the timer counter starts up counting. Once the TDR (TDR[23:0]) value reaches TCMP(TCMPR[23:0]) value, the TIF(TISR[0])will be set to 1 and TDR value keeps up counting. In the meantime, if the IE(TCSR[29]) is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU also. User can change different TCMP value immediately without disabling timer counting and restarting timer counting in this mode.

For example, TCMP value is set as 80, first. The TIFwill set to 1 when TDR value is equal to 80, timer counter is kept counting 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 TCMP value as 200 and clears TIF, the TIF will set to 1 again when TDR value reaches to 200. At last, user programs TCMP as 500 and clears TIF, the TIF will set to 1 again when TDR value reaches to 500.

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

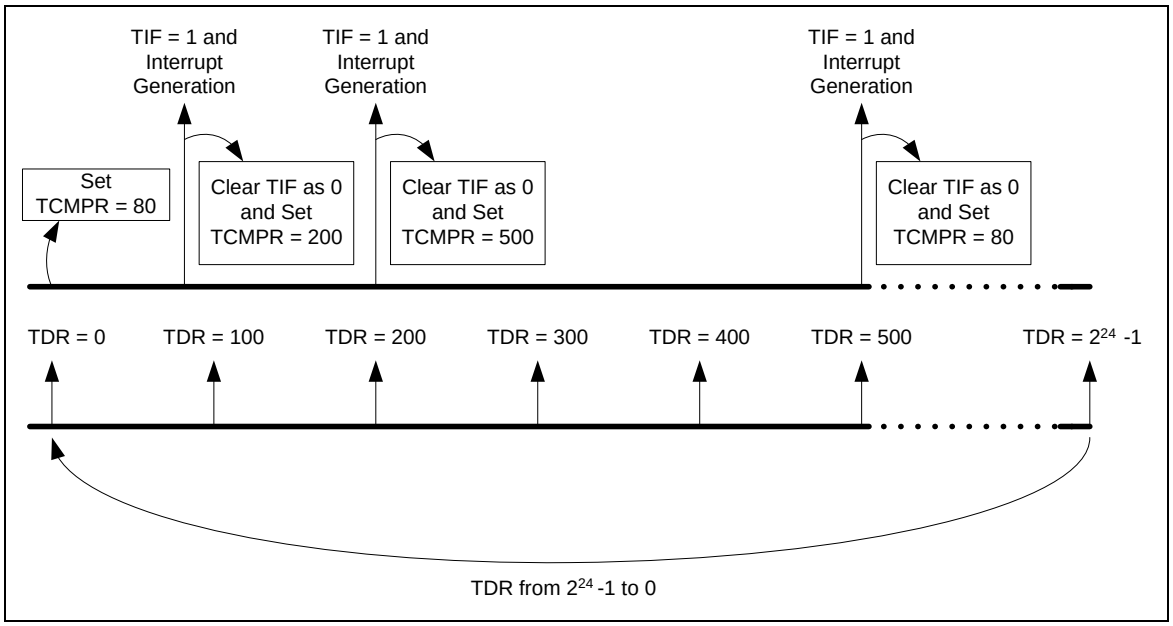


Figure 6-23 Continuous Counting Mode

6.6.5.7 *Event Counting Function*

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

User can enable or disable TMx pin de-bounce circuit by setting TCDB (TEXCON[7]). The input event frequency should be less than 1/3 HCLK if Tmxpin de-bounce disabled or less than 1/8 HCLK if TMx pin de-bounce enabled to assure the returned TDR value is correct, and user can also select edge detection phase of TMx pin by setting TX_PHASE (TEXCON[0]) bit.

In event counting mode, the timer counting operation mode can be selected as one-shot, periodic and continuous counting mode to counts the counter value TDR(TDR[23:0]) forTMx pin.

6.6.5.8 *External Pin Capture Function*

The event capture function is used to loadTDR(TDR[23:0]) value to TCAP(TCAP[23:0]) value while edge transition detected on TMx(x= 0~3)pin. In this mode, RSTCAPn (TEXCON[4]) should be as 0 for select TMx transition is using to trigger event capture function and the timer peripheral clock source should be set as HCLK.

User can enable or disable TMx pin de-bounce circuit by setting TEXDB (TEXCON[6]). The transition frequency of TMx pin should be less than 1/3 HCLK if TMx pin de-bounce disabled or less than 1/8 HCLK if TMx pin de-bounce enabled to assure the capture function can be work normally, and user can also select edge transition detection of TMx pin by setting TEX_EDGE (TEXCON[2:1]).

In event capture mode, user does not consider what timer counting operation mode is selected, the capture event occurred only if edge transition on TMx pin is detected.

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

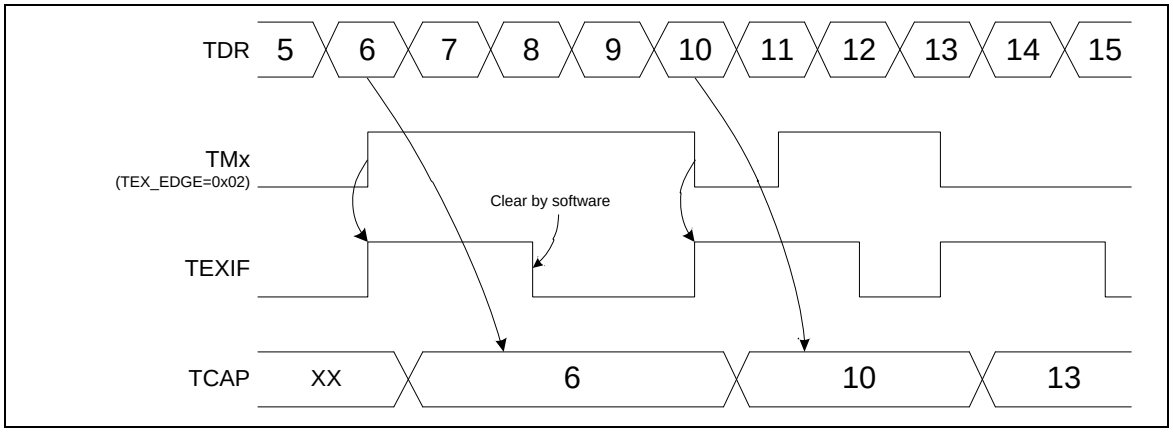


Figure 6-24External Capture Mode

6.6.5.9 *ExternalReset Counter Mode*

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

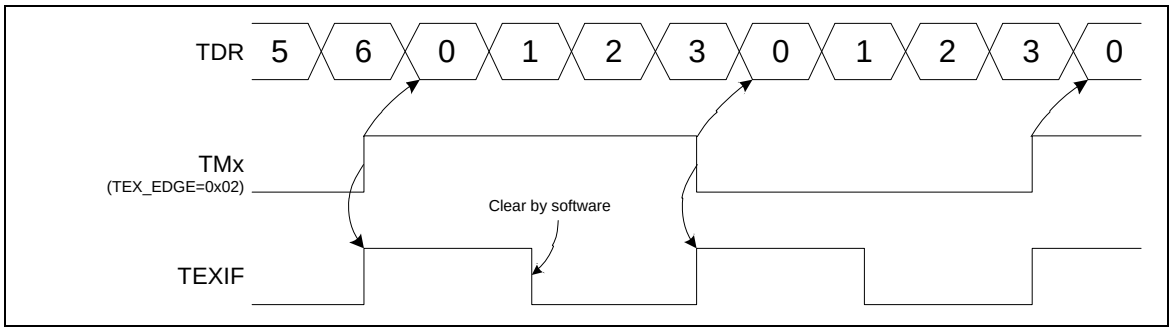


Figure 6-25External Reset Counter Mode

6.6.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
TMR Base Address: **TMR01_BA = 0x4001_0000** **TMR23_BA = 0x4011_0000**				
TCSR0	TMR01_BA+0x00	R/W	Timer0 Control and Status Register	0x0000_0005
TCMPR0	TMR01_BA+0x04	R/W	Timer0 Compare Register	0x0000_0000
TISR0	TMR01_BA+0x08	R/W	Timer0 Interrupt Status Register	0x0000_0000
TDR0	TMR01_BA+0x0C	R	Timer0 Data Register	0x0000_0000
TCAP0	TMR01_BA+0x10	R	Timer0 Capture Data Register	0x0000_0000
TEXCON0	TMR01_BA+0x14	R/W	Timer0 External Control Register	0x0000_0000
TEXISR0	TMR01_BA+0x18	R/W	Timer0 External Interrupt Status Register	0x0000_0000
TCSR1	TMR01_BA+0x20	R/W	Timer1 Control and Status Register	0x0000_0005
TCMPR1	TMR01_BA+0x24	R/W	Timer1 Compare Register	0x0000_0000
TISR1	TMR01_BA+0x28	R/W	Timer1 Interrupt Status Register	0x0000_0000
TDR1	TMR01_BA+0x2C	R	Timer1 Data Register	0x0000_0000
TCAP1	TMR01_BA+0x30	R	Timer1 Capture Data Register	0x0000_0000
TEXCON1	TMR01_BA+0x34	R/W	Timer1 External Control Register	0x0000_0000
TEXISR1	TMR01_BA+0x38	R/W	Timer1 External Interrupt Status Register	0x0000_0000
TCSR2	TMR23_BA+0x00	R/W	Timer2 Control and Status Register	0x0000_0005
TCMPR2	TMR23_BA+0x04	R/W	Timer2 Compare Register	0x0000_0000
TISR2	TMR23_BA+0x08	R/W	Timer2 Interrupt Status Register	0x0000_0000
TDR2	TMR23_BA+0x0C	R	Timer2 Data Register	0x0000_0000
TCAP2	TMR23_BA+0x10	R	Timer2 Capture Data Register	0x0000_0000
TEXCON2	TMR23_BA+0x14	R/W	Timer2 External Control Register	0x0000_0000
TEXISR2	TMR23_BA+0x18	R/W	Timer2 External Interrupt Status Register	0x0000_0000
TCSR3	TMR23_BA+0x20	R/W	Timer3 Control and Status Register	0x0000_0005
TCMPR3	TMR23_BA+0x24	R/W	Timer3 Compare Register	0x0000_0000
TISR3	TMR23_BA+0x28	R/W	Timer3 Interrupt Status Register	0x0000_0000
TDR3	TMR23_BA+0x2C	R	Timer3 Data Register	0x0000_0000
TCAP3	TMR23_BA+0x30	R	Timer3 Capture Data Register	0x0000_0000

TEXCON3	TMR23_BA+0x34	R/W	Timer3 External Control Register	0x0000_0000
TEXISR3	TMR23_BA+0x38	R/W	Timer3 External Interrupt Status Register	0x0000_0000

6.6.7 Register Description

Timer Control and Status Register (TCSR)

Register	Offset	R/W	Description	Reset Value
TCSR0	TMR01_BA+0x00	R/W	Timer0 Control and Status Register	0x0000_0005
TCSR1	TMR01_BA+0x20	R/W	Timer1 Control and Status Register	0x0000_0005
TCSR2	TMR23_BA+0x00	R/W	Timer2 Control and Status Register	0x0000_0005
TCSR3	TMR23_BA+0x20	R/W	Timer3 Control and Status Register	0x0000_0005

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

Bits	Description	
[31]	DBGACK_TMR	**ICE Debug Mode Acknowledge Disable (Write Protect)** 0 = ICE debug mode acknowledgement effects TIMER counting. TIMER counter will be held while CPU is held by ICE. 1 = ICE debug mode acknowledgement Disabled. TIMER counter will keep going no matter CPU is held by ICE or not. **Note:** This bit is write protected. Refer to the REGWRPROT register.
[30]	CEN	**Timer Enable Bit** 0 = Stops/Suspends counting. 1 = Starts counting. **Note1:** In stop status, and then set CEN to 1 will enable the 24-bit up counter to keep counting from the last stop counting value. **Note2:** This bit is auto-cleared by hardware in one-shot mode (TCSR[28:27] = 00) when the timer interrupt flag TIF (TISR[0]) is generated.
[29]	IE	**Interrupt Enable Bit** 0 = Timer Interrupt Disabled. 1 = Timer Interrupt Enabled. **Note:** If this bit is enabled, when the timer interrupt flag TIFis set to 1, the timer interrupt signal is generated and inform to CPU.

[28:27]	**MODE**	**Timer Counting Mode Select** 00 = The Timer controller is operated in One-shot mode. 01 = The Timer controller is operated in Periodic mode. 10 = The Timer controller is operated in Toggle-output mode. 11 = The Timer controller is operated in Continuous Counting mode.
[26]	**CRST**	**Timer Counter Reset Bit** Setting this bit will reset the 24-bit up counter value TDR and also force CEN (TCSR[30]) to 0 if CACT (TCSR[25]) is 1. 0 = No effect. 1 = Reset internal 8-bit prescale counter, 24-bit up counter value and CEN bit.
[25]	**CACT**	**Timer Active Status Bit (Read Only)** This bit indicates the up-timer status. 0 = Timer is not active. 1 = Timer is active.
[24]	**CTB**	**Counter Mode Enable Bit** This bit is for external counting pin function enabled. 0 = Event counter mode Disabled. 1 = Event counter mode Enabled.
[23:17]	**Reserved**	Reserved.
[16]	**TDR_EN**	**Data Load Enable Bit** When this bit is set, timer counter value (TDR) will be updated continuously to monitor internal 24-bit up counter value as the counter is counting. 0 = Timer Data Register update Disabled. 1 = Timer Data Register update Enabled while timer counter is active.
[15:8]	**Reserved**	Reserved.
[7:0]	**PRESCALE**	**Prescale Counter** Timer input clock source is divided by (PRESCALE+1) before it is fed to the timer up counter. If this field is 0 (PRESCALE = 0), then there is no scaling.

Timer Compare Register (TCMPR)

Register	Offset	R/W	Description	Reset Value
TCMPR0	TMR01_BA+0x04	R/W	Timer0 Compare Register	0x0000_0000
TCMPR1	TMR01_BA+0x24	R/W	Timer1 Compare Register	0x0000_0000
TCMPR2	TMR23_BA+0x04	R/W	Timer2 Compare Register	0x0000_0000
TCMPR3	TMR23_BA+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							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
TCMP							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
TCMP							

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**TCMP**	**Timer Compared Value** TCMP is a 24-bit compared value register. When the internal 24-bit up counter value is equal to TCMP value, the TIF (TISR[0] Timer Interrupt Flag) will set to 1. Time-out period = (Period of timer clock input) * (8-bit PRESCALE+ 1) * (24-bit TCMP). **Note1:** Never write 0x0 or 0x1 in TCMP field, or the core will run into unknown state. **Note2:** When timer is operating at continuous counting mode, the 24-bit up counter will keep counting continuously even if user writes a new value into TCMP field. But if timer is operating at other modes, the 24-bit up counter will restart counting from 0 and using newest TCMP value to be the timer compared value while user writes a new value into TCMP field.

Timer Interrupt Status Register (TISR)

Register	Offset	R/W	Description	Reset Value
TISR0	TMR01_BA+0x08	R/W	Timer0 Interrupt Status Register	0x0000_0000
TISR1	TMR01_BA+0x28	R/W	Timer1 Interrupt Status Register	0x0000_0000
TISR2	TMR23_BA+0x08	R/W	Timer2 Interrupt Status Register	0x0000_0000
TISR3	TMR23_BA+0x28	R/W	Timer3 Interrupt Status Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**TIF**	**Timer Interrupt Flag** This bit indicates the interrupt flag status of Timer while 24-bit timer up counter TDRvalue reaches to TCMP (TCMPR[23:0]) value. 0 = No effect. 1 = TDR value matches the TCMP value. **Note:**This bit is cleared by writing 1 to it.

Timer Data Register (TDR)

Register	Offset	R/W	Description	Reset Value
TDR0	TMR01_BA+0x0C	R	Timer0 Data Register	0x0000_0000
TDR1	TMR01_BA+0x2C	R	Timer1 Data Register	0x0000_0000
TDR2	TMR23_BA+0x0C	R	Timer2 Data Register	0x0000_0000
TDR3	TMR23_BA+0x2C	R	Timer3 Data Register	0x0000_0000

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

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

Timer Capture Data Register (TCAP)

Register	Offset	R/W	Description	Reset Value
TCAP0	TMR01_BA+0x10	R	Timer0 Capture Data Register	0x0000_0000
TCAP1	TMR01_BA+0x30	R	Timer1 Capture Data Register	0x0000_0000
TCAP2	TMR23_BA+0x10	R	Timer2 Capture Data Register	0x0000_0000
TCAP3	TMR23_BA+0x30	R	Timer3 Capture Data Register	0x0000_0000

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

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**TCAP**	**Timer Capture Data Register** When TEXEN (TEXCON[3]) bit is set, RSTCAPn (TEXCON[4]) bit is 0, and a transition on TMx (x=0~3)pin matched the TEX_EDGE (TEXCON[2:1]) setting, TEXIF (TEXISR[0]) will set to 1 and the current timer counter value TDRwill be auto-loaded into this TCAP field.

Timer External Control Register (TEXCON)

Register	Offset	R/W	Description	Reset Value
TEXCON0	TMR01_BA+0x14	R/W	Timer0 External Control Register	0x0000_0000
TEXCON1	TMR01_BA+0x34	R/W	Timer1 External Control Register	0x0000_0000
TEXCON2	TMR23_BA+0x14	R/W	Timer2 External Control Register	0x0000_0000
TEXCON3	TMR23_BA+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**	**RSTCAPSEL**	**TEXEN**	**TEX_EDGE**		**TX_PHASE**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**TCDB**	**Timer Counter Pin De-bounce Enable Bit** 0 = TMx (x= 0~3) pin de-bounce Disabled. 1 = TMx (x= 0~3) pin de-bounce Enabled. **Note:** If this bit is enabled, the edge detection of TMx pin is detected with de-bounce circuit.
[6]	**TEXDB**	**Timer External Capture Pin De-bounce Enable V** 0 = TMx (x= 0~3) pin de-bounce Disabled. 1 = TMx (x= 0~3) pin de-bounce Enabled. **Note:** If this bit is enabled, the edge detection of TMx pin is detected with de-bounce circuit.
[5]	**TEXIEN**	**Timer External Capture Interrupt Enable Bit** 0 = TMx (x= 0~3) pin detection Interrupt Disabled. 1 = TMx (x= 0~3) pin detection Interrupt Enabled. **Note:** TEXIEN is used to enable timer external interrupt. If TEXIEN enabled, timer will rise an interrupt when TEXIF (TEXISR[0]) is 1. For example, while TEXIEN = 1, TEXEN = 1, and TEX_EDGE = 00, a 1 to 0 transition on the TMx pin will cause the TEXIFto be set then the interrupt signal is generated and sent to NVIC to inform CPU.
[4]	**RSTCAPSEL**	**Capture Function Selection** 0 = External Capture Mode Enabled. 1 = External Reset Mode Enabled. **Note1:** When RSTCAPn is 0,transition on TMx (x= 0~3) pin is using to save the 24-bit timer counter value.

		Note2: When RSTCAPn is 1,transition on TMx (x= 0~3) pin is using to reset the 24-bit timer counter value.
[3]	**TEXEN**	**Timer External Capture Pin Enable Bit** This bit enables theTMx pin. 0 =TMx (x= 0~3) pin Disabled. 1 =TMx(x= 0~3) pin Enabled.
[2:1]	**TEX_EDGE**	**Timer External Capture Pin Edge Detect** 00 = A Fallingedge on TMx (x= 0~3) pin will be detected. 01 = A Risingedge on TMx (x= 0~3) pin will be detected. 10 = Either Rising or Fallingedge on TMx (x= 0~3) pin will be detected. 11 = Reserved.
[0]	**TX_PHASE**	**Timer External Count Phase** This bit indicates the detection phase of external counting pin TMx (x= 0~3). 0 = A Falling edge of external counting pin will be counted. 1 = A Rising edge of external counting pin will be counted.

Timer External Interrupt Status Register (TEXISR)

Register	Offset	R/W	Description	Reset Value
TEXISR0	TMR01_BA+0x18	R/W	Timer0 External Interrupt Status Register	0x0000_0000
TEXISR1	TMR01_BA+0x38	R/W	Timer1 External Interrupt Status Register	0x0000_0000
TEXISR2	TMR23_BA+0x18	R/W	Timer2 External Interrupt Status Register	0x0000_0000
TEXISR3	TMR23_BA+0x38	R/W	Timer3 External Interrupt Status Register	0x0000_0000

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

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

6.7 Basic PWM Generatorand Capture Timer (BPWM)

6.7.1 Overview

The NuMicro®NM1530 series has 1 set of BPWM group (BPWM0), supporting 1 set ofBPWM generators that can be configured as 2 independent BPWM outputs, BPWM0_CH0 and BPWM0_CH1, or as 1 complementary BPWM pairs, (BPWM0_CH0, BPWM0_CH1) with programmable dead-zone generator.

TheBPWM generator has one 8-bit prescaler, one clock divider with 5 divided frequencies (1, 1/2, 1/4, 1/8, 1/16), two BPWM Timers including two clock selectors, two 16-bit BPWM down-counters for BPWM period control, two 16-bit comparators for BPWM duty control and one dead-zone generator. The BPWM generator providestwo independent BPWM interrupt flags which are set by hardware when the corresponding BPWM period down counter reaches zero.

Each BPWM interrupt source with its corresponding enable bit can cause CPU to request BPWM interrupt. The BPWM generators can be configured as one-shot mode to produce only one BPWM cycle signal or auto-reload mode to output BPWM waveform continuously.BPWM can be used to trigger EADC when operation in center-aligned mode.

6.7.2 Features

6.7.2.1 BPWM Function:

- Up to 1 BPWM group to support 2 BPWM channels or 1 BPWM paired channels.
- Supports 8-bit prescaler from 1 to 255
- Up to 16-bit resolutionBPWM timer
- PWM timer supports edge-aligned and center-aligned operation type
- One-shot or Auto-reload mode BPWM
- PWM Interrupt request synchronized with BPWM period or duty
- Supports dead-zone generator with 8-bit resolution for BPWM paired channels
- Supports trigger EADC

6.7.2.2 Capture Function:

- Supports 2 Capture input channels shared with 2 BPWM output channels
- Supports rising or falling capture condition
- Supports rising or falling capture interrupt

6.7.3 Block Diagram

The Figure 6-26 and Figure 6-27illustrate the architecture of BPWM in pair.

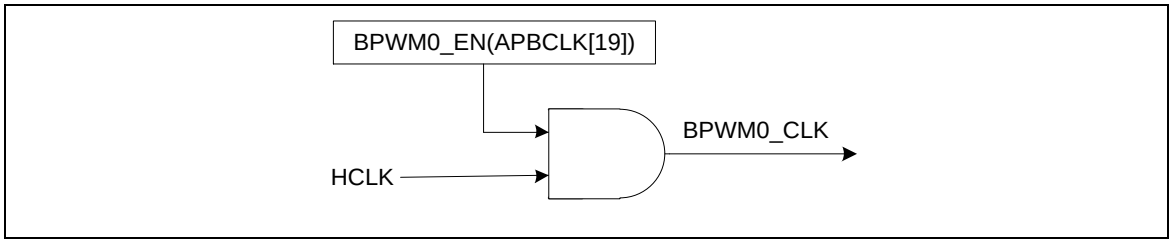


Figure 6-26BPWM Clock Source Control

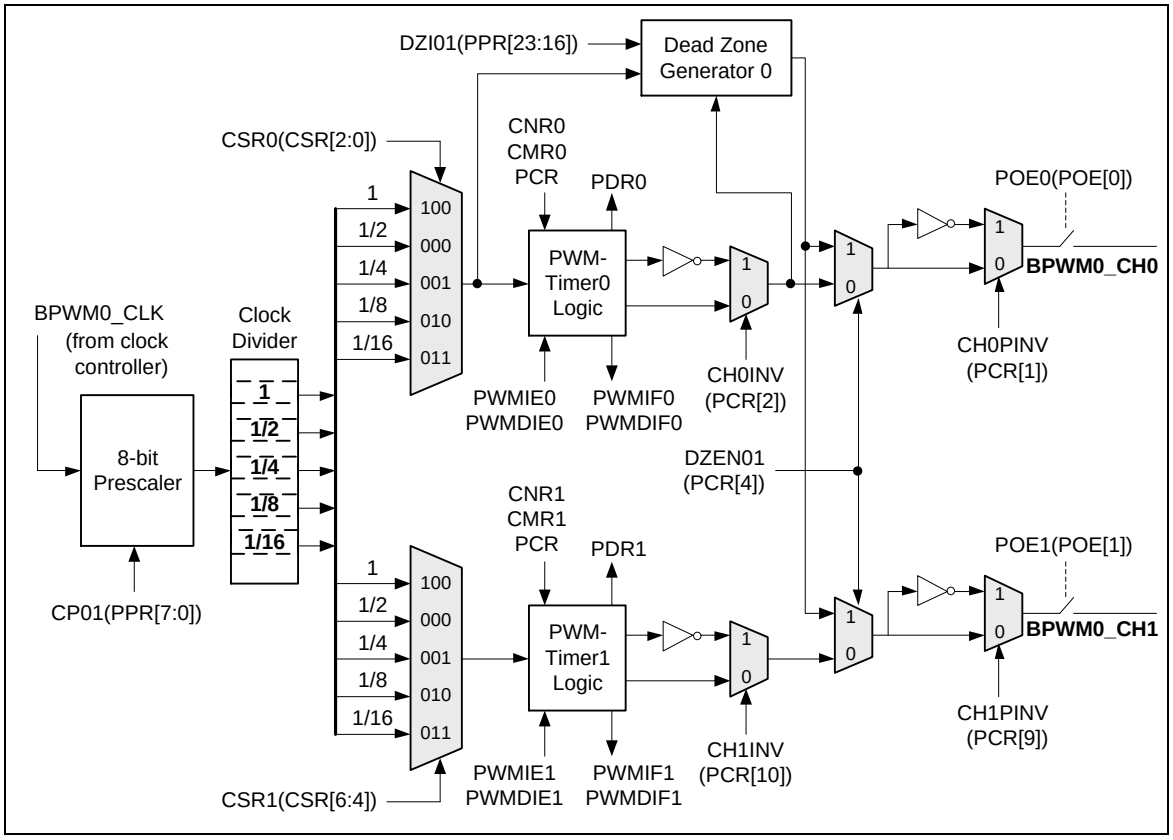


Figure 6-27BPWM Architecture Diagram

6.7.4 Basic Configuration

The BPWM pin functions are configured in P5_MFP[7:6] Multiple Function Registers.

The peripheral clock source of BPWM can be enabled in BPWM0_EN (APBCLK[19]).

The driving mode of BPWM ports are configured in PWMPOEN[4] PWM port output enable register.

6.7.5 Function Description

BPWM controller supports 2 operation modes: Edge-aligned and Center-aligned mode.

6.7.5.1 Edge-aligned BPWM (down-counter)

In Edge-aligned BPWM output mode, the 16 bits BPWM counter will start down-counting from period value (CNRn register)to zero to finish a BPWM period, then restart down-counting from period value to zero again if auto-reload mode is enabled(CHnMOD bit is 1 in PCR register).The value of BPWM counter will be compared with comparator value(CMRn register) to control output level of BPWM generator.The BPWM generator will output low when the value of BPWM counter is larger than comparator value and output high when the value of BPWM counter is equal or smaller than comparator value.

The BPWM period interrupt(BPWMIFn(PIIR[1:0])) will be triggered by setting BPWMPIEn(PIER[1:0]) to 1, and BPWM duty interrupt(BPWMDIFn(PIIR[9:8])) will be triggered by setting BPWMDIEn(PIER[9:8]) to 1.

The BPWM period and duty control are configured by BPWM down-counter register (CNR) and BPWM comparator register (CMR). The new period and comparator value will take effect at the start of next period.The BPWM-timer timing operation is shown inFigure 6-29. The pulse width modulation follows the formula below and the legend of BPWM-Timer Comparator is shown asFigure 6-28. Note that the corresponding GPIO pins must be configured as BPWM function (enable POE and disable CAPENR) for the corresponding BPWM channel.

- BPWM frequency = BPWM0_CLK/[(prescale+1)*(clock divider)*(CNR+1)].
- Duty ratio = (CMR+1)/(CNR+1)
- CMR >= CNR: BPWM output is always high
- CMR < CNR: BPWM low width= (CNR-CMR) unit[1]; BPWM high width = (CMR+1) unit
- CMR = 0: BPWM low width = (CNR) unit; BPWM high width = 1 unit

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

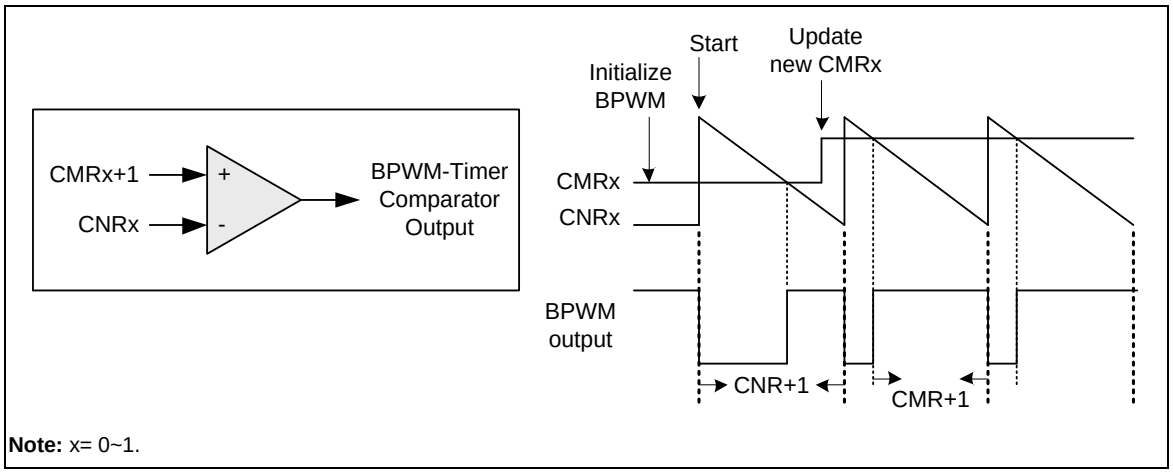


Figure 6-28 Legend of Internal Comparator Output of BPWM-Timer

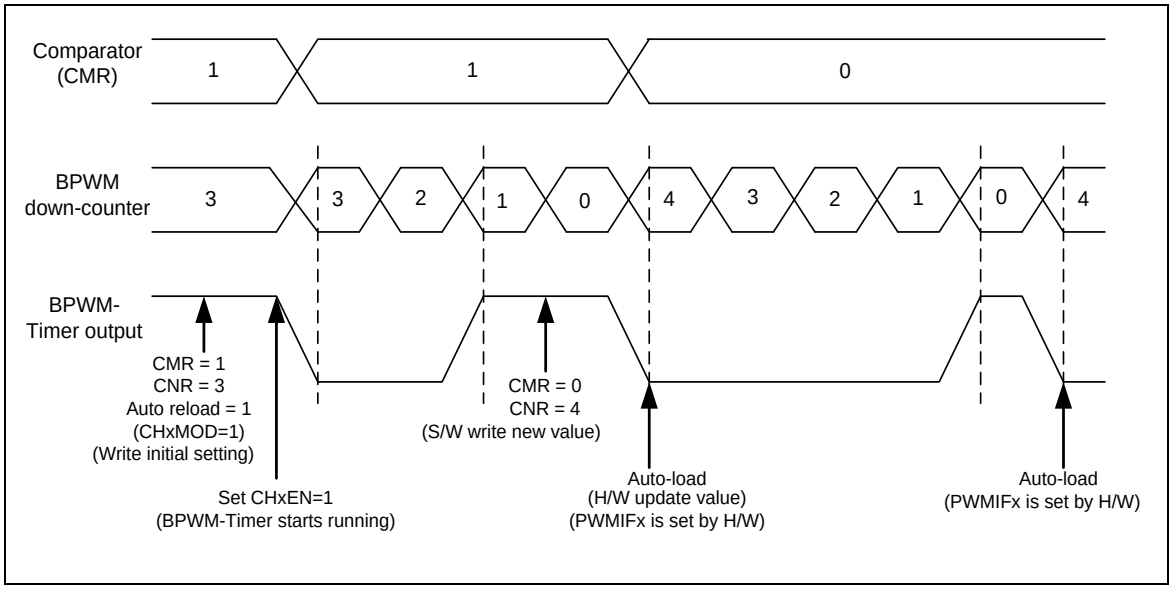


Figure 6-29BPWM-Timer Operation Timing

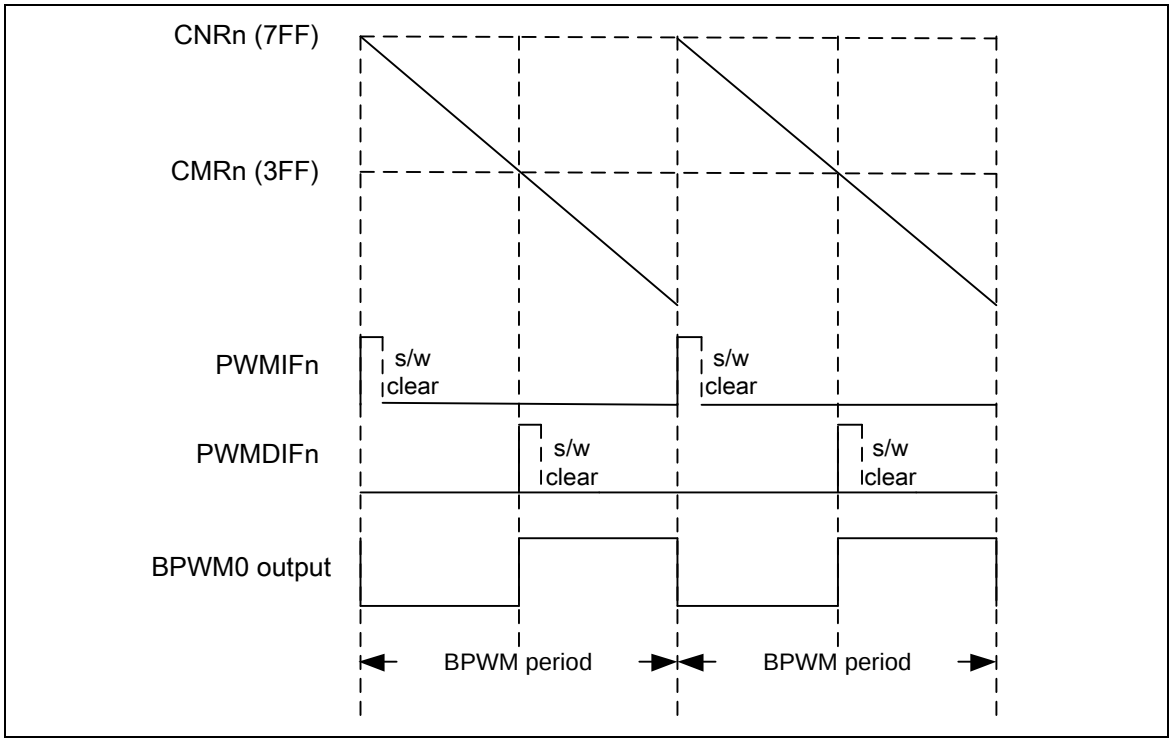


Figure 6-30BPWM Edge-aligned Interrupt Generate Timing Waveform

6.7.5.2 Center-aligned BPWM (up/down-counter)

The Center-aligned BPWM signals are produced by the module when the BPWM time base is configured in an Up/Down Counting mode. TheBPWM counter will start counting-up from zero to the value of CNRn register and then start counting down to zero to finish a BPWM period, then restart next BPWM period again if auto-reload mode is enabled (CHnMOD bit is 1 in PCR register).The value of BPWM counter will be compared with comparator value (CMRn register) to

control output level of BPWM generator. The BPWM generator will output low when the value of BPWM counter is larger than comparator value and output high when the value of BPWM counter is equal or smaller than comparator value.Once the BPWM counter underflows the new period and comparator value will take effect when BPWM timer is operating at auto-reload mode.

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

- BPWM frequency = BPWM0_CLK/[(prescale+1)*(clock divider)*(CNR+1)].
- Duty ratio = [(2 x CMR) + 1]/[2 x (CNR+1)]
- CMR > CNR: BPWM output is always high
- CMR <= CNR: BPWM low width= 2 x (CNR-CMR) + 1 unit[1]; BPWM high width = (2 x CMR) + 1 unit
- CMR = 0: BPWM low width = 2 x CNR + 1 unit; BPWM high width = 1 unit

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

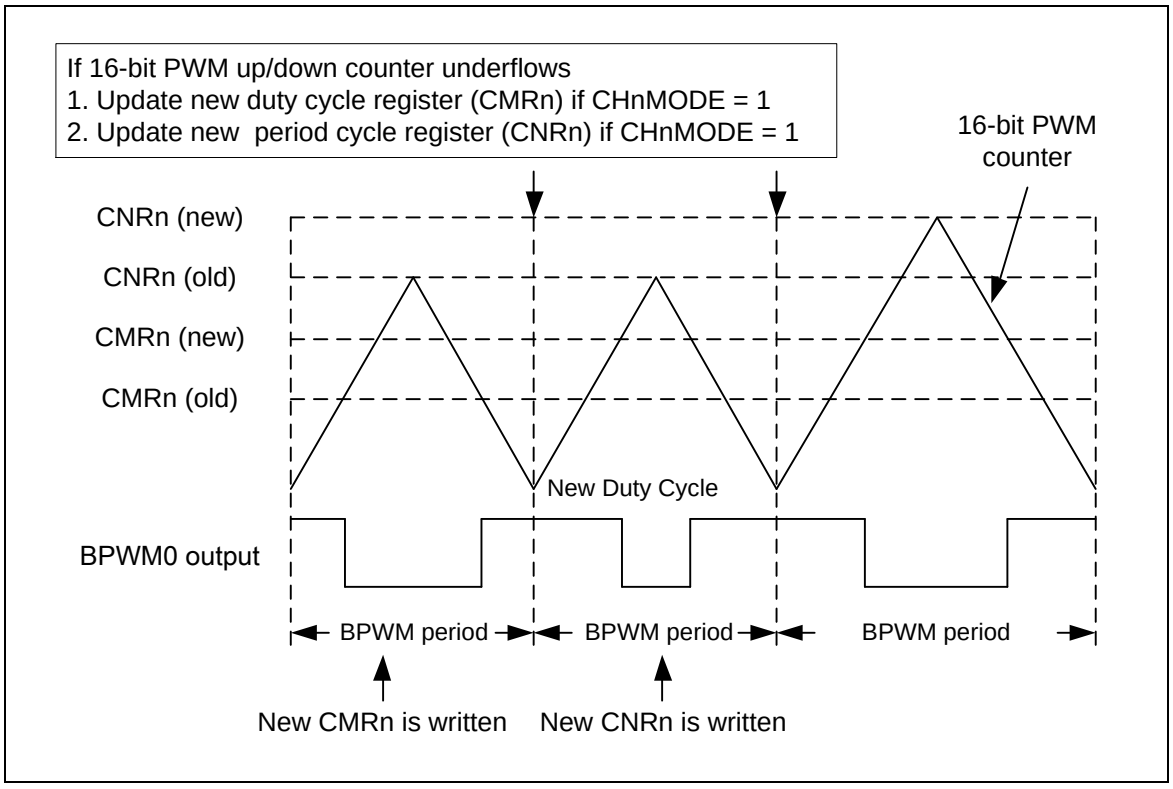


Figure 6-31 Center-aligned Type Output Waveform

In Center-aligned type, system can generate period interrupt, at two specified timings. BPWM period interrupt is generated at counter equals zero on down-count if INTTYPE (PIER[16]) = 0 or at counter equals CNRn on up-count if INTTYPE (PIER[16]) = 1, i.e. at center point of BPWM cycle.

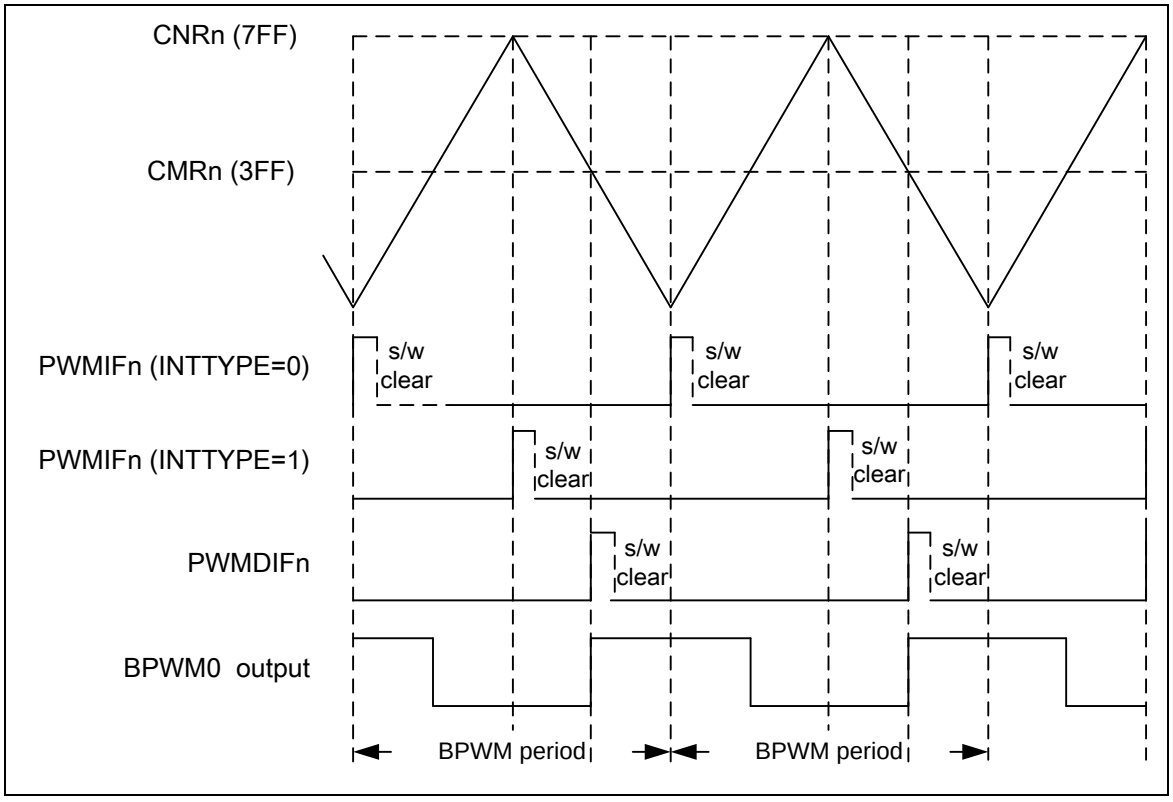


Figure 6-32BPWM Center-aligned Interrupt Generate Timing Waveform

6.7.5.3 BPWM Double Buffering, Auto-reload and One-shot Operation

To prevent BPWM 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 reload value is updated at the start of next period without affecting current timer operation and avoids glitch at BPWM outputs. The BPWM counter value can be written into CNRnand the current BPWM counter value can be read from PDRn.

BPWM0_CH0 will operate in One-shot mode if CH0MOD bit is set to 0, and operate in Auto-reload mode if CH0MOD bit is set to 1. It is recommend that switch BPWM0_CH0 operating mode before set CH0EN bit to 1 to enable BPWM0_CH0 counter start running because the content of CNR0 and CMR0 will be cleared to 0 to reset the BPWM0_CH0 period and duty setting when BPWM0_CH0 operating mode is changed.AsBPWM0_CH0 operate in One-shot mode, CNR0 and CMR0 should be written first and then set CH0EN bit to 1 to enable BPWM0_CH0 counter start running. After BPWM0_CH0 counter down count from CNR0 value to 0, CNR0 and CMR0 will be cleared to 0 by hardware and BPWM counter will be held. Software need to write new CNR0 and CMR0 value to set next one-shot period and duty. When re-start next one-shot operation, the CMR0 should be written first, because BPWM0_CH0 counter will auto re-start counting when CNR0 is written a non-zero value. AsBPWM0_CH0 operates at auto-reload mode, CNR0 and CMR0 should be written first and then set CH0EN bit to 1 to enable BPWM0_CH0 counter start running. The value of CNR0 will reload to BPWM0_CH0 counter when it down count reaches 0. If CNR0 is set to 0, BPWM0_CH0 counter will be held.BPWM0_CH1 performs the same function as BPWM0_CH0.

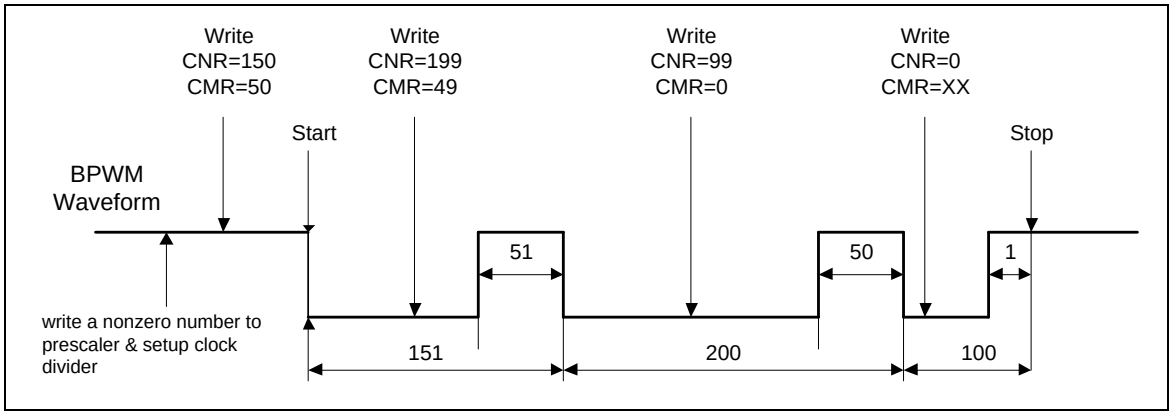


Figure 6-33BPWM Double Buffering Illustration

6.7.5.4 Modulate Duty Ratio

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

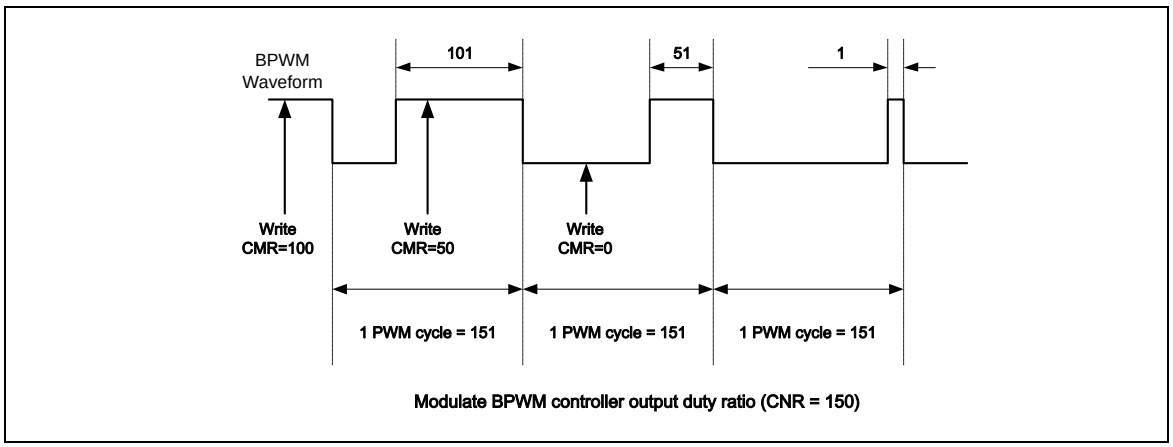


Figure 6-34BPWM Controller Output Duty Ratio

6.7.5.5 Dead-Zone Generator

The dead-zone generator inserts an “off” period called “dead-zone” between the turnings off of one pin to the turning on of the complementary pin of the paired pins. This is to prevent damage to the power switching devices that will be connected to the BPWM output pins. The complementary output pair mode has an 8-bit down counter DZI01 (PPR[23:16]) used to produce the dead-zone insertion. The complementary outputs are delayed until the timer counts down to zero.

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

Dead-Zone = BPWM0_CLK * (DZI01 (PPR[23:16]) +1).

The timing diagram below indicates the dead-zone insertion for pair of BPWM signals

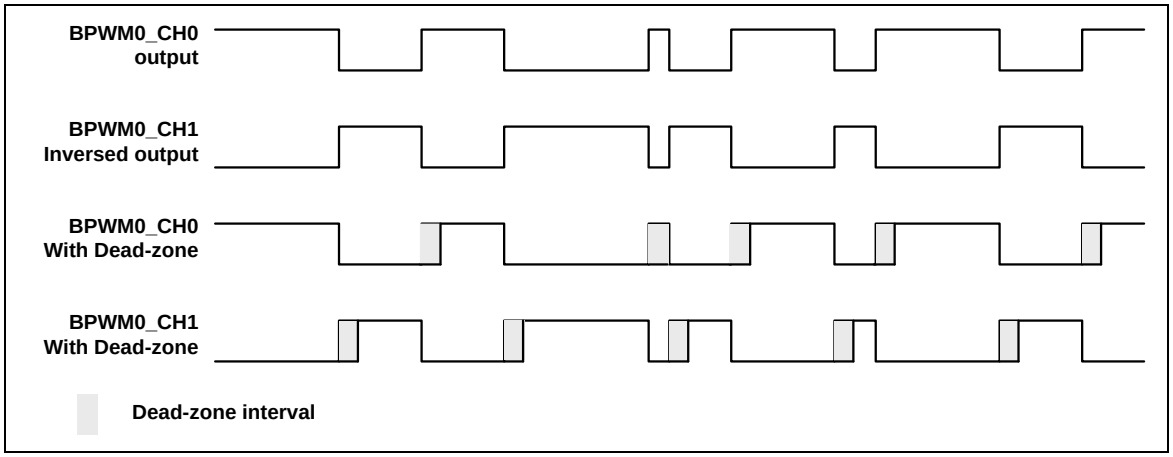


Figure 6-35 Paired-BPWM Output with Dead-zone Generation Operation

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

6.7.5.6 **Polarity Control**

BPWM0_CH0 andBPWM0_CH1 have independent polarity control to configure the polarity of active state of BPWM output. By default, the BPWM output is active high.

The following diagram shows the initial state before BPWM starts with different polarity settings.

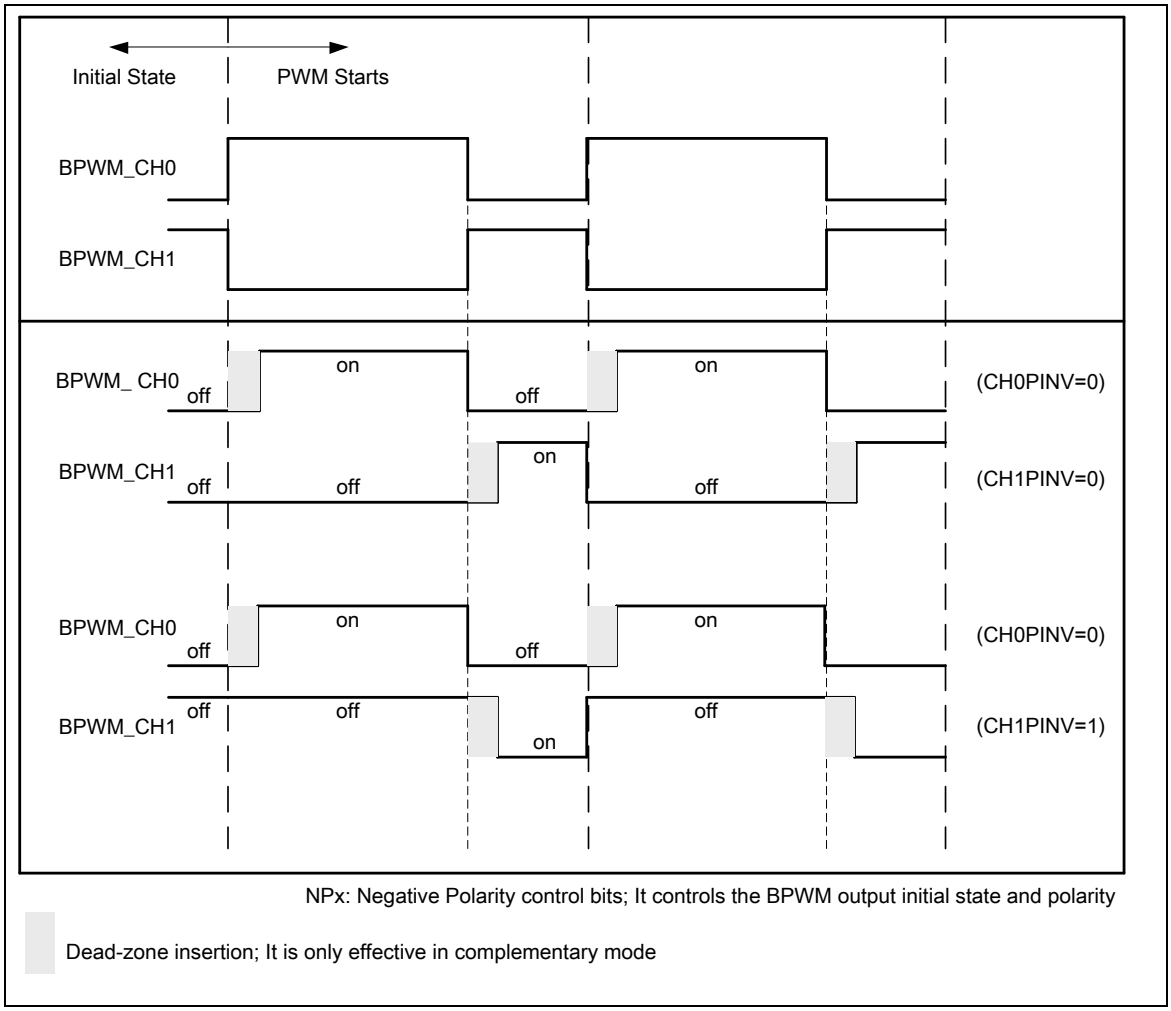


Figure 6-36Initial State and Polarity Control with Rising Edge Dead-zoneInsertion

6.7.5.7 **BPWM Trigger EADC Function**

BPWM can trigger EADC to start conversion when BPWM counter output has rising edge or, falling edge or count to period of edge-aligned, center-aligned mode or center of center-aligned mode.

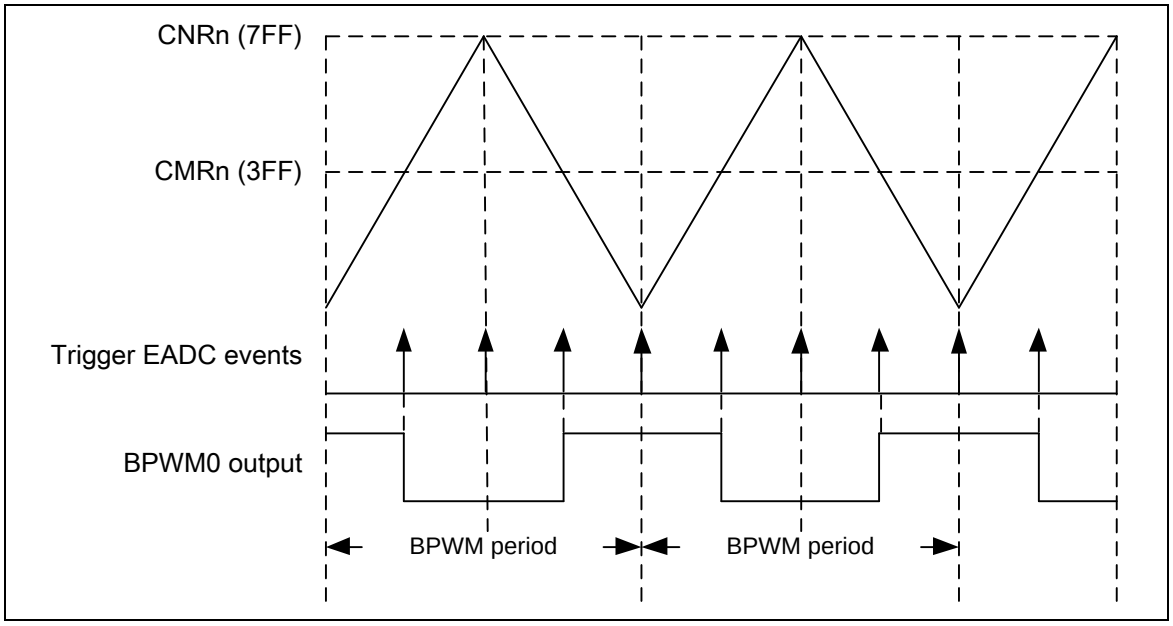


Figure 6-37BPWM Trigger EADC Timing Diagram

6.7.5.8 Capture Operation

The alternate feature of the BPWM-timer is digital input Capture function.The Capture input channel 0 and output channel 0 share one timer and pin(BPWM0_CH0); and the Capture input channel 1 and output channel 1 share another timer and pin (BPWM0_CH1). Therefore, BPWM function and Capture function in the same channel cannot be used at the same time. User must setup the BPWM-timer before enable Capture feature. After capture feature is enabled, the capture always latches BPWM-counter to Capture Rising Latch Register CRLRn when input channel has a rising transition and latches BPWM-counter to Capture Falling Latch Register CFLRn when input channel has a falling transition. Capture channel 0 interrupt is programmable by setting CRL_IE0 (CCR[1]) (Rising latch Interrupt enable) and CFL_IE0 (CCR[2]) (Falling latch Interrupt enable) to decide the condition of interrupt occur. Capture channel 1 has the same feature by setting CCR[17] and CCR[18], and etc. Whenever the Capture controller issues a capture interrupt, the corresponding BPWM counter will be reloaded with CNRn at this moment. Note that the corresponding GPIO pins must be configured as capture function (POE disabled and CAPENR enabled) for the corresponding capture channel.

The maximum captured frequency that BPWM 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 CRLRn/CFLRn (n=0, 1) 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.

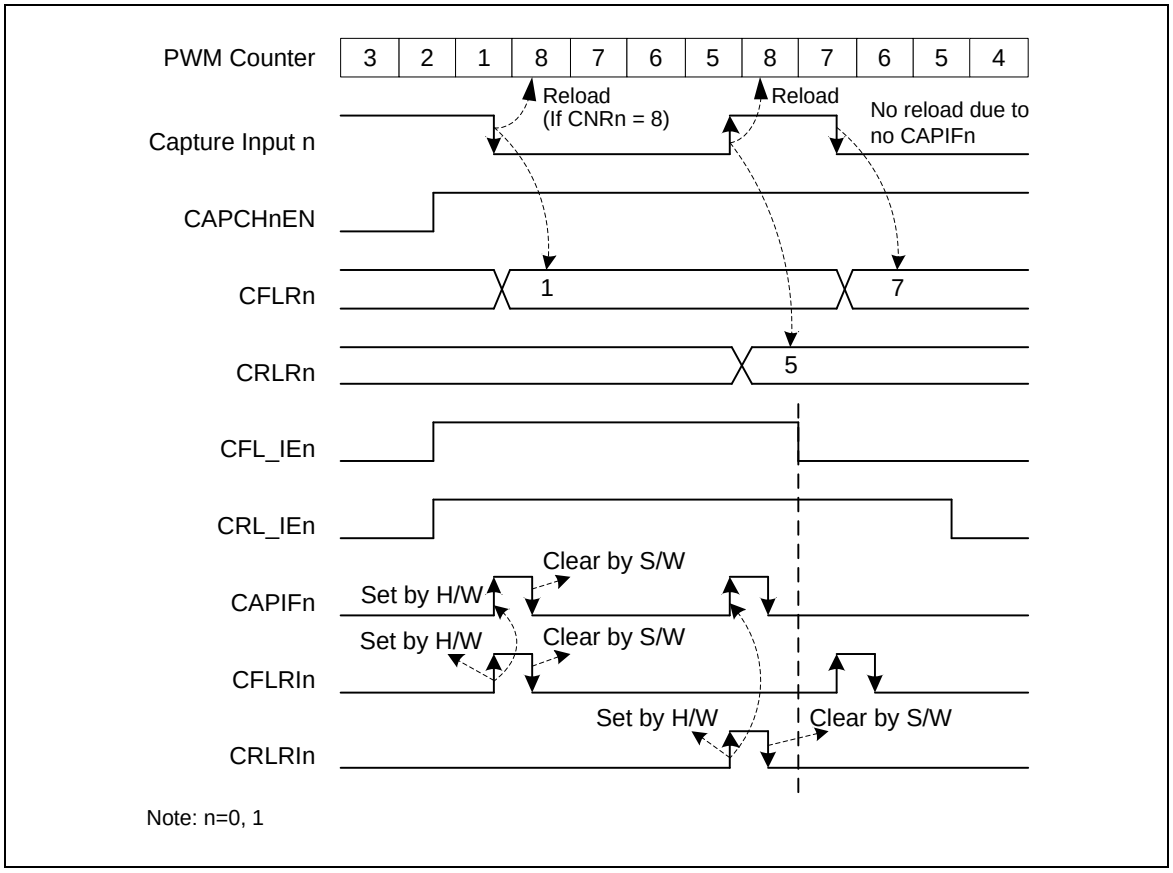


Figure 6-38 Capture Operation Timing

In this case, the CNR is 8:

1. The BPWM counter will be reloaded with CNRn when a capture interrupt flag (CAPIFn) is set.
2. The channel low pulse width is (CNR + 1 – CRLR).
3. The channel high pulse width is (CNR + 1 – CFLR).

6.7.5.9 BPWM-Timer Interrupt Architecture

There are two BPWM interrupts, BPCH0_INT and BPCH1_INT. BPWM output function 0 and capture input function 0 share one interrupt;BPWM output function 1 and capture input function 1 share the same interrupt. Therefore, BPWM function and Capture function in the same channel cannot be used at the same time. Figure 6-39demonstrates the architecture of BPWM Timer interrupts.

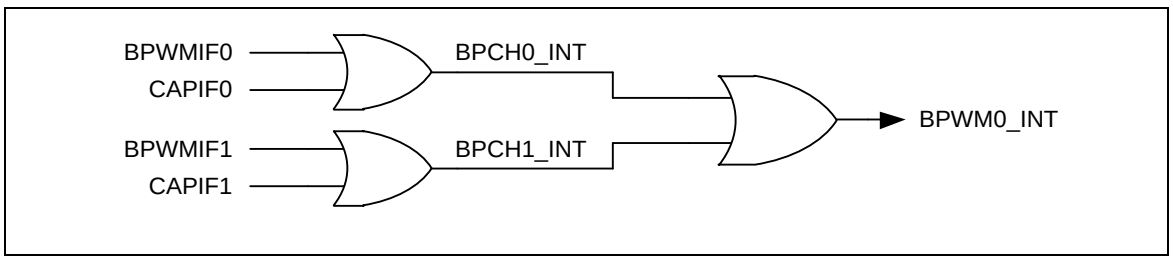


Figure 6-39BPWM Interrupt Architecture Diagram

6.7.5.10 BPWM-Timer Start Procedure

The following procedure is recommended for starting a BPWM drive.

1. Set clock source divider select register (CSR)
2. Set prescaler register (PPR)
3. Set inverter on/off, dead-zone generator on/off, auto-reload/one-shot mode and Stop BPWM-timer from BPWM control register (PCR)
4. Set comparator register (CMR) for setting BPWM duty.
5. Set BPWM down-counter register (CNR) for setting BPWM period.
6. Set interrupt enable register (PIER) (optional)
7. Set the corresponding GPIO pins as BPWM function (set HZ_BPWM (PWMPOEN[4]) = 0, enable POE and disable CAPENR) for the corresponding BPWM channel.
8. Enable BPWM timer start running (Set ChnEN = 1 in PCR)

6.7.5.11 BPWM-Timer Re-Start Procedure in Single-shot mode

After BPWM waveform is generated once in BPWM One-shot mode, BPWM-Timer will be stopped automatically. The following procedure is recommended for re-starting BPWM single-shot waveform.

- Set comparator register (CMR) for setting BPWM duty.
- Set BPWM down-counter register (CNR) for setting BPWM period. After setting CNR, BPWM wave will be generated.

6.7.5.12 BPWM-Timer Stop Procedure

Method 1:

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

Method 2:

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

Method 3:

Disable BPWM-Timer directly ((ChnEN in PCR). ***(Not recommended)***

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

6.7.5.13 Capture Start Procedure

1. Set clock source divider select register (CSR)

2. Set prescaler register (PPR)
3. Set channel enabled, rising/falling interrupt enable and input signal inverter on/off (CCR)
4. Set auto-reload mode, edge-aligned type and stop BPWM-timer (PCR)
5. Set BPWM down-counter (CNR)
6. Enable BPWM timer start running (Set ChnEN = 1 in PCR)
7. Set corresponding GPIO pins as capture function (set HZ_BPWM (PWMPOEN[4]) = 0, disable POE and enable CAPENR) for the corresponding BPWM channel.

6.7.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
BPWM0 Base Address: **BPWM0_BA = 0x4004_0000**				
PPR	BPWM0_BA+0x00	R/W	BPWM0 Prescaler Register	0x0000_0000
CSR	BPWM0_BA+0x04	R/W	BPWM0 Clock Source Divider Select Register	0x0000_0000
PCR	BPWM0_BA+0x08	R/W	BPWM0 Control Register	0x0000_0000
CNR0	BPWM0_BA+0x0C	R/W	BPWM0 Counter Register 0	0x0000_0000
CMR0	BPWM0_BA+0x10	R/W	BPWM0 Comparator Register 0	0x0000_0000
PDR0	BPWM0_BA+0x14	R	BPWM0 Data Register 0	0x0000_0000
CNR1	BPWM0_BA+0x18	R/W	BPWM0 Counter Register 1	0x0000_0000
CMR1	BPWM0_BA+0x1C	R/W	BPWM0 Comparator Register 1	0x0000_0000
PDR1	BPWM0_BA+0x20	R	BPWM0 Data Register 1	0x0000_0000
PIER	BPWM0_BA+0x40	R/W	BPWM0 Interrupt Enable Register	0x0000_0000
PIIR	BPWM0_BA+0x44	R/W	BPWM0 Interrupt Indication Register	0x0000_0000
CCR	BPWM0_BA+0x50	R/W	BPWM0 Capture Control Register	0x0000_0000
CRLR0	BPWM0_BA+0x58	R	BPWM0 Capture Rising Latch Register (Channel 0)	0x0000_0000
CFLR0	BPWM0_BA+0x5C	R	BPWM0 Capture Falling Latch Register (Channel 0)	0x0000_0000
CRLR1	BPWM0_BA+0x60	R	BPWM0 Capture Rising Latch Register (Channel 1)	0x0000_0000
CFLR1	BPWM0_BA+0x64	R	BPWM0 Capture Falling Latch Register (Channel 1)	0x0000_0000
CAPENR	BPWM0_BA+0x78	R/W	BPWM0 Capture Input Enable Register	0x0000_0000
POE	BPWM0_BA+0x7C	R/W	BPWM0 Output Enable	0x0000_0000

6.7.7 Register Description

BPWM0 Prescaler Register (PPR)

Register	Offset	R/W	Description	Reset Value
PPR	BPWM0_BA+0x00	R/W	BPWM0 Prescaler Register	0x0000_0000

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

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:16]	**DZI01**	**Dead-zone Interval for Pair of Channel 0and Channel 1** These 8-bit determine the Dead-zone length. The unit time of dead-zone length = [(prescale+1)*(clock source divider)]/ BPWM0_CLK.
[15:8]	**Reserved**	Reserved.
[7:0]	**CP01**	**Clock Prescaler** Clock input is divided by (CP01 + 1) before it is fed to the corresponding BPWM-timer If CP01=0, then the clock prescaler 0 output clock will be stopped. So corresponding BPWM-timer will also be stopped.

BPWM0 Clock Source Divider Select Register (CSR)

Register	Offset	R/W	Description	Reset Value
CSR	BPWM0_BA+0x04	R/W	BPWM0 Clock Source Divider 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	CSR1			Reserved	CSR0		

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6:4]	**CSR1**	**BPWM Timer 1 Clock Source Divider Selection** Select clock source divider for BPWM timer 1. 000 = Input clock divided by 2. 001 = Input clock divided by 4. 010 = Input clock divided by 8. 011 = Input clock divided by 16. 100 = Input clock divided by 1.
[3]	**Reserved**	Reserved.
[2:0]	**CSR0**	**BPWM Timer 0 Clock Source Divider Selection** Select clock source divider for BPWM timer 0, please refer to CSR1

BPWM0 Control Register (PCR)

Register	Offset	R/W	Description	Reset Value
PCR	BPWM0_BA+0x08	R/W	BPWM0Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved	PWM01TYPE	Reserved					
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved				CH1MOD	CH1INV	CH1PINV	CH1EN
7	6	5	4	3	2	1	0
Reserved			DZEN01	CH0MOD	CH0INV	CH0PINV	CH0EN

Bits	Description	
[31]	**Reserved**	Reserved.
[30]	**PWM01TYPE**	**BPWM0_CH0/1 Aligned Type Selection Bit** 0 = Edge-aligned type. 1 = Center-aligned type.
[29:12]	**Reserved**	Reserved.
[11]	**CH1MOD**	**BPWM-timer 1 Auto-reload/One-shot Mode** 0 = One-shot mode. 1 = Auto-reload mode. **Note:** If there is a transition at this bit, it will cause CNR1 and CMR1 be cleared.
[10]	**CH1INV**	**BPWM-timer 1 Output Inverter Enable Bit** 0 = Inverter Disabled. 1 = Inverter Enabled.
[9]	**CH1PINV**	**BPWM-timer 1 Output Polar Inverse Enable Bit** 0 = BPWM0_CH1 output polar inverse Disabled. 1 = BPWM0_CH1 output polar inverse Enabled.
[8]	**CH1EN**	**BPWM-timer 1 Enable Bit** 0 = Corresponding BPWM-Timer Stopped. 1 = Corresponding BPWM-Timer Start Running.
[7:5]	**Reserved**	Reserved.
[4]	**DZEN01**	**Dead-zone 0 Generator Enable Bit** 0 = Dead-zone 0 generatorDisabled. 1 = Dead-zone 0 generator Enabled. **Note:** When Dead-zone generator is enabled, the pair of BPWM0_CH0 and BPWM0_CH1 becomes a complementary pair.
[3]	**CH0MOD**	**BPWM-timer 0 Auto-reload/One-shot Mode**

		0 = One-shot mode. 1 = Auto-reload mode. **Note:** If there is a transition at this bit, it will cause CNR0 and CMR0 be cleared.
[2]	**CH0INV**	**BPWM-timer 0 Output Inverter Enable Bit** 0 = Inverter Disabled. 1 = Inverter Enabled.
[1]	**CH0PINV**	**BPWM-timer 0 Output Polar Inverse Enable Bit** 0 = BPWM0_CH0 output polar inverse Disabled. 1 = BPWM0_CH0 output polar inverse Enabled.
[0]	**CH0EN**	**BPWM-timer 0 Enable Bit** 0 = The corresponding BPWM-Timer stopsrunning. 1 = The corresponding BPWM-Timer startsrunning.

BPWM0 Counter Register 0-1 (CNR0-1)

Register	Offset	R/W	Description	Reset Value
CNR0	BPWM0_BA+0x0C	R/W	BPWM0Counter Register 0	0x0000_0000
CNR1	BPWM0_BA+0x18	R/W	BPWM0Counter 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**
CNR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CNR							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CNR**	**BPWM Timer Loaded Value** CNR determines the BPWM period. BPWM frequency = BPWM0_CLK/[(prescale+1)*(clock divider)*(CNR+1)]. For Edge-aligned type: ● Duty ratio = (CMR+1)/(CNR+1). ● CMR >= CNR: BPWM output is always high. ● CMR < CNR: BPWM low width = (CNR-CMR) unit; BPWM high width = (CMR+1) unit. ● CMR = 0: BPWM low width = (CNR) unit; BPWM high width = 1 unit. For Center-aligned type: ● Duty ratio = [(2 x CMR) + 1]/[2 x (CNR+1)]. ● CMR > CNR: BPWM output is always high. ● CMR <= CNR: BPWM low width = 2 x (CNR-CMR) + 1 unit; BPWM high width = (2 x CMR) + 1 unit. ● CMR = 0: BPWM low width = 2 x CNR + 1 unit; BPWM high width = 1 unit. (Unit = one BPWM clock cycle). **Note:** Any write to CNR will take effect in next BPWM cycle. **Note:** When BPWM operating at Center-aligned type, CNR value should be set between 0x0000 to 0xFFFE. If CNR equal to 0xFFFF, the BPWM will work unpredictable. **Note:**When CNR value is set to 0, BPWM output is always high.

BPWM0 Comparator Register 0-1 (CMR0-1)

Register	Offset	R/W	Description	Reset Value
CMR0	BPWM0_BA+0x10	R/W	BPWM0 Comparator Register 0	0x0000_0000
CMR1	BPWM0_BA+0x1C	R/W	BPWM0 Comparator 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**
CMR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CMR							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CMR**	**BPWM Comparator Register** CMR determines the BPWM duty. BPWM frequency = BPWM_CLK/[(prescale+1)*(clock divider)*(CNR+1)]. For Edge-aligned type: ● Duty ratio = (CMR+1)/(CNR+1). ● CMR >= CNR: BPWM output is always high. ● CMR < CNR: BPWM low width = (CNR-CMR) unit; BPWM high width = (CMR+1) unit. ● CMR = 0: BPWM low width = (CNR) unit; BPWM high width = 1 unit. For Center-aligned type: ● Duty ratio = [(2 x CMR) + 1]/[2 x (CNR+1)]. ● CMR > CNR: BPWM output is always high. ● CMR <= CNR: BPWM low width = 2 x (CNR-CMR) + 1 unit; BPWM high width = (2 x CMR) + 1 unit. ● CMR = 0: BPWM low width = 2 x CNR + 1 unit; BPWM high width = 1 unit. (Unit = one BPWM clock cycle). **Note:** Any write to CMR will take effect in next BPWM cycle.

BPWM0 Data Register 0-1 (PDR 0-1)

Register	Offset	R/W	Description	Reset Value
PDR0	BPWM0_BA+0x14	R	BPWM0 Data Register 0	0x0000_0000
PDR1	BPWM0_BA+0x20	R	BPWM0 Data Register 1	0x0000_0000

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

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

BPWM0 Interrupt Enable Register (PIER)

Register	Offset	R/W	Description	Reset Value
PIER	BPWM0_BA+0x40	R/W	BPWM0 Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							INTTYPE
15	14	13	12	11	10	9	8
Reserved						BPWMDIE1	BPWMDIE0
7	6	5	4	3	2	1	0
Reserved						BPWMPIE1	BPWMPIE0

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**INTTYPE**	**BPWM Interrupt Period Type Selection Bit** 0 = BPWMIFn will be set if BPWM counter underflow. 1 = BPWMIFn will be set if BPWM counter matches CNRn register. **Note:** This bit is effective when BPWM in Center-aligned type only.
[15:10]	**Reserved**	Reserved.
[9]	**BPWMDIE1**	**BPWM Channel 1 Duty Interrupt Enable Bit** 0 = BPWM0_CH1 duty interrupt Disabled. 1 = BPWM0_CH1 duty interrupt Enabled.
[8]	**BPWMDIE0**	**BPWM Channel 0 Duty Interrupt Enable Bit** 0 = BPWM0_CH0 duty interruptDisabled. 1 = BPWM0_CH0 duty interrupt Enabled.
[7:2]	**Reserved**	Reserved.
[1]	**BPWMPIE1**	**BPWM Channel 1 Period Interrupt Enable Bit** 0 = BPWM0_CH1 period interruptDisabled. 1 = BPWM0_CH1 period interrupt Enabled.
[0]	**BPWMPIE0**	**BPWM Channel 0 Period Interrupt Enable Bit** 0 = BPWM0_CH0 period interruptDisabled. 1 = BPWM0_CH0 period interrupt Enabled.

BPWM0 Interrupt Indication Register (PIIR)

Register	Offset	R/W	Description	Reset Value
PIIR	BPWM0_BA+0x44	R/W	BPWM0 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						BPWMDIF1	BPWMDIF0
7	6	5	4	3	2	1	0
Reserved						BPWMIF1	BPWMIF0

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**BPWMDIF1**	**BPWM Channel 1 Duty Interrupt Flag** Flag is set by hardware when BPWM0_CH1 counter down count and reaches CMR1, software can clear this bit by writing a one to it. **Note:** If CMR equal to CNR, this flag is not working in Edge-aligned type selection
[8]	**BPWMDIF0**	**BPWM Channel 0 Duty Interrupt Flag** Flag is set by hardware when BPWM0_CH0 counter down count and reaches CMR0, software can clear this bit by writing a one to it. **Note:**If CMR equal to CNR, this flag is not working in Edge-aligned type selection
[7:2]	**Reserved**	Reserved.
[1]	**BPWMIF1**	**BPWM Channel 1 Period Interrupt Flag** This bit is set by hardware when BPWM0_CH1 counter reaches the requirement of interrupt (depend on INTTYPE bit of PIER register), software can write 1 to clear this bit to 0.
[0]	**BPWMIF0**	**BPWM Channel 0 Period Interrupt Flag** This bit is set by hardware when BPWM0_CH0 counter reaches the requirement of interrupt (depend on INTTYPE bit of PIER register), software can write 1 to clear this bit to 0.

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

BWM0 Capture Control Register (CCR)

Register	Offset	R/W	Description	Reset Value
CCR	BPWM0_BA+0x50	R/W	BPWM0 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 BPWM0 input channel 1 has a falling transition, CFLR1 was latched with the value of BPWM0 down-counter and this bit is set by hardware. Software can write 1 to clear this bit to 0
[22]	**CRLRI1**	**CRLR1 Latched Indicator Bit** When BPWM0 input channel 1 has a rising transition, CRLR1 was latched with the value of BPWM0 down-counter and this bit is set by hardware. Software can write 1 to clear this bit to0.
[5]	**Reserved**	Reserved.
[20]	**CAPIF1**	**Channel 1 Capture Interrupt Indication Flag** If BPWM0_CH1 rising latch interrupt is enabled (CRL_IE1=1), a rising transition occurs at BPWM0_CH1 will result in CAPIF1 to high; Similarly, a falling transition will cause CAPIF1 to be set high if BPWM0_CH1 falling latch interrupt is enabled (CFL_IE1=1). Write 1 to clear this bit to 0.
[19]	**CAPCH1EN**	**Channel 1 Capture Function Enable Bit** 0 = Capture function on BPWM0_CH1 Disabled. 1 = Capture function on BPWM0_CH1 Enabled. When Enabled, Capture latched the BPWM0-counter and saved to CRLR (Rising latch) and CFLR (Falling latch). When disabled, capture does not update CRLR and CFLR, and disable BPWM0_CH1 interrupt.
[18]	**CFL_IE1**	**Channel 1 Falling Latch Interrupt Enable Bit** 0 = Falling latch interrupt Disabled. 1 = Falling latch interrupt Enabled. When enabled, if capture detects BPWM0_CH1 has falling transition, capture will issue an interrupt.
[17]	**CRL_IE1**	**Channel 1 Rising Latch Interrupt Enable Bit** 0 = Rising latch interrupt Disabled.

		1 = Rising latch interrupt Enabled. When enabled, if capture detects BPWM0_CH1 has rising transition, capture will issue an interrupt.
[16]	**INV1**	**Channel 1 Inverter Enable Bit** 0 = Inverter Disabled. 1 = Inverter Enabled. Reverse the input signal from GPIO before fed to capture timer
[15:8]	**Reserved**	Reserved.
[7]	**CFLRI0**	**CFLR0 Latched Indicator Bit** When BPWM0 input channel 0 has a falling transition, CFLR0 was latched with the value of BPWM0 down-counter and this bit is set by hardware. Software can write 1 to clear this bit to0.
[6]	**CRLRI0**	**CRLR0 Latched Indicator Bit** When BPWM0 input channel 0 has a rising transition, CRLR0 was latched with the value of BPWM0 down-counter and this bit is set by hardware. Software can write 1 to clear this bit to 0.
[5]	**Reserved**	Reserved.
[4]	**CAPIF0**	**Channel 0 Capture Interrupt Indication Flag** If BPWM0_CH0 rising latch interrupt is enabled (CRL_IE0=1), a rising transition occurs at BPWM0_CH0 will result in CAPIF0 to high; Similarly, a falling transition will cause CAPIF0 to be set high if BPWM0_CH0 falling latch interrupt is enabled (CFL_IE0=1). Write 1 to clear this bit to 0.
[3]	**CAPCH0EN**	**Channel 0 Capture Function Enable Bit** 0 = Capture function on BPWM0_CH0 Disabled. 1 = Capture function on BPWM0_CH0 Enabled. When enabled, capture latched the BPWM0-counter value and saved to CRLR (Rising latch) and CFLR (Falling latch). When disabled, capture does not update CRLR and CFLR, and disable BPWM0 channel 0 Interrupt.
[2]	**CFL_IE0**	**Channel 0 Falling Latch Interrupt Enable Bit** 0 = Falling latch interrupt Disabled. 1 = Falling latch interrupt Enabled. When Enabled, if Capture detects BPWM0 channel 0 has falling transition, Capture will issue an Interrupt.
[1]	**CRL_IE0**	**Channel 0 Rising Latch Interrupt Enable Bit** 0 = Rising latch interrupt Disabled. 1 = Rising latch interrupt Enabled. When enabled, if capture detects BPWM0 channel 0 has rising transition, capture will issue an interrupt.
[0]	**INV0**	**Channel 0 Inverter Enable Bit** 0 = Inverter Disabled. 1 = Inverter Enabled. Reverse the input signal from GPIO before fed to capture timer

BPWM0 Capture Rising Latch Register 0-1 (CRLR0-1)

Register	Offset	R/W	Description	Reset Value
CRLR0	BPWM0_BA+0x58	R	BPWM0 Capture Rising Latch Register (Channel 0)	0x0000_0000
CRLR1	BPWM0_BA+0x60	R	BPWM0 Capture Rising Latch Register (Channel 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**
CRLR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CRLR							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CRLR**	**Capture Rising Latch Register** Latch the BPWM0 counter when Channel 0/1 has rising transition.

BPWM0 Capture Falling Latch Register 0-1 (CFLR0-1)

Register	Offset	R/W	Description	Reset Value
CFLR0	BPWM0_BA+0x5C	R	BPWM0 Capture Falling Latch Register (Channel 0)	0x0000_0000
CFLR1	BPWM0_BA+0x64	R	BPWM0 Capture Falling Latch Register (Channel 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**
CFLR							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
CFLR							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CFLR**	**Capture Falling Latch Register** Latch the BPWM0 counter when Channel 0/1 has Falling transition.

BPWM0 Capture Input Enable Register (CAPENR)

Register	Offset	R/W	Description	Reset Value
CAPENR	BPWM0_BA+0x78	R/W	BPWM0 Capture Input Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**CINEN1**	**CINEN0**

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**CINEN1**	**Channel 1 Capture Input Enable Bit** 0 = BPWM0_CH1 capture input path Disabled. The input of BPWM0_CH1 capture function is always regarded as 0. 1 = BPWM0_CH1 capture input path Enabled. The input of BPWM0_CH1 capture function comes from correlative multifunction pin if GPIO multi-function is set as BPWM0_CH1.
[0]	**CINEN0**	**Channel 0 Capture Input Enable Bit** 0 = BPWM0_CH0 capture input path Disabled. The input of BPWM0_CH0 capture function is always regarded as 0. 1 = BPWM0_CH0 capture input path Enabled. The input of BPWM0_CH0 capture function comes from correlative multifunction pin if GPIO multi-function is set as BPWM0_CH0.

BPM0 Output Enable Register (POE)

Register	Offset	R/W	Description	Reset Value
POE	BPWM0_BA+0x7C	R/W	BPWM0 Output 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							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved						**POE1**	**POE0**

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**POE1**	**Channel 1 Output Enable Register** 0 = BPWM0_CH1 output to pin Disabled. 1 = BPWM0_CH1 output to pin Enabled. **Note:** The corresponding GPIO pin must also be switched to BPWM0 function
[0]	**POE0**	**Channel 0 Output Enable Register** 0 = BPWM0_CH0 output to pin Disabled. 1 = BPWM0_CH0 output to pin Enabled. **Note:** The corresponding GPIO pin must also be switched to BPWM0 function

6.8 Enhanced PWM Generator (EPWM)

6.8.1 Overview

This device has two built-in PWM units with the same architecture whose function is specially designed for driving motor control applications.

6.8.2 Features

Each unit supports the features below:

- Three independent 16-bit PWM duty control units with maximum 6 port pins:
  - 3 independent PWM output:
    EPWM0_CH0, EPWM0_CH2 and EPWM0_CH4 for Unit 0
    EPWM1_CH0, EPWM1_CH2 and EPWM1_CH4 for Unit 1
  - 3 complementary PWM pairs, with each pin in a pair mutually complement to each other and capable of programmable dead-time insertion:
    (EPWMx_CH0, EPWMx_CH1), (PWMx_CH2, EPWMx_CH3) and (EPWMx_CH4, EPWMx_CH5) where x=0~1.
  - 3 synchronous PWM pairs, with each pin in a pair in-phase:
    (EPWMx_CH0, EPWMx_CH1), (EPWMx_CH2, EPWMx_CH3) and (EPWMx_CH4, EPWMx_CH5) where x=0~1
- Group control bits:
  EPWMx_CH2 and EPWMx_CH4 are synchronized with EPWMx_CH0
- Supports Edge aligned mode and Center aligned mode
- Programmable dead-time insertion between complementary paired PWMs
- Each pin of EPWMx_CH0 to EPWMx_CH5 has independent polarity setting control
- Mask output control for Electrically Commutated Motor operation
- Tri-state output at reset and brake state
- Hardware brake protection
- Two Interrupt Sources:
  - Interrupt is synchronously requested at PWM frequency when up/down counter comparison matched (edge and center aligned modes) or underflow (center aligned mode).
  - Interrupt is requested when external brake pins asserted
- PWM signals before polarity control stage are defined in the view of positive logic. The PWM ports is active high or active low are controlled by polarity control register.
- High Source/Sink current.
- Supports trigger EADC
- Supports auto-load PWM duty by MDU

6.8.3 Block Diagram

After CPU reset, the internal output of the each PWM channels depends on the polarity setting. The interval between successive outputs is controlled by a 16–bit up/down counter which uses a software selectable clock source with configurable internal clock pre-scalar as its input. The PWM counter clock has the frequency as the clock source F_{PWM}= EPWMx_CLK/Pre-scalar; Here the EPWMx_CLK synchronized with CPU clock HCLK.

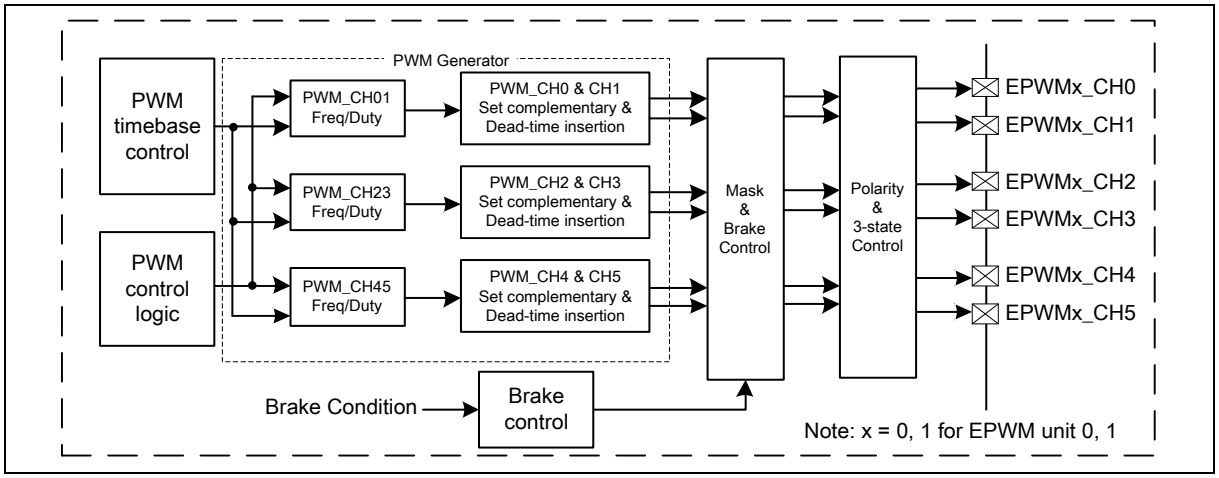


Figure 6–40EPWM Block Diagram

6.8.4 Basic Configuration

The EPWM pin functions are configured in P0_MFP and P1_MFP Multiple Function Registers.

The peripheral clock source of EPWM can be enabled in EPWM0_EN (APBCLK[20]) and EPWM1_EN (APBCLK[21]).

6.8.5 Function Description

The following diagrams show the EPWM clock source control and PWM time-base generator.

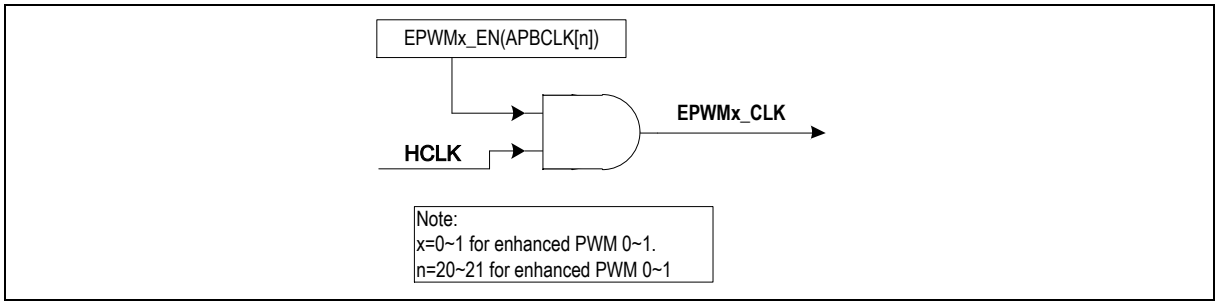


Figure 6–41EPWM Clock Source Control

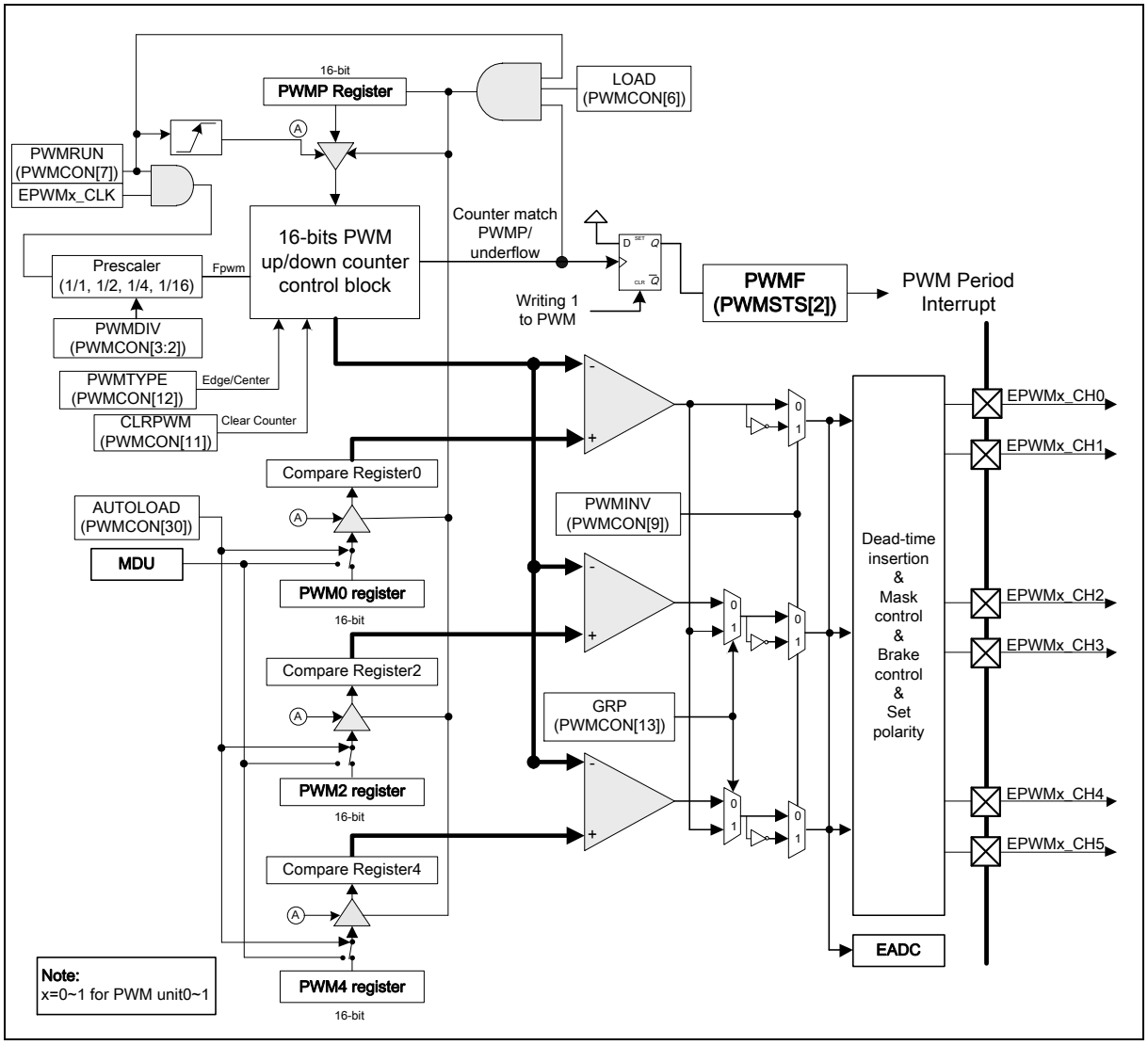


Figure 6–42 PWM Time-base Generator

The overall functioning of the PWM module is controlled by the contents of the PWMCONx0, PWMB and PWMCON registers. The operation of most of the control bits is straightforward. PWMRUN(PWMCON[7]) allows the PWM to be either in the run or idle state. The transfer of the data from the PWMP register to 16-bit PWM counter will occur on the rising edge of PWMRUN or during PWMRUN with LOAD(PWMCON[6]) and PWM counter match/underflow occurs. The transfer of the data from the PWMn registers to the compare registers is controlled by the Load bit (PWMCON[6]) (with the condition that PWMRUN = 1 and match/underflow occurs).

A compare value greater than the counter reloaded value resulted in the PWM output being permanently high. In addition there are two special cases. If compare register is set to 0x0000, the PWM_CHn output will stay at low, and if compare register is set to 0xFFFF, the PWM_CHn output will stuck at high until there is a change in the compare register. [n = 0-5].

During ICP, ISP or ICE mode, PWM pins will be tri-stated. PWM operation will stop and module reset. When exit from ICP, ISP or ICE mode, the PWM pins will follow the control register settings.

In ICE mode, the condition which causes CPU stops or pauses running will force the PWM_CHn output pins in tri-stated, when CPU runs the PWM_CHn pins will follow the control register settings.

When a PWM period value is written to the PWMP register by software, the value is saved in the holding register first, and thenthe value of the holding register will be reloaded to the actual PWM period when the following conditions are met:LOAD (PWMCON[6])= 1, PWMRUN (PWMCON[7]) = 1 and PWM match/underflow. The width of each PWM output pulse is determined by the value in the appropriate compare register. Each PWM registers ofPWMP, PWM0, PWM2 and PWM4, in the format of 16-bit width, decides the PWM period and each channel's duty cycle. If the PWMINV (PWMCON[9]) (Inverse PWM Comparator Output) is set to high the PWM comparator output signals will be inversed, therefore the PWM Duty (in percentage) is changed to (1-Duty) and PWM Duty registers, PWM0/2/4 represent Duty-off time.

Note I duty registers PWM0/2/4 and the period registers PWMP are double-buffered registers used to set the duty cycle and counting period for the PWM time base respectively. For the 1st buffer it is accessible by user while the 2nd buffer holds the actual compare value used in the present period. Load bit must be set to 1 to enable the 1st buffer value to be loaded in to the 2nd buffer register when counter underflow/match.

6.8.5.1 PWM Operation Mode

This device supports two operation modes: Edge-aligned and Center-aligned mode.

The following equations show the formula for period and duty for each PWM operation mode:

Edge-aligned:

Period = (PWMP +1)* EPWMx_CLK period/pre-scalar

Duty = (duty + 1) * EPWMx_CLK period/pre-scalar

Center-aligned:

Period = (PWMP* 2)* EPWMx_CLK period/pre-scalar

Duty = (duty*2 + 1) * EPWMx_CLK period/pre-scalar

Note: "duty" refers to PWM0/2/4/6 register value.

Edge aligned PWM (Up Counter)

In Edge-aligned PWM output mode, the 16-bits PWM counter will start counting from zero to period value (PWMP register) to finish a PWM period, then restart counting from zero to period value again.The value of PWM counter will be compared with duty value (PWMn register) to control output level of PWM generator.The PWM generator will output low when the value of PWM counter is larger than duty value and output high when the value of PWM counter is equal or smaller than duty value.

The PWM period interrupt (PWMF (PWMSTS[2])) will be triggered by setting PWMI_En (PWMCON[4]) to 1, and PWM edge interrupt(PWMnEF(PWMSTS[6:4])) will be triggered by setting PWMnEI_EN(PWMEIC[2:0]) to 1.

The PWM period and duty control are configured by PWM period register (PWMP) and PWM duty register (PWMn). The new period and duty value will take effect at the start of next period if LOAD (PWMCON[6]) is set to 1.

Figure 6–43, Figure 6–44and Figure 6–45depict the Edge-aligned PWM timing and operation flow.

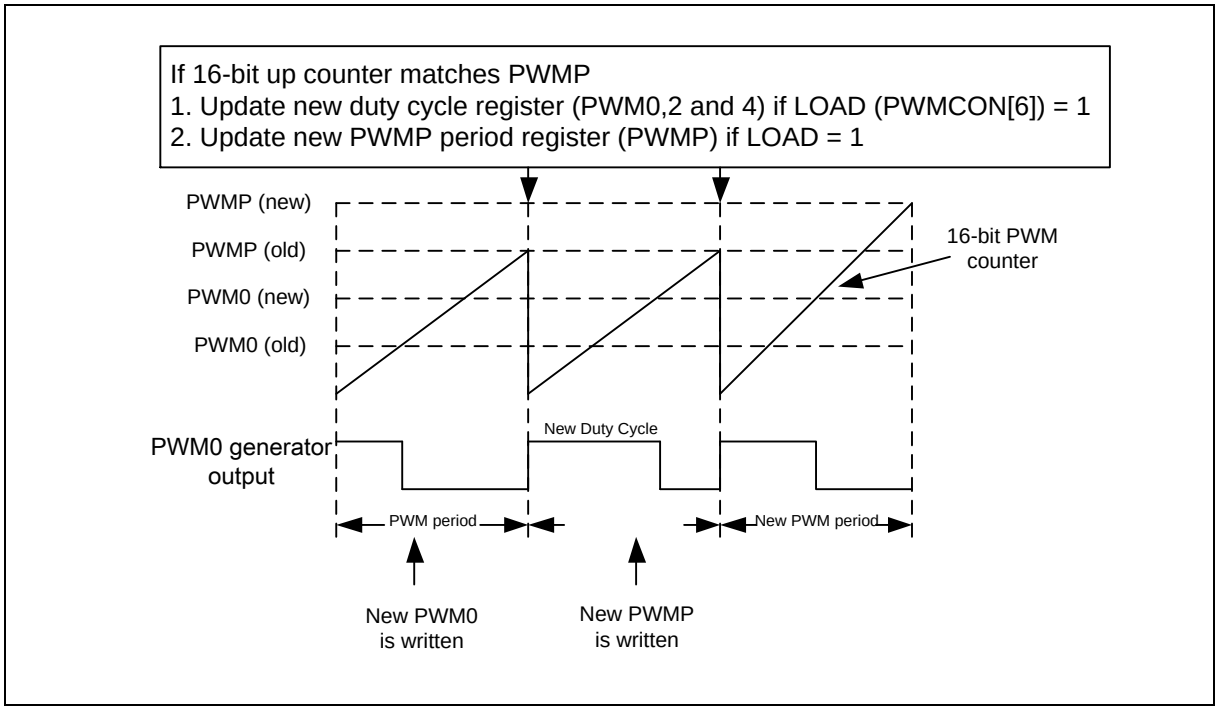


Figure 6–43 Edge-aligned PWM

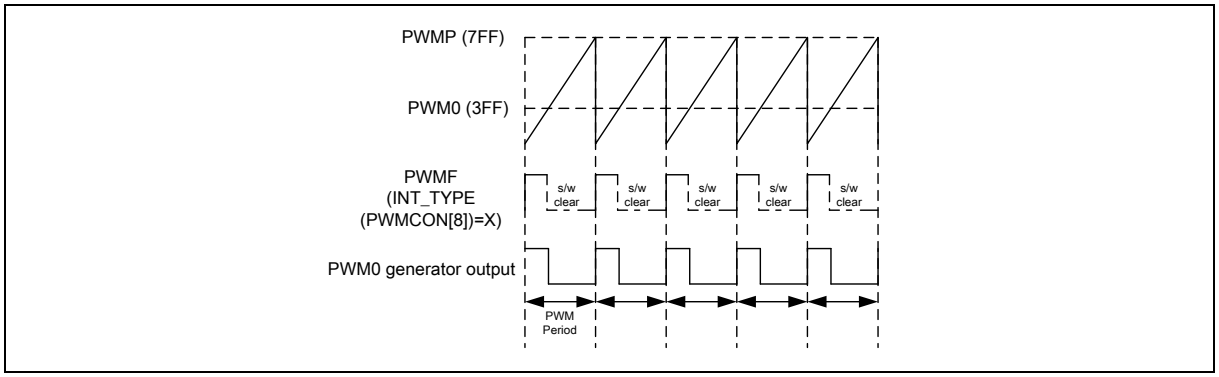


Figure 6–44 PWM0 Edge aligned Waveform Output

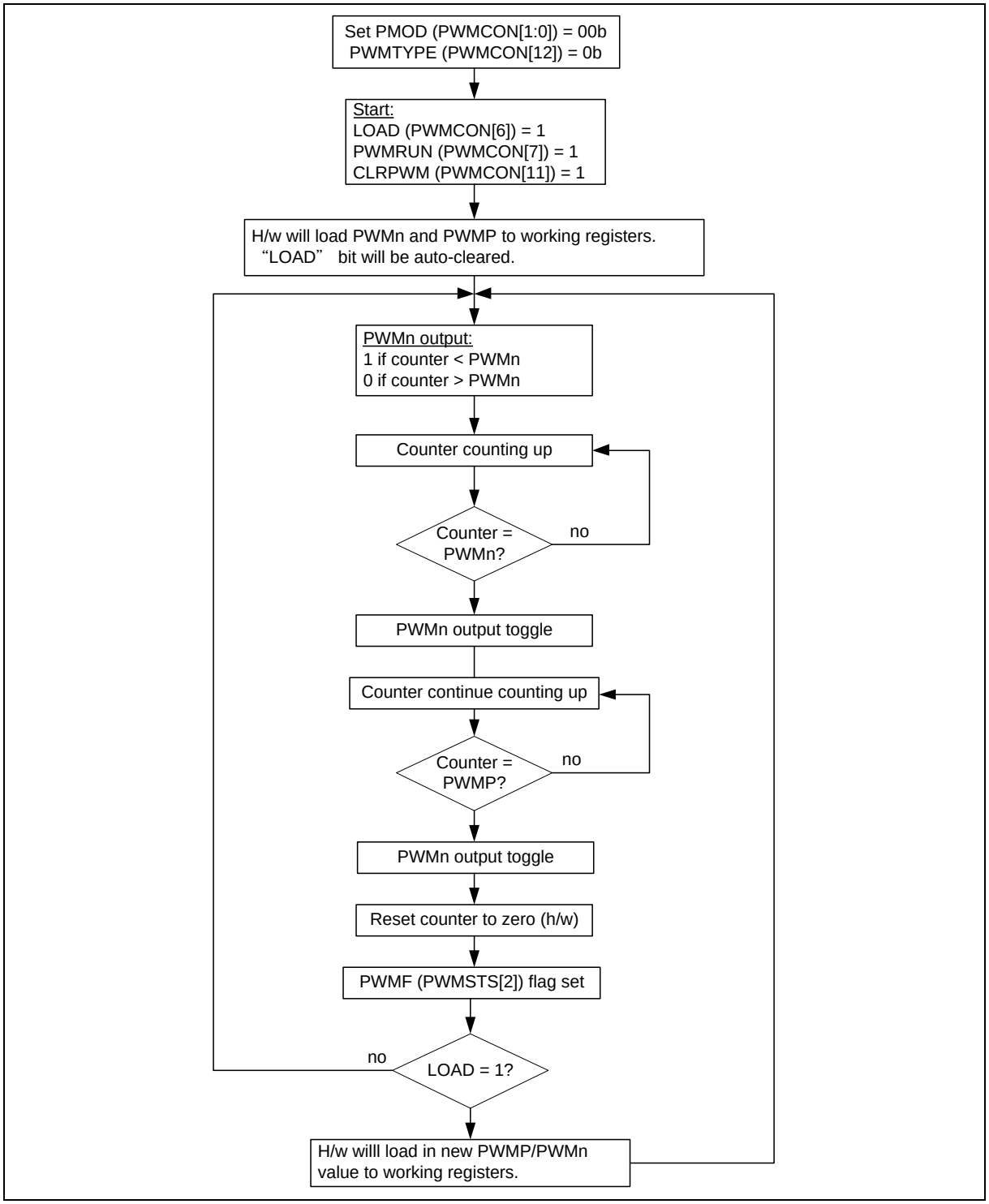


Figure 6–45 Edge-aligned Flow Diagram

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 zero to the value of PWMP register and then start counting down to zero to finish a PWM period, then restart next PWM period again.The value of PWM counter will be compared with duty value (PWMn register) to control output level of PWM generator. The PWM generator will output low when the value of PWM counter is larger than duty value and output high when the value of PWM counter is equal or smaller than comparator value.Once the PWM counter underflows the new period and duty value will take effect if LOAD (PWMCON[6]) is set to 1.

In Center-aligned mode, the PWM interrupt is requested at down-counter underflow if INT_TYPE (PWMCON0[8]) =0, i.e. at start (end) of each PWM cycle or at up-counter matching with PWMP if INT_TYPE (PWMCON0[0]) =1, i.e. at center point of PWM cycle.

Figure 6–46, Figure 6–47and Figure 6–48depict the Center-aligned PWM timing and operation flow.

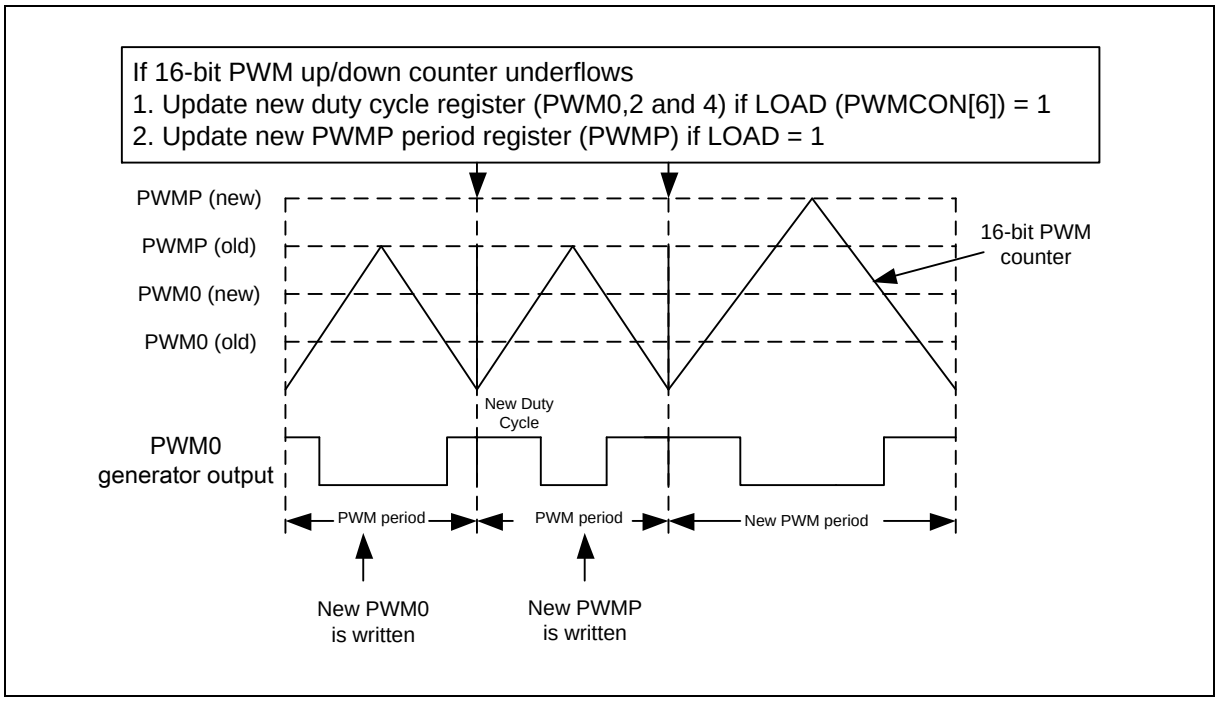


Figure 6–46 Center-aligned Mode

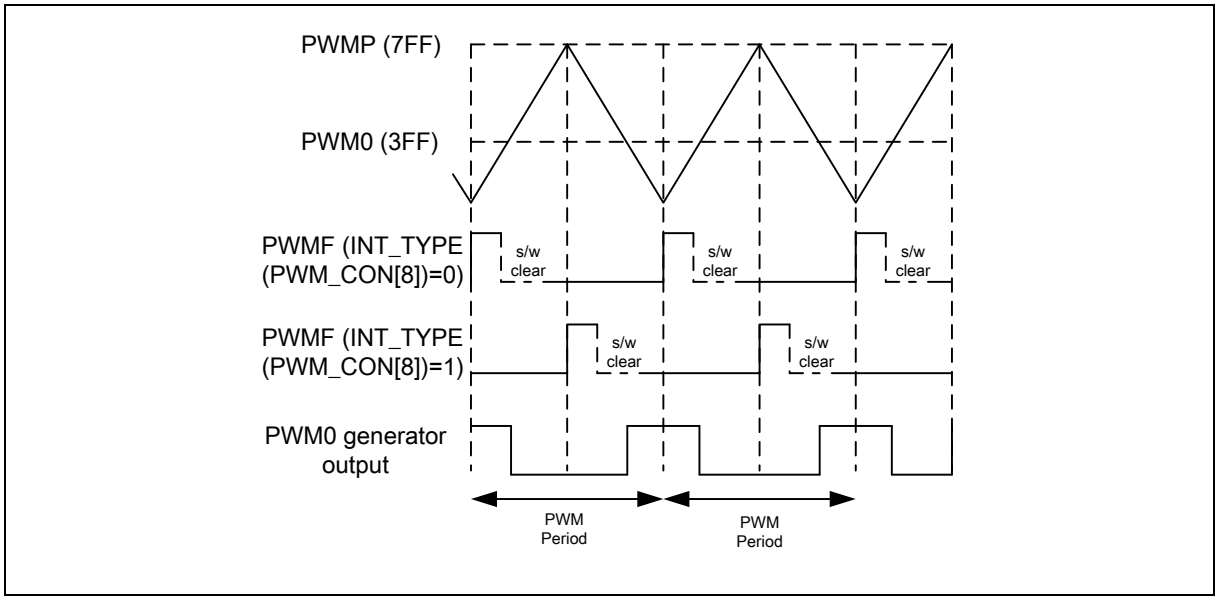


Figure 6–47 Example PWM0 Center-aligned Waveform Output

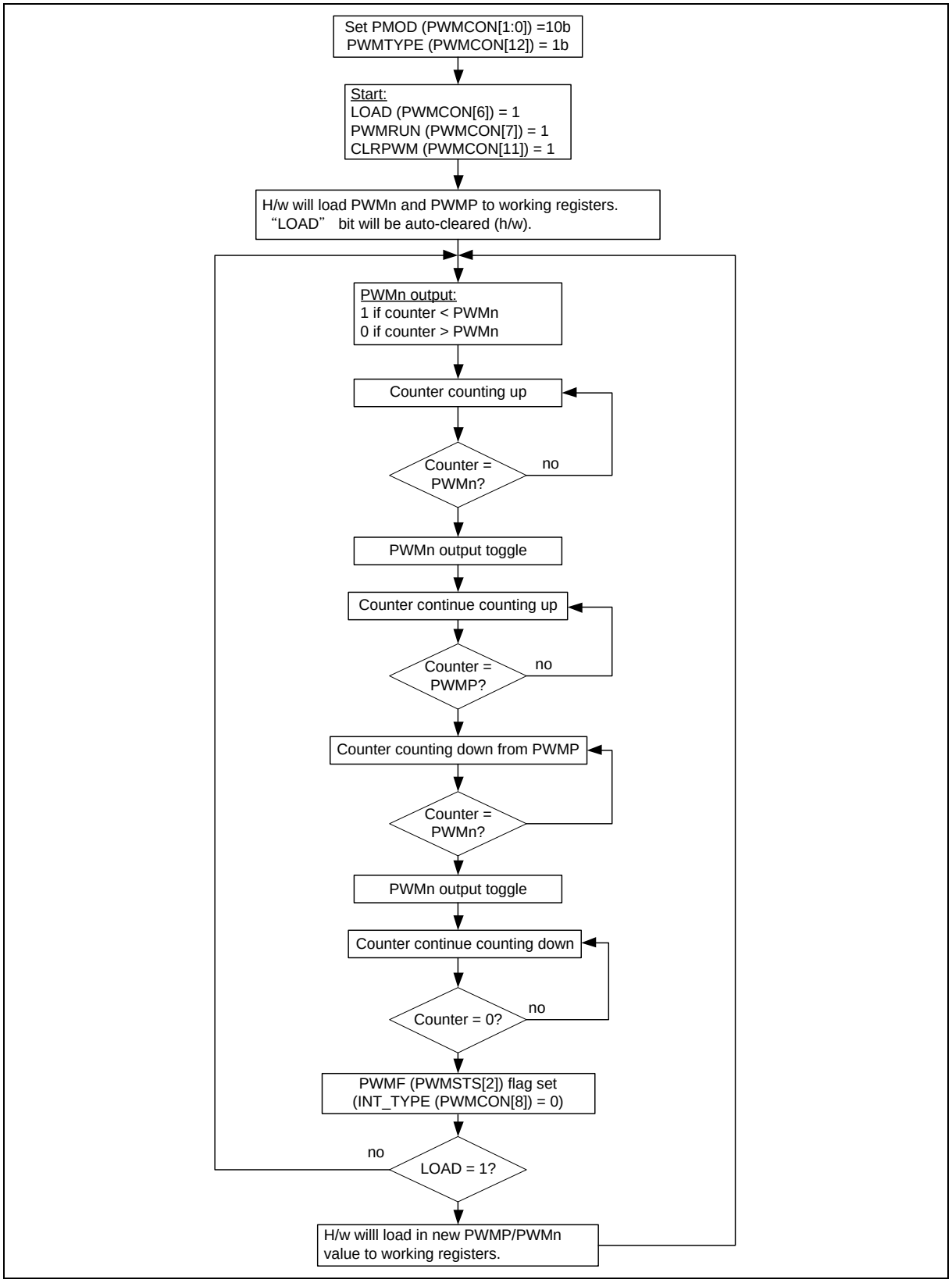


Figure 6–48 Center-aligned Flow Diagram (INT_TYPE = 0)

6.8.5.2 PWM Brake

This device supportstwo brake detectors, BK0 and BK1, and each of them has 4 brake signals, one external brake pin (EPWMx_BRAKE0 for BK0 andEPWMx_BRAKE1 for BK1), and three analog comparator outputs. Both external brake pins have each 4-degree digital filter that is user controllable through BKxFILT (PWMCON[23:20]) (x=0 and 1 for BK0 and BK1). The Brake function is controlled by the contents of PWMCON register.

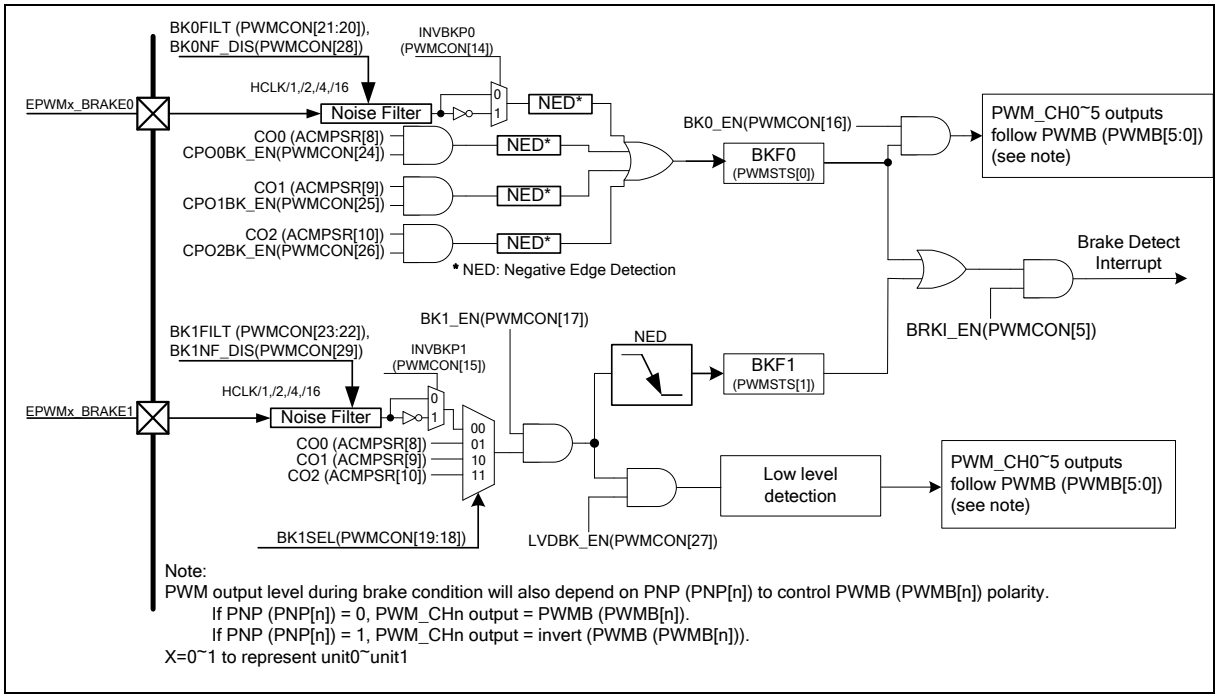


Figure 6–49 PWM Brake Function

The Table 6-10summarizesthe effect of each brake pins.Figure 6–50 and Figure 6–51illustrate the brake signals vs. PWM operation.

Brake Source	Brake Trigger Condition	Actions
BKP0 (I/O pin); CO0 (ACMPSR[8]) (Comparator0 output); CO1 (ACMPSR[9]) (Comparator1 output); CO2 (ACMPSR[10]) (Comparator2 output)	A falling edge at brake 0 edge detector.	● BKF0 (PWMSTS[0]) and BKLK0 (PWMSTS[8]) flags set by the falling edge at brake 0 edge detector; but cleared by S/W. ● PWM_CH0~5 outputs follow PWMB (PWMB[n]) bits. PNP (PNP[n]) bits are also able to control the polarity of PWMB. If PNP (PNP[n])=0, PWMn pin output = PWMB (PWMB[n]). If PNP (PNP[n])=1, PWMn pin output = invert(PWMB (PWMB[n])). ● PWM_CH0~5 output will keep in PWMB(PWMB[n]) state before BKLK0 (PWMSTS[8]) flag is cleared by S/W. ● PWMRUN (PWMCON[7]) bit remain asserted to keep PWM generators running. ● If the BKLK0 (PWMSTS[8]) flag is cleared, the brake state will be released on next PWM cycle/period. ● If PWMRUN (PWMCON[7]) is cleared to 0 before the BKLK0 (PWMSTS[8]) flag is cleared, PWM_CH0~5 output will remain in PWMB (PWMB[5:0]) state.

BKP1 (I/O pin); CO0 (ACMPSR[8]) (Comparator0 output); CO1 (ACMPSR[9]) (Comparator1 output); CO2 (ACMPSR[10]) (Comparator2 output)	A falling edge at brake 1 edge detector; Low level state	1. BKF1 (PWMSTS[1]) flag is set by the falling edge at brake 1 edge detector; but cleared by S/W. 2. If LVDBK_EN (PWMCON[27])=1 and Brake 1 signal detected as low level, PWM0~5 pin outputs follow PWMB (PWMB[5:0]) bits. Otherwise, PWM_CH0~5 output will continue follow PWM generators' output. In both situations, PNP (PNP[5:0]) bits are also able to control port polarity. 3. PWMRUN (PWMCON[7]) bit remain asserted to keep PWM generators running. 4. PWM_CH0~5 resume if EPWMx_BRAKE1 (x=0~1) pin state returns to high state. PWM_CH0~5 will resume on start of next PWM cycle/period.

Table 6-10Brake Source, Condition and Action

Note: The brake enable bits, BK0_EN and BK1_EN, must be set in order for the above to be effective.

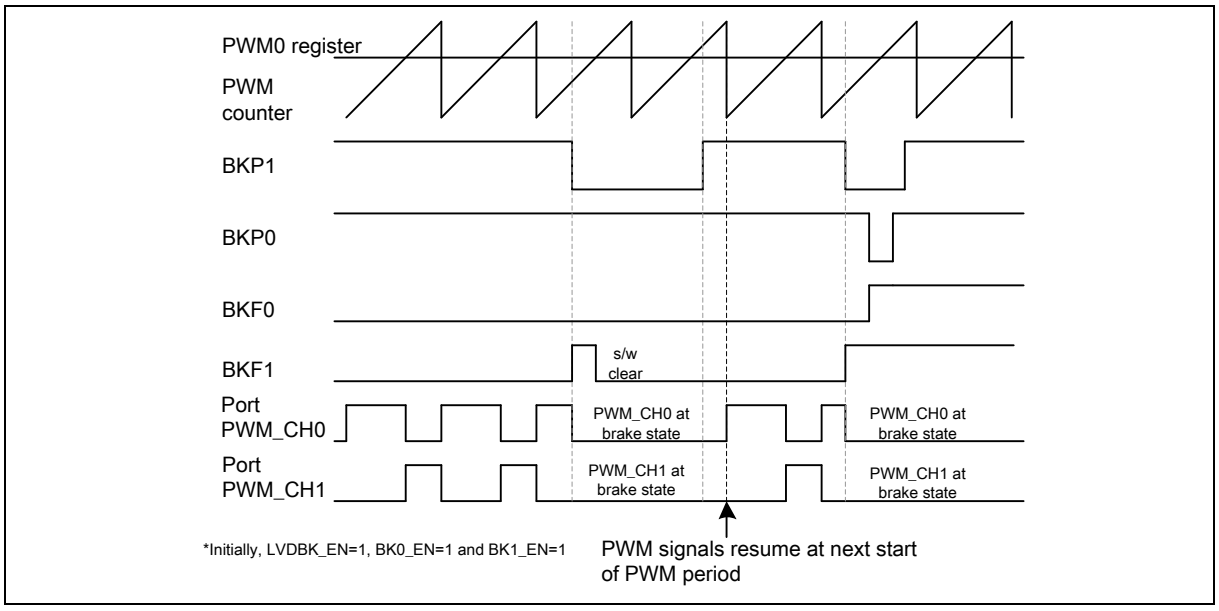


Figure 6–50 PWM Brake Condition (Edge-aligned Mode)

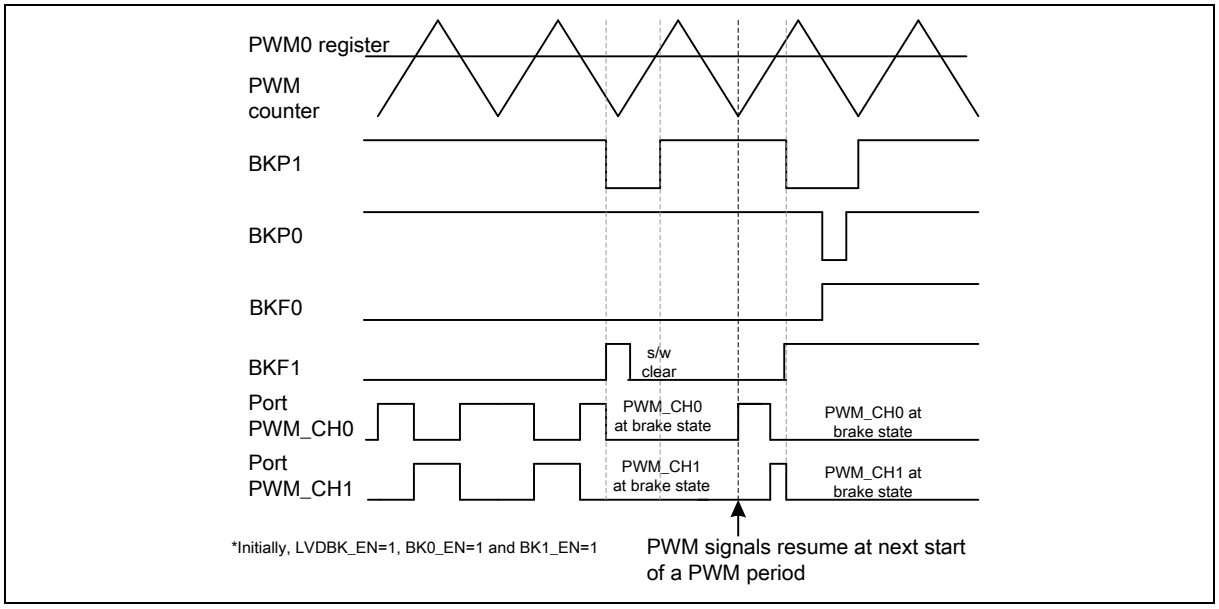


Figure 6–51 PWM Brake Condition (Center-aligned Mode)

Since both brake conditions being asserted will automatically cause BKFn flag to be set, the user program can poll these brake flag bits or enable PWM's brake interrupt to determine which condition will cause a brake to occur.

6.8.5.3 PWM Port Output Driving Control

There are two enhanced PWM units and each unit has six output pins in this device. The PWM port outputs are P0.0~P0.5 and P1.0~P1.5 for unit 0 and unit 1, respectively.

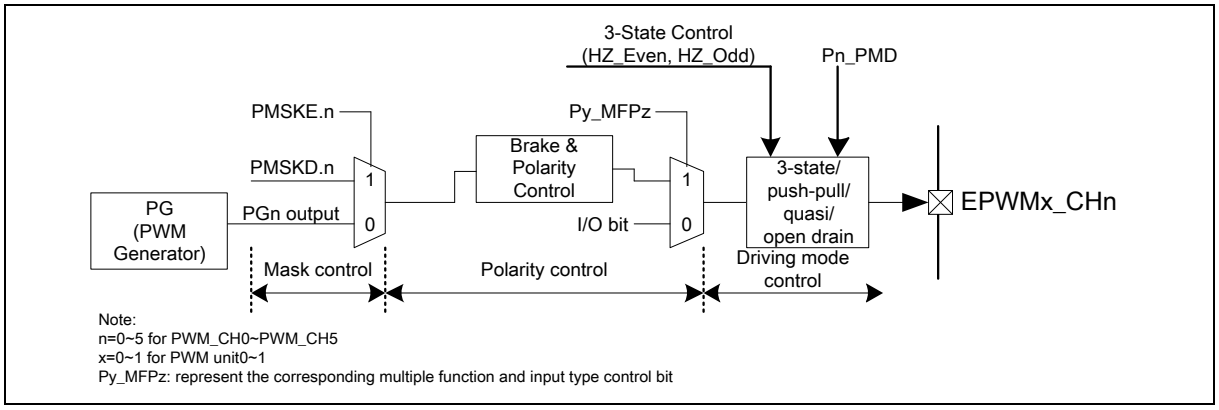


Figure 6–52 PWM Output Driving Control

The driving type of PWM output ports can be initialized as Tri-state type or other types depending on the Pn_PMD register setting after any reset. As shown in the figure above, the PWM output structures are controllable through config-bit, register bitsHZ_Even0/1 (PWMPOEN[0, 2]), Hz_Odd0/1 (PWMPOEN[1, 3] and Pn_PMD mode registers.

HZ_Even/HZ_Odd	Even/Odd PWM Outputs Drive Mode

(In PWMPOEN Register)	
0	Depending on Pn_PMD
1	Driving mode = Hi-Z.

Table 6-11Even/Odd Outputs Drive Mode

Note: Registerbits for HZ_Even and HZ_Odd are latched from config0 during all reset.

6.8.5.4 PWM Modes

This powerful PWM unit supports Independent mode which may be applied to DC and BLDC motor system, Complementary mode with dead-time insertion which may be used in the application of AC induction motor and synchronous motor, Synchronous mode that makes both pins of each pair are in phase. Besides, the Group mode, forces the PWM0, PWM2 and PWM4 synchronous with PWM0 generator, may simplify updating duty control in DC and BLDC motor applications.

6.8.5.4.1 Independent Mode

Independent mode is enabled when PMOD (PWMCON[1:0]) = 00b.

By default, the PWM is operating in Independent mode with three PWM even channels outputs: PWM0, PWM2and PWM4. Each channel is running off its own duty-cycle generator module. The states of PWM1, PWM3 and PWM5 are reset to 0 by default if PWM Mask output function is not enabled (PMSKE (PMSKE[5:0]) =0x00).

6.8.5.4.2 Complementary Mode

Complementary mode is enabled when PMOD (PWMCON[1:0]) = 01b.

In this module there are three duty-cycle generators utilized for complementary mode, with total of three PWM output pair pins in this module. The total six PWM outputs are grouped into output pairs of even and odd numbered outputs. In complimentary modes, the internal odd PWM signal Pgxmust always be the complement of the corresponding even PWM signal. For example, PG1 will be the complement of PG0. PG3 will be the complement of PG2 and PG5 will be the complement of PG4. The time base for the PWM module is provided by its own 16-bit timer, which also incorporates selectable pre-scalar options.

6.8.5.5 Dead-Time Insertion

The dead time generator inserts an “off” period called “dead time” between the turnings off of one pin to the turning on of the complementary pin of the paired pins. This is to prevent damage to the power switching devices that will be connected to the PWM output pins. The complementary output pair mode hasan 11-bit down counter used to produce the dead time insertion. The complementary outputs are delayed until the timer counts down to 0.

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

Dead-time = EPWMx_CLK * (DTCNT (PDTC[10:0])+1).

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

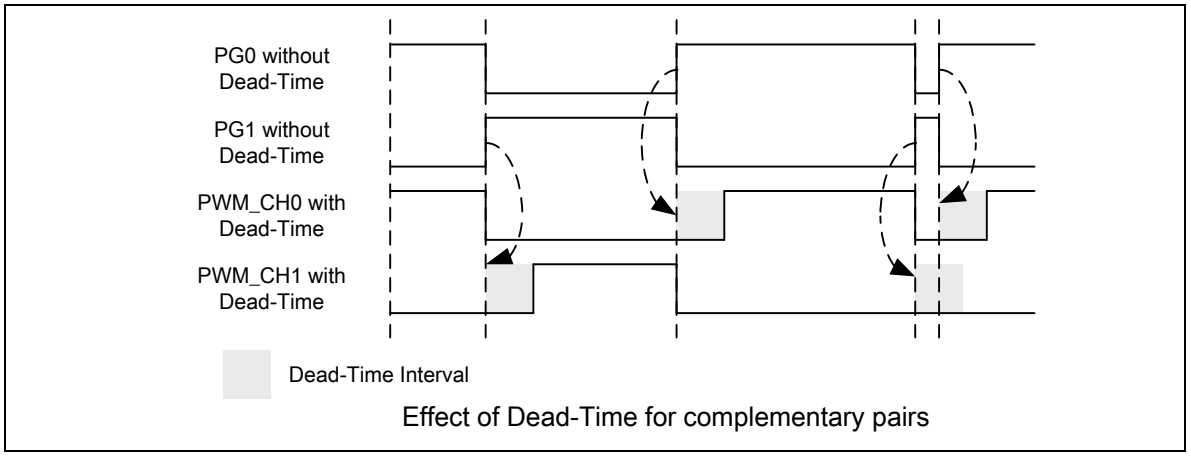


Figure 6–53 Dead-Time Insertion

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

6.8.5.6 Synchronous Mode

Synchronous mode is enabled when PMOD (PWMCON[1:0]) = 10b.

In the synchronization mode the PWM pair signals from PWM Generator are in-phase.

PG1=PG0, PG3=PG2 and PG5=PG4.

6.8.5.7 Group Mode

Group mode is enabled when GRP (PWMCON[13]) = 1.

This device supports Group Mode control. This control allows all even PWM channels output to be duty controllable by PWM0 duty register.

If GRP = 1, both (PG2, PG3) and (PG4, PG5) pairs will follow (PG0, PG1), which imply;

PG4 = PG2 = PG0;

PG5 = PG3 = PG1 = invert (PG0) if Complementary mode is enabled (PMOD (PWMCON[1:0])=01b)

6.8.5.8 Polarity Control

Each PWM port of from PWM_CH0 to PWM_CH5 has independent polarity control to configure the polarity of active state of PWM output. By default, the PWM output is active high. This implies the PWM OFF state is low and ON state is high. This is controllable through the PWM Negative Polarity Control Register, PNPC, on each individual PWM channel.

The following diagram show the initial state before PWM starts with different polarity settings.

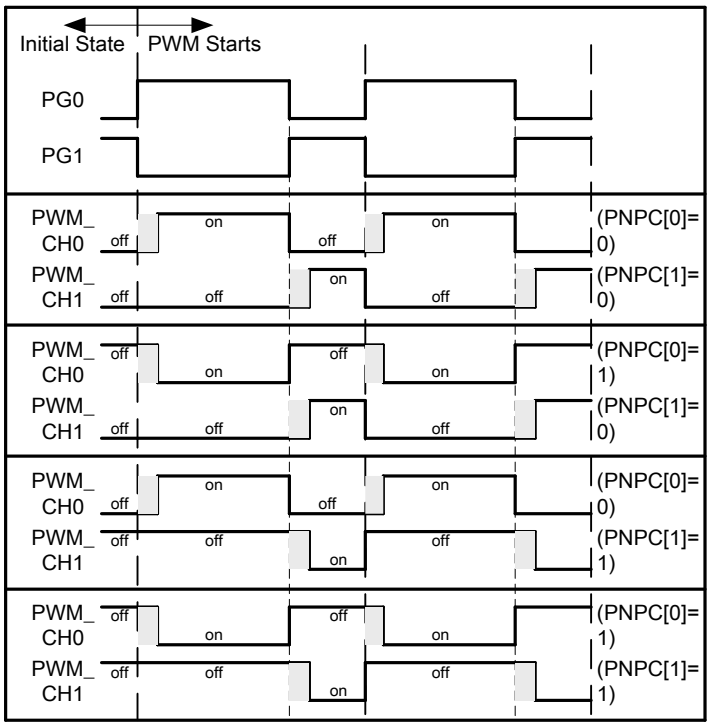


Figure 6–54 Initial State and Polarity Control with Rising Edge Dead Time Insertion

6.8.5.9 Asymmetric PWM Output

This device supports asymmetric PWM output. When asymmetric function is enabled, each PWM channel has an alternate duty register (ASPWMx0/2/4) which is reloaded into the PWM compare register after first time compare matched. The following legends example the PWM0 output with its architecture and output waveform. The active duty width in asymmetric PWM output type, for example PWM0 of unit0, is (PWM00 + ASPWM00 +1).

The following are the constraints in using Asymmetric PWM function:

1. PWM duty register must less than (ASPWM + 1) if ASPWMCON.ASPLDM[1:0] = 01b.
2. PWM duty register must greater than (ASPWM + 1) if ASPWMCON.ASPLDM[1:0] = 10b.
3. That ASPWM equals PWMP or ASPWM equals 00H is not allowed
4. That PWM duty equals PWMP or PWM duty equals 00H is not allowed.

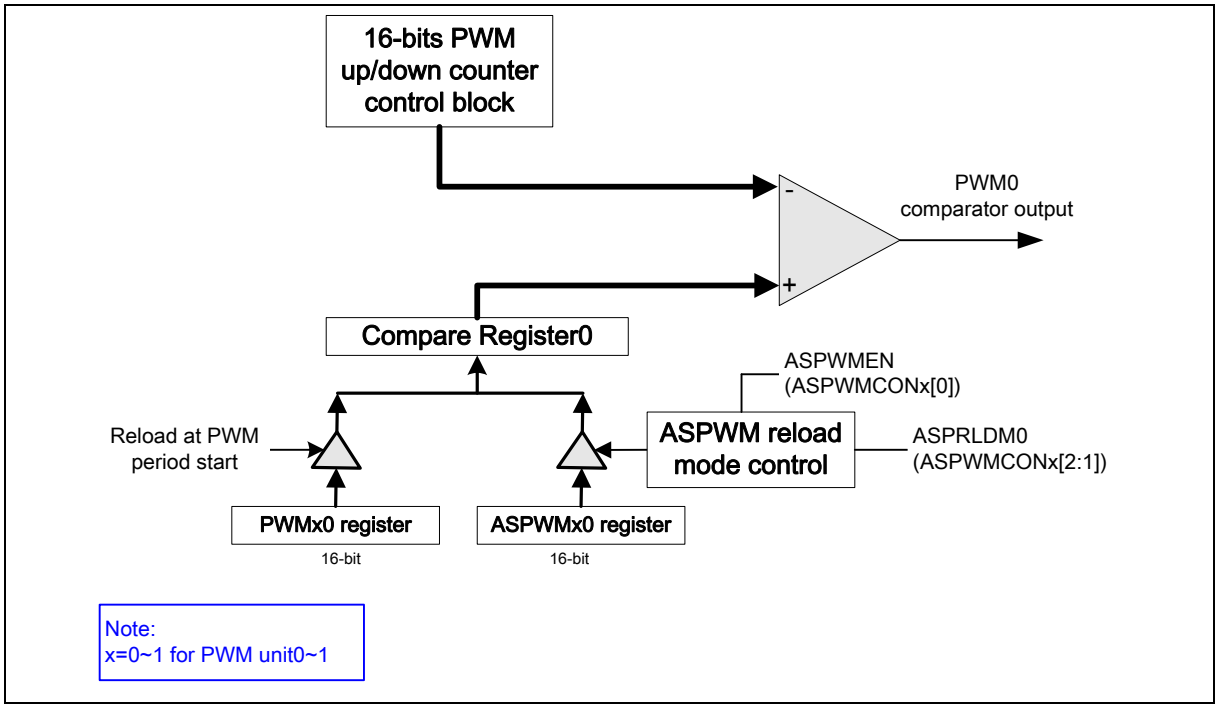


Figure 6–55 Asymmetric PWM Architecture

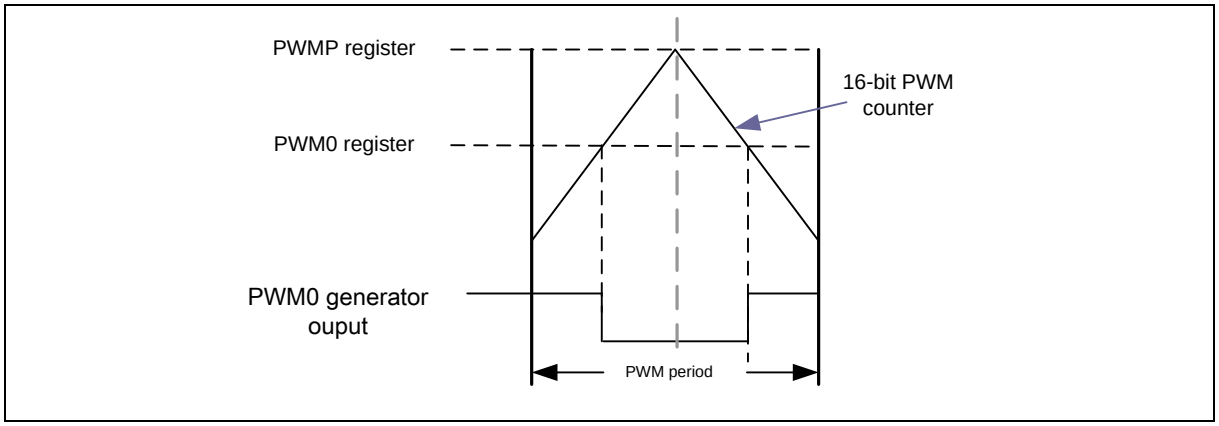


Figure 6–56 Symmetric PWM Output in Centre Aligned Mode

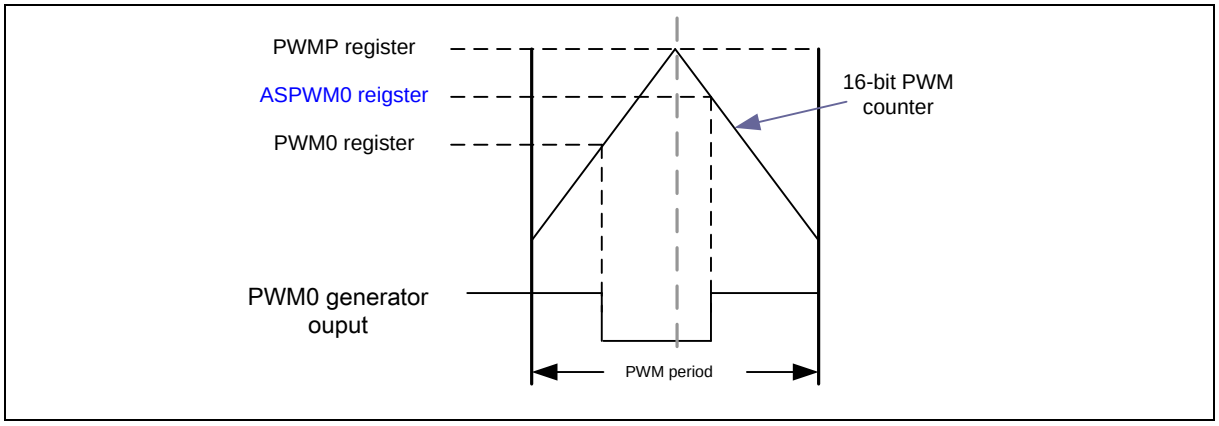


Figure 6–57 Asymmetric PWM Output for ASPRLDM=00b

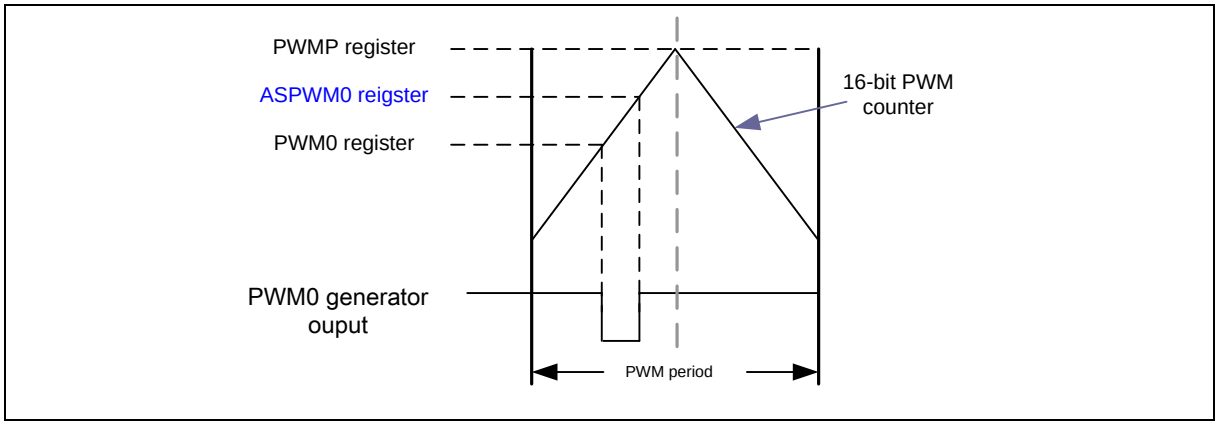


Figure 6–58 Asymmetric PWM Output for ASPRLDM=01b

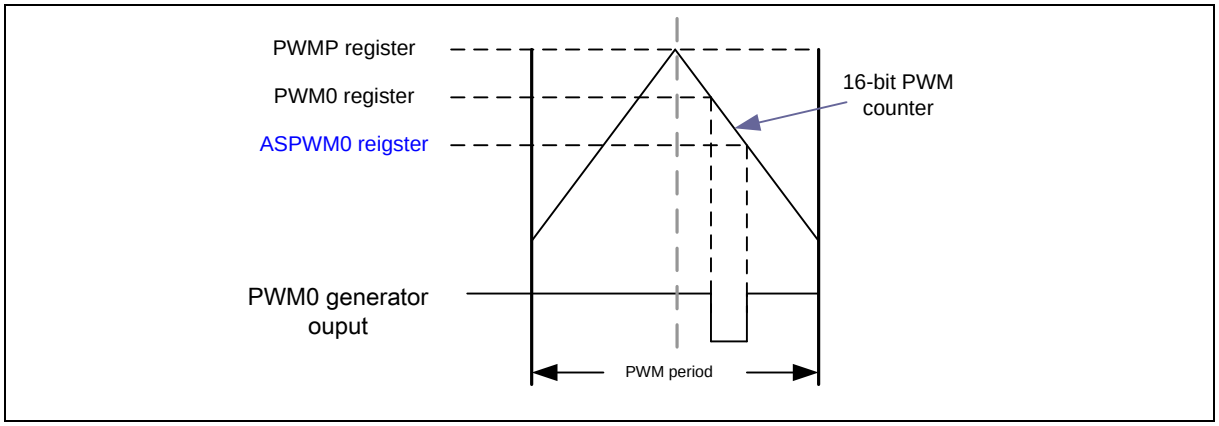


Figure 6–59 Asymmetric PWM Output for ASPRLDM=10b

6.8.5.10 PWM Mask Output Function

Each of the PWM output channels can be manually overridden by using the appropriate bits in the PWM Mask Enable Control Register (PMSKE) and PWM Masked Data Register (PMSKD) to drive the PWM I/O pins to specified logic states independent of the duty cycle comparison units. The PWM mask bits are useful when controlling various types of Electrically Commutated Motor (ECM) like a BLDC motor. The PMSKE register contains six bits, PMSKE[5:0] determine which PWM I/O pins will be overridden. On reset PMSKE is 00H. The PMSKD register contains six bits, PMSKD[5:0] determine the state of the PWM I/O pins when a particular output is masked via the PMSKD bits. On reset PMSKD is 00H. The PMSKE[5:0] bits are active-high. When the PMSKE[5:0] bits are set, the corresponding PMSKD[5:0] bit will have effect on the PWM channel. When one of the PMSKE bits is sets, the output on the corresponding PWM I/O pin will be determined by the state of the PMSKD bit and polarity control bit.

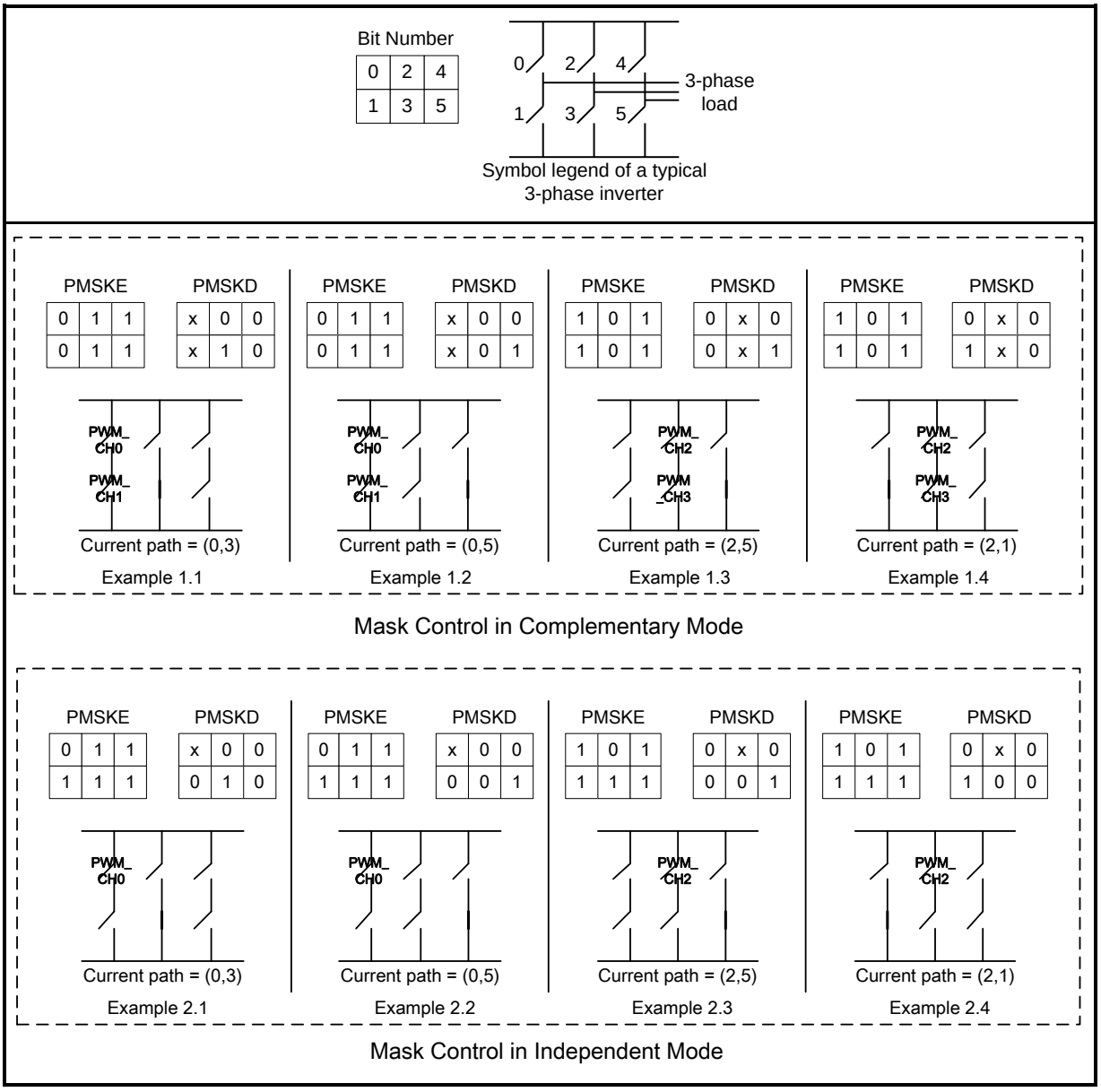


Figure 6–60 Illustration of Mask Control

Figure 6–60 shows example of how PWM mask control can be used for the override feature.

In example 1.1; PMSKE[5:0] = 11 1100b, PMSKD[5:0] = 0010xxb (Complementary Mode)

PWM channels 2-5 are masked from PWM frequency/duty generators.

PWM channels 2-5 outputs are determined by state of PMSKD bits.

PWM channels 0 and 1 follow PWM generator.

Switch 0 (On/Off) : Control by PWM_CH0 (PWM0 frequency/duty generator).

Switch 1 (On/Off) : Control by PWM_CH1 (inverted of PWM0, complementary mode).

Switch 2 (Off) : PMSKD[2] = 0.

Switch 3 (On) : PMSKD[3] = 1.

Switch 4 (Off) : PMSKD[4] = 0.

Switch 5 (Off) : PMSKD[5] = 0.

In example 1.3; PMSKE[5:0] = 11 0011b, PMSKD[5:0] = 10xx00b (Complementary Mode)

PWM channels 0, 1, 4 and 5 are masked from PWM frequency/duty generators.

PWM channels 0, 1, 4 and 5 outputs are determined by state of PMSKD bits.

PWM channels 2 and 3 follow PWM generator.

Switch 0 (Off) : PMSKD[0] = 0.

Switch 1 (Off) : PMSKD[1] = 0.

Switch 2 (On/Off) : Control by PWM_CH2 (PWM2 frequency/duty generator).

Switch 3 (On/Off) : Control by PWM_CH3 (inverted of PWM2, complementary mode).

Switch 4 (Off) : PMSKD[4] = 0.

Switch 5 (On) : PMSKD[5] = 1.

In example 2.1; PMSKE[5:0] = 11 1110b, PMSKD[5:0] = 00100xb (Independent Mode)

PWM channels 1-5 are masked from PWM frequency/duty generators.

PWM channels 1-5 outputs are determined by state of PMSKD bits.

PWM channel 0 follow PWM generator.

Switch 0 (On/Off) : Control by PWM_CH0 (PWM0 frequency/duty generator).

Switch 1 (Off) : PMSKD[1] = 0.

Switch 2 (Off) : PMSKD[2] = 0.

Switch 3 (On) : PMSKD[3] = 1.

Switch 4 (Off) : PMSKD[4] = 0.

Switch 5 (Off) : PMSKD[5] = 0.

In example 2.3; PMSKE[5:0] = 11 1011b, PMSKD[5:0] = 100x00b (Independent Mode)

PWM channels 0,1,3,4 and 5 are masked from PWM frequency/duty generators.

PWM channels 0,1,3,4 and 5 outputs are determined by state of PMSKD bits.

PWM channel 2 follow PWM generator.

Switch 0 (Off) : PMSKD[0] = 0.

Switch 1 (Off) : PMSKD[1] = 0.

Switch 2 (On/Off) : Control by PWM_CH2 (PWM2 frequency/duty generator).

Switch 3 (Off) : PMSKD[3] = 0.

Switch 4 (Off) : PMSKD[4] = 0.

Switch 5 (On) : PMSKD[5] = 1.

6.8.5.11 Enhanced PWM trigger EADC

EPWM can generate trigger signals to EADC, when PWM channels output rising or falling asFigure 6–61.

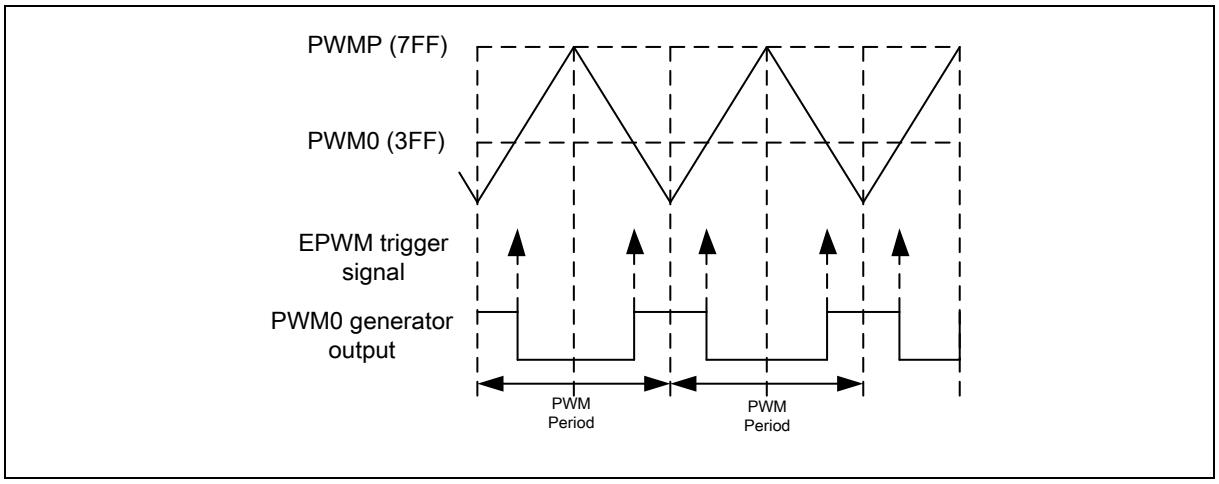


Figure 6–61EPWM Trigger EADC Timing Diagram

6.8.5.12 Interrupt Architecture of Enhanced PWM

There are six interrupt sources for each PWM unit, including PWM period flag (PWMF), Brake0 flag (BKF0), Brake1 flag (BKF1), PWM0 edge flag, PWM2 edge flag and PWM4 edge flag. The bit BRKI_EN (PWMCON[5]) controls the brake interrupt enable; the bit PWMI_EN (PWMCON[4]) controls the PWM periodic interrupt enable; the bit PWM0EI_EN (PWMEIC[0]) controls the PWM0 edge interrupt enable; the bit PWM2EI_EN (PWMEIC[1]) controls the PWM2 edge interrupt enable; the bit PWM4EI_EN (PWMEIC[2]) controls the PWM4 edge interrupt enable.

Note: All the interrupt flags are set by hardware and must be cleared by writing 1 to flags through software.

Figure 6–62demonstrates the architecture of enhanced PWM interrupts.

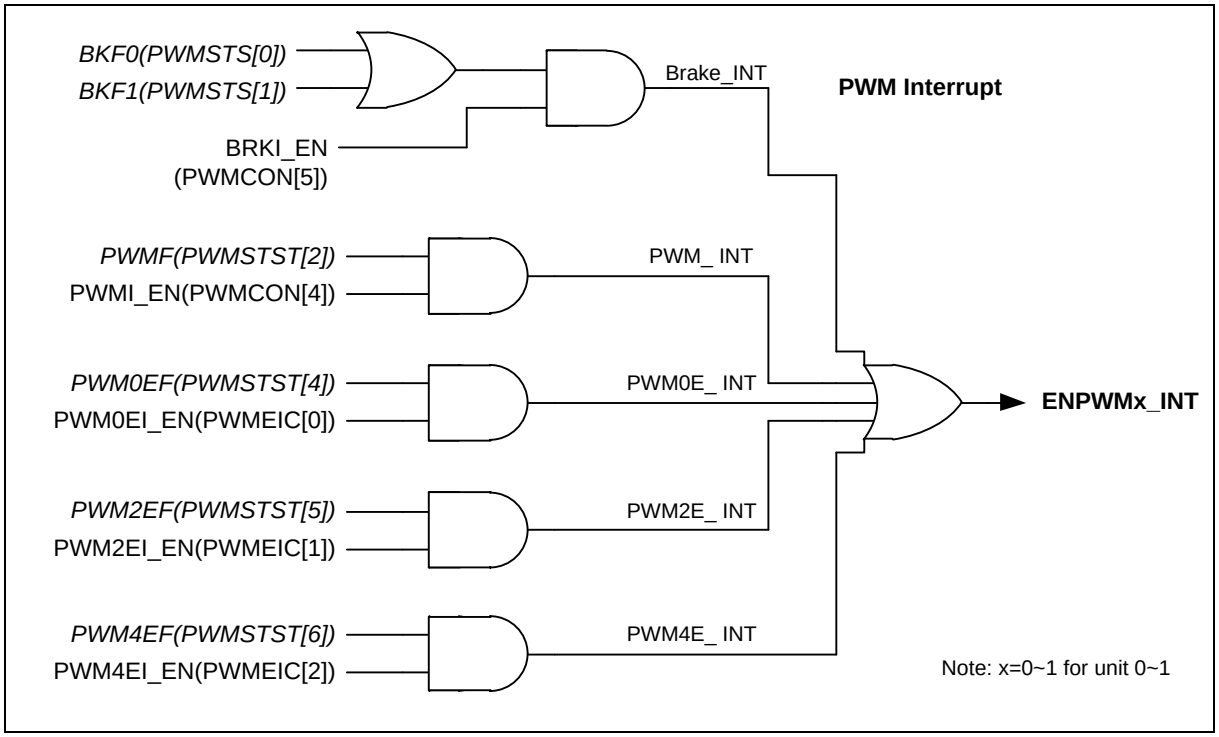


Figure 6–62 Architecture of Enhanced PWM Interrupts

6.8.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
EPWM Base Address: **EPWMx_BA = 0x4019_0000 + (0x4000 * x)** **x = 0, 1**				
PWMCON	EPWMx_BA+0x00	R/W	EPWM Control Register	0x0000_0000
PWMSTS	EPWMx_BA+0x04	R/W	EPWM Status Register	0x0000_0000
PWMP	EPWMx_BA+0x08	R/W	EPWM Period Register	0x0000_0000
PWM0	EPWMx_BA+0x0C	R/W	EPWM PWM0 Duty Register	0x0000_0000
PWM2	EPWMx_BA+0x10	R/W	EPWM PWM2 Duty Register	0x0000_0000
PWM4	EPWMx_BA+0x14	R/W	EPWM PWM4 Duty Register	0x0000_0000
PMSKE	EPWMx_BA+0x18	R/W	EPWM Mask Mode Enable Register	0x0000_0000
PMSKD	EPWMx_BA+0x1C	R/W	EPWM Mask Mode Data Register	0x0000_0000
ASPWMx0	EPWMx_BA+0x20	R/W	Asymmetric PWM0 Duty Register	0x0000_0000
ASPWMx2	EPWMx _BA+0x24	R/W	Asymmetric PWM2 Duty Register	0x0000_0000
ASPWMx4	EPWMx _BA+0x28	R/W	Asymmetric PWM4 Duty Register	0x0000_0000
PDTC	EPWMx_BA+0x2C	R/W	EPWM Dead-time Control Register	0x0000_0000
PWMB	EPWMx_BA+0x30	R/W	EPWM Brake Output Register	0x0000_0000
PNPC	EPWMx_BA+0x34	R/W	EPWM Negative Polarity Control Register	0x0000_0000
ASPWMCONx	EPWMx _BA+0x38	R/W	Asymmetric PWM Control Register	0x0000_0000
PWMFCNT	EPWMx_BA+0x3C	R/W	EPWMF Compared Counter Register	0x0000_0000
PWMEIC	EPWMx_BA+0x40	R/W	EPWM Edge Interrupt Control Register	0x0000_0000

6.8.7 Register Description

EPWM Control Register (PWMCON)

Register	Offset	R/W	Description	Reset Value
PWMCON	EPWMx_BA+0x00	R/W	EPWM Control Register	0x0000_0000

31	30	29	28	27	26	25	24
CLDMD	AUTOLD	BK1NF_DIS	BK0NF_DIS	LVDBK_EN	CPO2BK_EN	CPO1BK_EN	CPO0BK_EN
23	22	21	20	19	18	17	16
BK1FILT		BK0FILT		BK1SEL		BKEN1	BKEN0
15	14	13	12	11	10	9	8
INVBKP1	INVBKP0	GRP	PWMTYPE	CLRPWM	Reserved	PWMINV	INT_TYPE
7	6	5	4	3	2	1	0
PWMRUN	LOAD	BRKI_EN	PWMI_EN	PWMDIV		PWMMOD	

Bits	Description	
[31]	CLDMD	**Center Reload Mode Enable Bit** 0 = EPWM reload duty register at the period point of PWM counter. 1 = EPWM reload duty register at the center point of PWM counter. This bit only works when EPWM operatinginCenter-aligned mode.
[30]	AUTOLD	**Auto Load Enable Bit** 0 = PWM duty registers PWM0/PWM2/PWM4 are updated by software. 1 = PWM duty registers PWM0/PWM2/PWM4 are auto-load from motor drive unit (MDU) when MDU update a set of new duty values for PWM unit.
[29]	BK1NF_DIS	**PWM Brake 1 Noise Filter Disable Bit** 0 = Noise filter of PWM Brake 1 Enabled. 1 = Noise filter of PWM Brake 1 Disabled.
[28]	BK0NF_DIS	**PWM Brake 0 Noise Filter Disable Bit** 0 = Noise filter of PWM Brake 0 Enabled. 1 = Noise filter of PWM Brake 0 Disabled.
[27]	LVDBK_EN	**Low-level Detection Trigger PWM Brake Function 1 Enable Bit** 0 = Brake Function 1 triggered by Low-level detection Disabled. 1 = Brake Function 1 triggered by Low-level detection Enabled.
[26]	CPO2BK_EN	**ACMP2Digital Output As Brake 0 Source Enable Bit** 0 = CO2 (ACMPSR[10]) as one brake source in Brake 0 Disabled. 1 = CO2 (ACMPSR[10]) as one brake source in Brake 0 Enabled.
[25]	CPO1BK_EN	**ACMP1Digital Output As Brake 0 Source Enable Bit** 0 = CO1(ACMPSR[9]) as one brake source in Brake 0 Disabled. 1 = CO1 (ACMPSR[9]) as one brake source in Brake 0 Enabled.

Bits	Description	
[24]	**CPO0BK_EN**	**ACMP0 Digital Output As Brake0 Source Enable Bit** 0 = CO0(ACMPSR[8]) as one brake source in Brake 0 Disabled. 1 = CO0 (ACMPSR[8]) as one brake source in Brake 0 Enabled.
[23:22]	**BK1FILT**	**Brake 1(BKPx1 Pin) Edge Detector Filter Clock Selection** 00 = filter clock is HCLK. 01 = filter clock is HCLK/2. 10 = filter clock is HCLK/4. 11 = filter clock is HCLK/16.
[21:20]	**BK0FILT**	**Brake 0(BKPx0 Pin) Edge Detector Filter Clock Selection** 00 = filter clock is HCLK. 01 = filter clock is HCLK/2. 10 = filter clock is HCLK/4. 11 = filter clock is HCLK/16.
[19:18]	**BK1SEL**	**Brake Function 1 Source Selection** 00 = brake signal is from external pin EPWMx_BRAKE1 (x=0~1 for unit0~1). 01 = brake signal is from analog comparator 0 output CO0 (ACMPSR[8]). 10 = brake signal is from analog comparator 1 output CO1 (ACMPSR[9]). 11 = brake signal is from analog comparator 2 output CO2 (ACMPSR[10]).
[17]	**BKEN1**	**BRAKE1 Pin Trigger Brake Function 1Enable Bit** 0 = PWMx brake function 1 Disabled. 1 = PWMx brake function 1 Enabled. **Note:** x=0~1 for PWM unit0~1.
[16]	**BKEN0**	**BRAKE0 Pin Trigger Brake Function0 Enable Bit** 0 = PWMx brake function 0 Disabled. 1 = PWMx brake function 0 Enabled. **Note:** x = 0~1 for PWM unit0~1.
[15]	**INVBKP1**	**Inverse Brake 1 Pin State** 0 = The state of pin EPWMx_BRAKE1 is passed to the negative edge detector. 1 = The inversed state of pin EPWMx_BRAKE1 is passed to the negative edge detector.
[14]	**INVBKP0**	**Inverse Brake 0 Pin State** 0 = The state of pin EPWMx_BRAKE0 is passed to the negative edge detector. 1 = The inversed state of pin EPWMx_BRAKE0 is passed to the negative edge detector.
[13]	**GRP**	**Group Bit** 0 = The signals timing of PWM_CH0, PWM_CH2 and PWM_CH4 are independent. 1 = Unify the signals timing of PWM_CH0, PWM_CH2 and PWM_CH4 in the same phase which is controlled by PWM_CH0.
[12]	**PWMTYPE**	**PWM Aligned Type Selection Bit** 0 = Edge-aligned type. 1 = Center-aligned type.
[11]	**CLRPWM**	**Clear PWM Counter Control Bit** 0 = Ignored. 1 = Clear 16-bit PWM counter to 0000H. **Note:** It is automatically cleared by hardware.

Bits	Description	
[10]	**Reserved**	Reserved.
[9]	**PWMINV**	**Inverse PWM Comparator Output** When PWMINV is set to high the PWM comparator output signals will be inversed, therefore the PWM Duty (in percentage) is changed to (1-Duty) before PWMINV is set to high. 0 = Not inverse PWM comparator output. 1 = Inverse PWM comparator output.
[8]	**INT_TYPE**	**PWM Interrupt Type Selection Bit** 0 = PWMF will be set if PWM counter underflow. 1 = PWMF will be set if PWM counter matches PWMP register. **Note:** This bit is effective when PWM is in Center-aligned mode only.
[7]	**PWMRUN**	**Start PWMRUN Control Bit** 0 = The PWM stops running. 1 = The PWM counter starts running.
[6]	**LOAD**	**Reload PWM Period Registers(PWMP)and PWM Duty Registers (PWM0~4)Control Bit** 0 = No action if written with 0. The value of PWM period register (PWMP) and PWM duty registers (PWM0~PWM4) are not loaded to PWM counter and Comparator registers. 1 = Hardware will update the value of PWM period register (PWMP) and PWM duty registers (PWM0~PWM4) to PWM Counter and Comparator register at the time of PWM Counter matches PWMP in Edge- and Center-aligned modes or at the time of PWM Counter down counts with underflow in Center-aligned mode. **Note:** This bit is written by software, cleared by hardware, and always read as 0.
[5]	**BRKI_EN**	**Enable Brake0 and Brak1 Interrupt** 0 = Flags BKF0 (PWMSTS[0]) and BKF1 (PWMSTS[1]) Disabled to trigger PWM interrupt. 1 = Flags BKF0 (PWMSTS[0]) and BKF1 (PWMSTS[1]) Enabled to trigger PWM interrupt.
[4]	**PWMI_EN**	**Enable PWM Interrupt** 0 = Flag PWMF (PWMSTS[2]) Disabled to trigger PWM interrupt. 1 = Flag PWMF (PWMSTS[2]) Enabledto trigger PWM interrupt.
[3:2]	**PWMDIV**	**PWM Clock Pre-divider Selection** 00 = PWM clock is EPWMx_CLK. 01 = PWM clock is EPWMx_CLK/2. 10 = PWM clock is EPWMx_CLK/4. 11 = PWM clock isEPWMx_CLK/16.
[1:0]	**PWMMOD**	**PWM Mode Selection** 00 = PWM mode is independent mode. 01 = PWM mode is pair/complementary mode. 10 = PWM mode is synchronized mode. 11 = Reserved.

EPWM Status Register (PWMSTS)

Register	Offset	R/W	Description	Reset Value
PWMSTS	EPWMx_BA+0x04	R/W	EPWM Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved						BK1STS	BK0STS
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							BKLK0
7	6	5	4	3	2	1	0
Reserved	PWM4EF	PWM2EF	PWM0EF	Reserved	PWMF	BKF1	BKF0

Bits	Description	
[31:26]	**Reserved**	Reserved.
[25]	**BK1STS**	**Brake 1 Status (Read Only)** 0 = PWM had been out of Brake 1 state. 1 = PWM is in Brake 1 state.
[24]	**BK0STS**	**Brake 0 Status (Read Only)** 0 = PWM had been out of Brake 0 state. 1 = PWM is in Brake 0 state.
[23:9]	**Reserved**	Reserved.
[8]	**BKLK0**	**PWM Brake0 Locked** 0 = Brake 0 state is released. 1 = When PWM Brake detects a falling signal at EPWMx_BRAKE0, x=0, 1.This flag will be set to high to indicate the Brake0 state is locked. **Note:** This bit must be cleared by writing 1 to itself through software.
[7]	**Reserved**	Reserved.
[6]	**PWM4EF**	**PWM Channel 4 Edge Flag** 0 = PWM_CH4 not toggled. 1 = Hardware will set this flag to high at the time of PWM_CH4 rising or falling. If EINT4_TYPE (PWMEIC[10]) = 0, this bit is set when PWM_CH4 falling is detected. If EINT4_TYPE (PWMEIC[10]) = 1, this bit is set when PWM_CH4 rising is detected. **Note:** This bit must be cleared by writing 1 to itself through software.
[5]	**PWM2EF**	**PWM Channel 2 Edge Flag** 0 = PWM_CH2 not toggled. 1 = Hardware will set this flag to high at the time of PWM_CH2 rising or falling. If EINT2_TYPE (PWMEIC[9]) = 0, this bit is set when PWM_CH2 falling is detected. If EINT2_TYPE (PWMEIC[9]) = 1, this bit is set when PWM_CH2 rising is detected. **Note:** This bit must be cleared by writing 1 to itself through software.

[4]	**PWM0EF**	**PWM Channel 0 Edge Flag** 0 = PWM_CH0 not toggled. 1 = Hardware will set this flag to high at the time of PWM_CH0 rising or falling. If EINT0_TYPE (PWMEIC[8]) = 0, this bit is set when PWM_CH0 falling is detected. If EINT0_TYPE (PWMEIC[8]) = 1, this bit is set when PWM_CH0 rising is detected. **Note:** This bit must be cleared by writing 1 to itself through software.
[3]	**Reserved**	Reserved.
[2]	**PWMF**	**PWM Period Flag** 0 = The PWM Counter has not up counted to the value of PWMP or down counted with underflow. 1 = Hardware will set this flag to high at the time of PWM Counter matches PWMP in Edge- and Center-aligned modes or at the time of PWM Counter down counts with underflow in Center-aligned mode. **Note:** This bit must be cleared by writing 1 to itself through software.
[1]	**BKF1**	**PWM Brake1 Flag** 0 = PWM Brake 1 is able to poll falling signal at EPWMx_BRAKE1, x=0, 1 and has not recognized any one. 1 = When PWM Brake 1 detects a falling signal at pin EPWMx_BRAKE1, x=0, 1, this flag will be set to high. **Note:** This bit must be cleared by writing 1 to itself through software.
[0]	**BKF0**	**PWM Brake0 Flag** 0 = PWM Brake 0 is able to poll falling signal at EPWMx_BRAKE0, x=0, 1 and has not recognized any one. 1 = When PWM Brake 0 detects a falling signal at EPWMx_BRAKE0, x=0, 1, this flag will be set to high. **Note:** This bit must be cleared by writing 1 to itself through software.

EPWM Period Register (PWMP)

Register	Offset	R/W	Description	Reset Value
PWMP	EPWMx_BA+0x08	R/W	EPWM Period 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**
PWMP							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PWMP							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PWMP**	**PWM Period Register** **Edge-aligned**: Period = (PWMP +1) * EPWMx_CLK period/pre-scalar. Duty= (Duty + 1) * EPWMx_CLK period/pre-scalar. **Centre-aligned**: Period = (PWMP* 2) * EPWMx_CLK period/pre-scalar. Duty= (Duty*2 + 1) * EPWMx_CLK period/pre-scalar.

EPWM Duty Register (PWM0/2/4)

Register	Offset	R/W	Description	Reset Value
PWM0	EPWMx_BA+0x0C	R/W	EPWM PWM0 Duty Register	0x0000_0000
PWM2	EPWMx_BA+0x10	R/W	EPWM PWM2 Duty Register	0x0000_0000
PWM4	EPWMx_BA+0x14	R/W	EPWM PWM4 Duty 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**
PWM_Duty							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
PWM_Duty							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PWM_Duty**	**PWM Duty Register** **Edge-aligned**: Period = (PWMP +1) * EPWMx_CLK period/pre-scalar. Duty= (Duty + 1) * EPWMx_CLK period/pre-scalar. **Centre-aligned**: Period = (PWMP* 2) * EPWMx_CLK period/pre-scalar. Duty = (Duty*2 + 1) * EPWMx_CLK period/pre-scalar.

EPWM Mask Mode Enable Control Register (PMSKE)

Register	Offset	R/W	Description	Reset Value
PMSKE	EPWMx_BA+0x18	R/W	EPWM Mask Mode 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		PMSKE					

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**PMSKE**	**PWM Mask Enable Bits** The PWM generator signal will be masked when this bit is enabled. The corresponding PWM channel n will be output with PMSKD[n] data. 0 = PWM generator signal is output to next stage. 1 = PWM generator signal is masked and PMSKD[n] is output to next stage, n = 0~5.

EPWM Mask Mode Data Register (PMSKD)

Register	Offset	R/W	Description	Reset Value
PMSKD	EPWMx_BA+0x1C	R/W	EPWM Mask Mode Data Register	0x0000_0000

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

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**PMSKD**	**PWM Mask Data Bit** This data bit control the state of PWM_CHn output pin, if corresponding PMSKE[n] = 1, n = 0~5. 0 = Output logic low to PWM_CHn. 1 = Output logic high to PWM_CHn.

Asymmetric PWM Control Register (ASPWMCON0-1)

Register	Offset	R/W	Description	Reset Value
ASPWMCON0	EPWM0_BA+0x38	R/W	(Unit0) Asymmetric PWM Control Register	0x0000_0000
ASPWMCON1	EPWM1_BA+0x38	R/W	(Unit1) Asymmetric PWM Control Register	0x0000_0000

31	30	29	28	27	26	25	24
-						ASPRLDM4	
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
-						ASPRLDM2	
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
-						ASPRLDM0	
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
-							ASPWMEN

Bits	Descriptions	
[31:26]	-	**Reserved.**

<table>
<tr><th>Bits</th><th>Descriptions</th><th></th></tr>
<tr><td>[25:24]</td><td>ASPLDM4[1:0]</td><td>
Asymmetric PWMx4 Reload Mode Setting
<table>
<tr><th>ASPLDM0[1:0]</th><th>PWM Compare Operation</th></tr>
<tr><td>00</td><td>1. PWM compare register 4 is reload PWM_Duty (PWMx4[15:0]) at PWM cycle start.
2. PWM module compares the PWM counter (PWMCNT) with PWM compare register 4 in up-counting cycle
3. PWM compare output toggles at compare match, in the meanwhile, PWM compare register 4 is reloaded with ASPWM_Duty (ASPWMx4[15:00]).
4. PWM module compares the PWM counter (PWMCNT) with PWM compare register 4 in down-counting cycle.</td></tr>
<tr><td>01</td><td>1. PWM compare register 4 is reload PWM_Duty (PWMx4[15:0]) at PWM cycle start.
2. PWM module compares the PWM counter (PWMCNT) with PWM compare register 4 in up-counting cycle
3. PWM compare output toggles at compare match, in the meanwhile, PWM compare register 0 is reloaded with ASPWM_Duty (ASPWMx4[15:00]).
4. PWM module compares the PWM counter (PWMCNT) with PWM compare register 4 in up-counting cycle.
5. PWMx4 must be less than ASPWMx4.</td></tr>
<tr><td>10</td><td>1. PWM compare register 4 is reload PWM_Duty (PWMx4[15:0]) at PWM cycle start.
2. PWM module compares the PWM counter (PWMCNT) with PWM compare register 4 in down-counting cycle
3. PWM compare output toggles at compare match, in the meanwhile, PWM compare register 4 is reloaded with ASPWM_Duty (ASPWMx4[15:00]).
4. PWM module compares the PWM counter (PWMCNT) with PWM compare register 4 in down-counting cycle.
5. PWMx4 must be greater than ASPWMx4.</td></tr>
<tr><td>11</td><td>Reserved</td></tr>
</table>
Note:
1. x=0~1 for PWM unit 0~1.
2. This bit field is available only when ASPWMEN=1.
3. The setting will be effected at the condition of PWMCON.LOAD=1 and from the start of next PWM cycle.
</td></tr>
<tr><td>[23:18]</td><td>-</td><td>Reserved.</td></tr>
</table>

<table>
<tr><th>Bits</th><th>Descriptions</th><th></th></tr>
<tr><td>[17:16]</td><td>ASPLDM2[1:0]</td><td>
Asymmetric PWMx2 Reload Mode Setting
<table>
<tr><th>ASPLDM0[1:0]</th><th>PWM Compare Operation</th></tr>
<tr><td>00</td><td>1. PWM compare register 2 is reload PWM_Duty (PWMx2[15:0]) at PWM cycle start.
2. PWM module compares the PWM counter (PWMCNT) with PWM compare register 2 in up-counting cycle
3. PWM compare output toggles at compare match, in the meanwhile, PWM compare register 2 is reloaded with ASPWM_Duty (ASPWMx2[15:00]).
4. PWM module compares the PWM counter (PWMCNT) with PWM compare register 2 in down-counting cycle.</td></tr>
<tr><td>01</td><td>1. PWM compare register 2 is reload PWM_Duty (PWMx2[15:0]) at PWM cycle start.
2. PWM module compares the PWM counter (PWMCNT) with PWM compare register 2 in up-counting cycle
3. PWM compare output toggles at compare match, in the meanwhile, PWM compare register2 is reloaded with ASPWM_Duty (ASPWMx2[15:00]).
4. PWM module compares the PWM counter (PWMCNT) with PWM compare register 2 in up-counting cycle.
5. PWMx2 must be less than ASPWMx2</td></tr>
<tr><td>10</td><td>1. PWM compare register 2 is reload PWM_Duty (PWMx2[15:0]) at PWM cycle start.
2. PWM module compares the PWM counter (PWMCNT) with PWM compare register 2 in down-counting cycle
3. PWM compare output toggles at compare match, in the meanwhile, PWM compare register 2 is reloaded with ASPWM_Duty (ASPWMx2[15:00]).
4. PWM module compares the PWM counter (PWMCNT) with PWM compare register 2 in down-counting cycle.
5. PWMx2 must be greater than ASPWMx2.</td></tr>
<tr><td>11</td><td>Reserved</td></tr>
</table>
Note:
1. x=0~1 for PWM unit 0~1.
2. This bit field is available only when ASPWMEN=1.
3. The setting will be effected at the condition of PWMCON.LOAD=1 and from the start of next PWM cycle.
</td></tr>
<tr><td>[15:10]</td><td>-</td><td>Reserved.</td></tr>
</table>

<table>
<tr><th>Bits</th><th>Descriptions</th><th></th></tr>
<tr><td>[9:8]</td><td>ASPLDM0[1:0]</td><td>
Asymmetric PWMx0 Reload Mode Setting
<table>
<tr><th>ASPLDM0[1:0]</th><th>PWM Compare Operation</th></tr>
<tr><td>00</td><td>1. PWM compare register 0 is reload PWM_Duty (PWMx0[15:0]) at PWM cycle start.
2. PWM module compares the PWM counter (PWMCNT) with PWM compare register 0 in up-counting cycle
3. PWM compare output toggles at compare match, in the meanwhile, PWM compare register 0 is reloaded with ASPWM_Duty (ASPWMx0[15:00]).
4. PWM module compares the PWM counter (PWMCNT) with PWM compare register 0 in down-counting cycle.</td></tr>
<tr><td>01</td><td>1. PWM compare register 0 is reload PWM_Duty (PWMx0[15:0]) at PWM cycle start.
2. PWM module compares the PWM counter (PWMCNT) with PWM compare register 0 in up-counting cycle
3. PWM compare output toggles at compare match, in the meanwhile, PWM compare register 0 is reloaded with ASPWM_Duty (ASPWMx0[15:00]).
4. PWM module compares the PWM counter (PWMCNT) with PWM compare register 0 in up-counting cycle.
5. PWMx0 must be less than ASPWMx0</td></tr>
<tr><td>10</td><td>1. PWM compare register 0 is reload PWM_Duty (PWMx0[15:0]) at PWM cycle start.
2. PWM module compares the PWM counter (PWMCNT) with PWM compare register 0 in down-counting cycle
3. PWM compare output toggles at compare match, in the meanwhile, PWM compare register 0 is reloaded with ASPWM_Duty (ASPWMx0[15:00]).
4. PWM module compares the PWM counter (PWMCNT) with PWM compare register 0 in down-counting cycle.
5. PWMx0 must be greater than ASPWMx0.</td></tr>
<tr><td>11</td><td>Reserved</td></tr>
</table>
Note:
1. x=0~1 for PWM unit 0~1.
2. This bit field is available only when ASPWMEN=1.
3. The setting will be effected at the condition of PWMCON.LOAD=1 and from the start of next PWM cycle.
</td></tr>
<tr><td>[7:1]</td><td>-</td><td>Reserved.</td></tr>
<tr><td>[0]</td><td>ASPWMEN</td><td>Asymmetric PWM Enable bit.
1 = Asymmetric PWM function is enabled.
0 = Asymmetric PWM function is disabled.
This control bit is only valid when PWM module is set in centre-aligned mode.</td></tr>
</table>

Asymmetric PWM Register (ASPWM00/02/04, ASPWM10/12/14)

Register	Offset	R/W	Description	Reset Value
ASPWM00	EPWM0_BA+0x20	R/W	(Unit0) Asymmetric PWM0 Setting Register	0x0000_0000
ASPWM02	EPWM0_BA+0x24	R/W	(Unit0) Asymmetric PWM2 Setting Register	0x0000_0000
ASPWM04	EPWM0_BA+0x28	R/W	(Unit0) Asymmetric PWM4 Setting Register	0x0000_0000
ASPWM10	EPWM1_BA+0x20	R/W	(Unit1) Asymmetric PWM0 Setting Register	0x0000_0000
ASPWM12	EPWM1_BA+0x24	R/W	(Unit1) Asymmetric PWM2 Setting Register	0x0000_0000
ASPWM14	EPWM1_BA+0x28	R/W	(Unit1) Asymmetric PWM4 Setting Register	0x0000_0000

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

Bits	Descriptions	
[31:16]	-	**Reserved.**
[15:0]	**ASPWM_Duty[15:0]**	**Asymmetric PWM Duty Register** When the asymmetric PWM function is enabled, this 16-bit field determines the second time compared value after PWM counter has matched with PWM_Duty in the first half PWM cycle.

EPWM Dead-time Control Register (PDTC)

Register	Offset	R/W	Description	Reset Value
PDTC	EPWMx_BA+0x2C	R/W	EPWM Dead-time Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved					**DTEN4**	**DTEN2**	**DTEN0**
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**DTCNT**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DTCNT							

Bits	Description	
[31:19]	**Reserved**	Reserved.
[18]	**DTEN4**	**Enable Dead-time Insertion for PWMx Pair (PWM_CH4, PWM_CH5)** Dead-time insertion is only active when this pair of complementary PWM is enabled. If dead-time insertion is inactive, the outputs of pin pair are complementary without any delay. 0 = Dead-time insertion Disabled on the pin pair (PWM_CH4, PWM_CH5). 1 = Dead-time insertion Enabled on the pin pair (PWM_CH4, PWM_CH5). **Note:** x=0~1 for PWM unit0~1.
[17]	**DTEN2**	**Enable Dead-time Insertion for PWMx Pair (PWM_CH2, PWM_CH3)** Dead-time insertion is only active when this pair of complementary PWM is enabled. If dead-time insertion is inactive, the outputs of pin pair are complementary without any delay. 0 = Dead-time insertion Disabled on the pin pair (PWM_CH2, PWM_CH3). 1 = Dead-time insertion Enabled on the pin pair (PWM_CH2, PWM_CH3). **Note:** x=0~1 for PWM unit0~1.
[16]	**DTEN0**	**Enable Dead-time Insertion for PWMx Pair (PWM_CH0, PWM_CH1)** Dead-time insertion is only active when this pair of complementary PWM is enabled. If dead-time insertion is inactive, the outputs of pin pair are complementary without any delay. 0 = Dead-time insertion Disabled on the pin pair (PWM_CH0, PWM_CH1). 1 = Dead-time insertion Enabled on the pin pair (PWM_CH0, PWM_CH1). **Note:** x=0~1 for PWM unit0~1.
[15:11]	**Reserved**	Reserved.
[10:0]	**DTCNT**	**Dead-time Counter** The dead-time can be calculated according to the following formula: Dead-time = EPWMx_CLK * (DTCNT+1).

EPWM Brake Output Register (PWMB)

Register	Offset	R/W	Description	Reset Value
PWMB	EPWMx_BA+0x30	R/W	EPWM Brake Output Register	0x0000_0000

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

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**PWMB**	**PWM Brake Output** When PWM Brake is asserted, the PWM_CHn output state before polarity control will follow PWMB[n] setting, n = 0~5. 0 = PWM_CHn output before polarity control is low when Brake is asserted. 1 = PWM_CHn output before polarity control is high when Brake is asserted.

EPWM Negative Polarity Control Register (PNPC)

Register	Offset	R/W	Description	Reset Value
PNPC	EPWMx_BA+0x34	R/W	EPWM Negative Polarity 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		PNP					

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**PNP**	**PWM Negative Polarity Control** The register bit controls polarity/active state of real PWM_CHn output, n = 0~5. 0 = PWM_CHn output is active high. 1 = PWM_CHn output is active low.

EPWMF Compared Counter Register (PWMFCNT)

Register	Offset	R/W	Description	Reset Value
PWMFCNT	EPWMx_BA+0x3C	R/W	EPWMF Compared 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				**PWMFCNT**			

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3:0]	**PWMFCNT**	**PWMF Compared Counter** The register sets the count number which defines how many times of PWM period occurs to set bit PWMF (PWMSTS[2]) to request the PWM period interrupt. PWMF (PWMSTS[2]) will be set in every (PWMFCNT[3:0] + 1) times of PWM period or center point defined by INT_TYPE at PWMCON[8] occurs

EPWM Edge Interrupt Control Register (PWMEIC)

Register	Offset	R/W	Description	Reset Value
PWMEIC	EPWMx_BA+0x40	R/W	EPWM Edge Interrupt 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					**EINT4_TYPE**	**EINT2_TYPE**	**EINT0_TYPE**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved					**PWM4EI_EN**	**PWM2EI_EN**	**PWM0EI_EN**

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10]	**EINT4_TYPE**	**PWM Channel 4 Edge Interrupt Type** 0 = PWM4EF will be set if falling edge is detected at PWM_CH4. 1 = PWM4EF will be set if rising edge is detected at PWM_CH4.
[9]	**EINT2_TYPE**	**PWM Channel 2 Edge Interrupt Type** 0 = PWM2EF will be set if falling edge is detected at PWM_CH2. 1 = PWM2EF will be set if rising edge is detected at PWM_CH2.
[8]	**EINT0_TYPE**	**PWM Channel 0 Edge Interrupt Type** 0 = PWM0EF will be set if falling edge is detected at PWM_CH0. 1 = PWM0EF will be set if rising edge is detected at PWM_CH0.
[7:3]	**Reserved**	Reserved.
[2]	**PWM4EI_EN**	**Enable PWM Channel 4 Edge Interrupt** 0 = Flag PWM4EF Disabled to trigger PWM interrupt. 1 = Flag PWM4EF Enabledto trigger PWM interrupt.
[1]	**PWM2EI_EN**	**Enable PWM Channel 2 Edge Interrupt** 0 = Flag PWM2EF Disabled to trigger PWM interrupt. 1 = Flag PWM2EF Enabledto trigger PWM interrupt.
[0]	**PWM0EI_EN**	**Enable PWM Channel 0 Edge Interrupt** 0 = Flag PWM0EF Disabled to trigger PWM interrupt. 1 = Flag PWM0EF Enabledto trigger PWM interrupt.

6.9 Enhanced Input Capture Timer (ECAP)

6.9.1 Overview

This device provides up to two units of Input Capture Timer/Counter which capture function can detect the digital edge changed signal at channel inputs. Each unit has three input capture channels. The timer/counter is equipped with up counting, reload and compare-match capabilities.

6.9.2 Features

- Up to two Input Capture Timer/Counter Units, Input Capture 0 and Input Capture 1.
- Each unit has own interrupt vector
- 24-bit Input Capture up-counting timer/counter
- With noise filter in front end of input ports
- Edge detector with three options
  - Rising edge detection
  - Falling edge detection
  - Both edge detection
- Each input channel is supported with one capture counter hold register
- Captured event reset/reload capture counter option
- Supports the compare-match function

6.9.3 Input Capture Timer/Counter Architecture

Each of the input capture timer/counter unit supports 3 input channels with three programmable input signal sources. The pins ECAPx_IC0 to ECAPx_IC2 can be fed to the inputs of capture unit, besides, the QEI controller input signals (CHA, CHB and CHX), analog comparator output(Cox), OPA digital output (OPDOx), and ADC compare output (ADCMPOx) can also be internally routed to the capture inputs by software configuration. TheFigure 6–63illustrate the architecture of Input Capture.

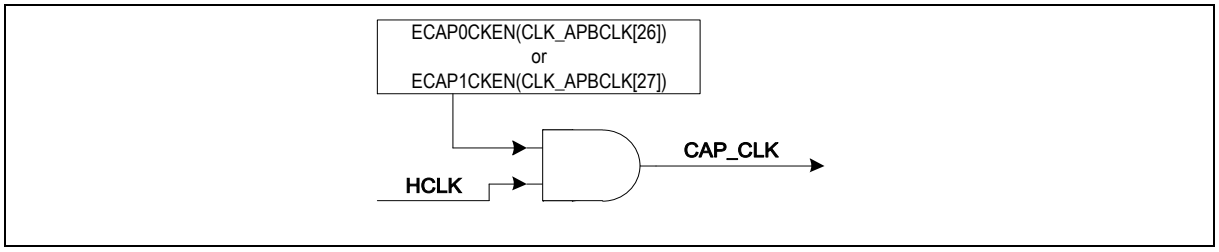


Figure 6–63 Input Capture Timer/Counter Clock Source Control

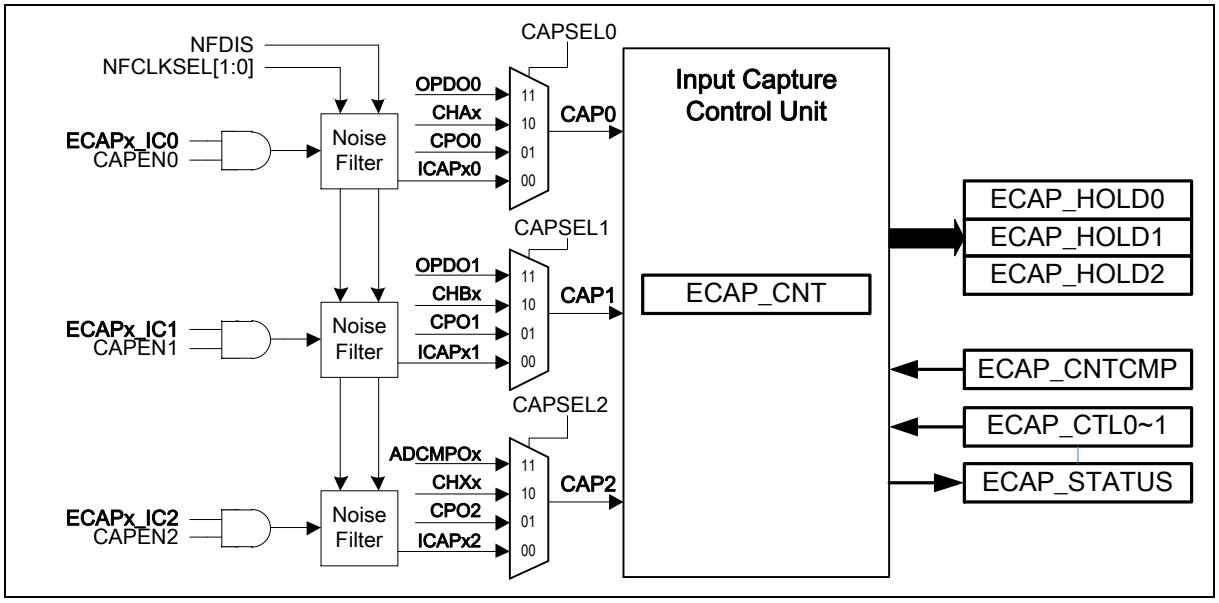


Figure 6–64 Input Capture Timer/Counter Architecture

6.9.4 Input Noise Filter

Each pin of ECAP inputs is equipped a noise filter which can filter the unwanted noise. The ECAPx_IC0, ECAPx_IC1 and ECAPx_IC2 noise filters can be disabled through bitsCAPNF_DIS (ECAP_CTLx[3]). If enabled, the capture logic required to sample 4 consecutive same capture input value in order to recognize an edge as a capture event. The interval between pulses requirement for input capture is 4 CAP_CLK clocks width. Any pulse width less than or equal to 3 CAP_CLK clockswill not have any trigger.ICAPx0, ICAPx1 and ICAPx2 are the outputs of ECAPx_IC0, ECAPx_IC1 and ECAPx_IC2 respectively after going through noise filter and polarity controlto theFigure 6–65. If the noise filter is disabled the input signals ECAPx_IC0, ECAPx_IC1 and ECAPx_IC2 are passed to the internal signals ICAPx0, ICAPx1 and ICAPx2 respectively without any delay.

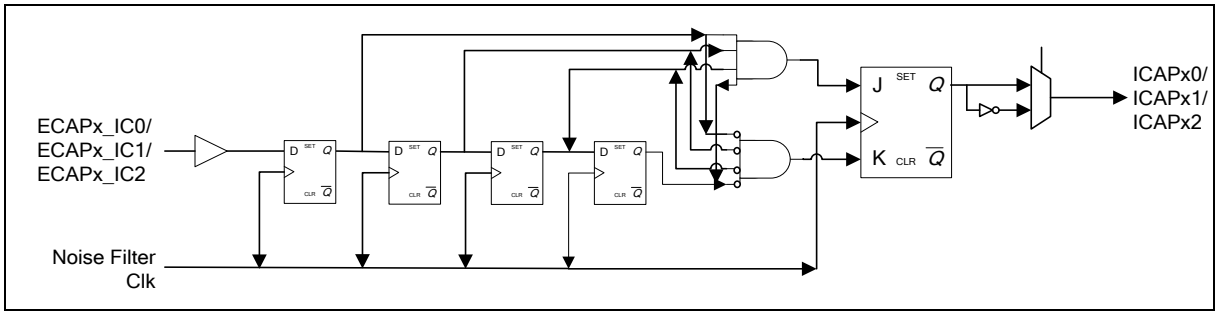


Figure 6–65 Noise Filter Structure

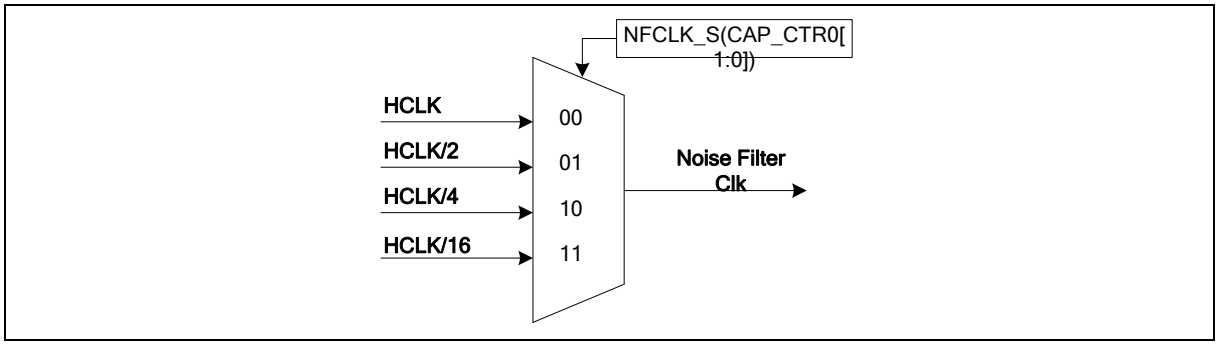


Figure 6–66Noise Filter Sampling Clock Selection

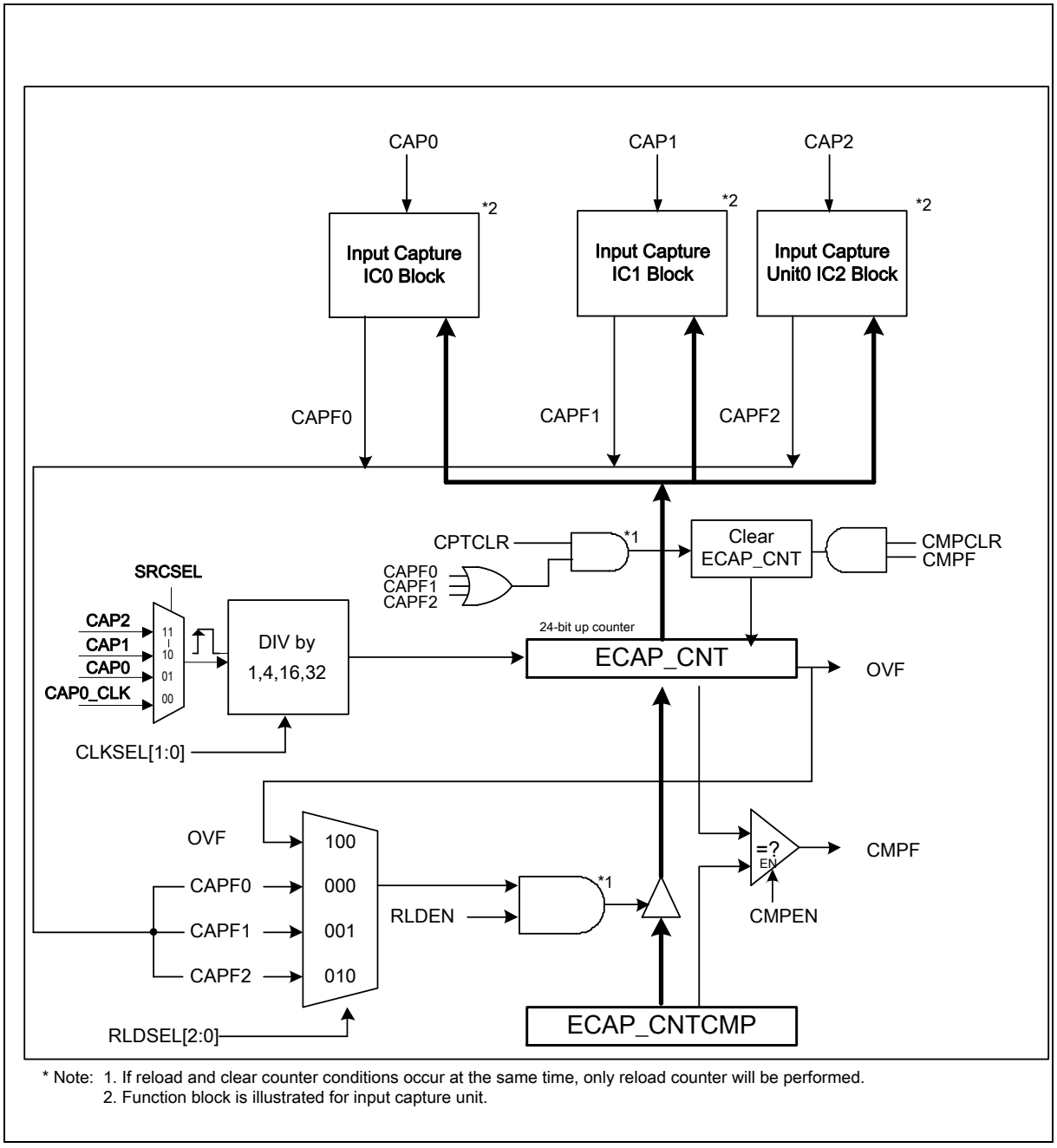
6.9.5 Operation of Input Capture Timer/Counter

The capture modules are functioned to detect and measure pulse width and period of a square wave. The input channel 0 to 2 have their own edge detector but share with one capture timer/counter i.e. CAP_CNT. The trigger option is programmable through CAPEDG in CAP_CTR1 register. It supports positive edge, negative edge and both edge triggers. Each capture module consists of an enable control bit, IC0_EN to IC2_EN. The capture counter (CAP_CNT) serves as a 24-bit up counter.It supports reload and compared modes. The Input Capture Timer/Counter Enable bit (CAP_EN) must be set to enable Input Capture Timer/Counter functions. More details are described in next sections.

6.9.5.1 Capture Function

Each time the capture input trigger is validated, the content of the free running 24 bits capture counter ECAP_CNT will be captured/transferred into the capture hold registers, ECAP_HOLD0~2, depending on which channel trigger. This action also causes the CAPFflag bits in ECAP_STATUS to be set, which will also generate an interrupt (if enabled by CAPIENx (ECAP_CTL0[16]). The CAPF0~2 flags are logical “OR” to the interrupt module. Flag is set by hardware and cleared by software. Software will have to resolve on the priority of the interrupt flags.

Setting the CPTCLR (ECAP_CTL0[26]), will allow hardware to reset capture counter (ECAP_CNT) automatically after the value of ECAP_CNT has been captured. Priority is given to reset counter after capturing the counter value in the capture register.



* Note: 1. If reload and clear counter conditions occur at the same time, only reload counter will be performed.
2. Function block is illustrated for input capture unit.

Figure 6–67 Input Capture Timer/Counter Functions Block

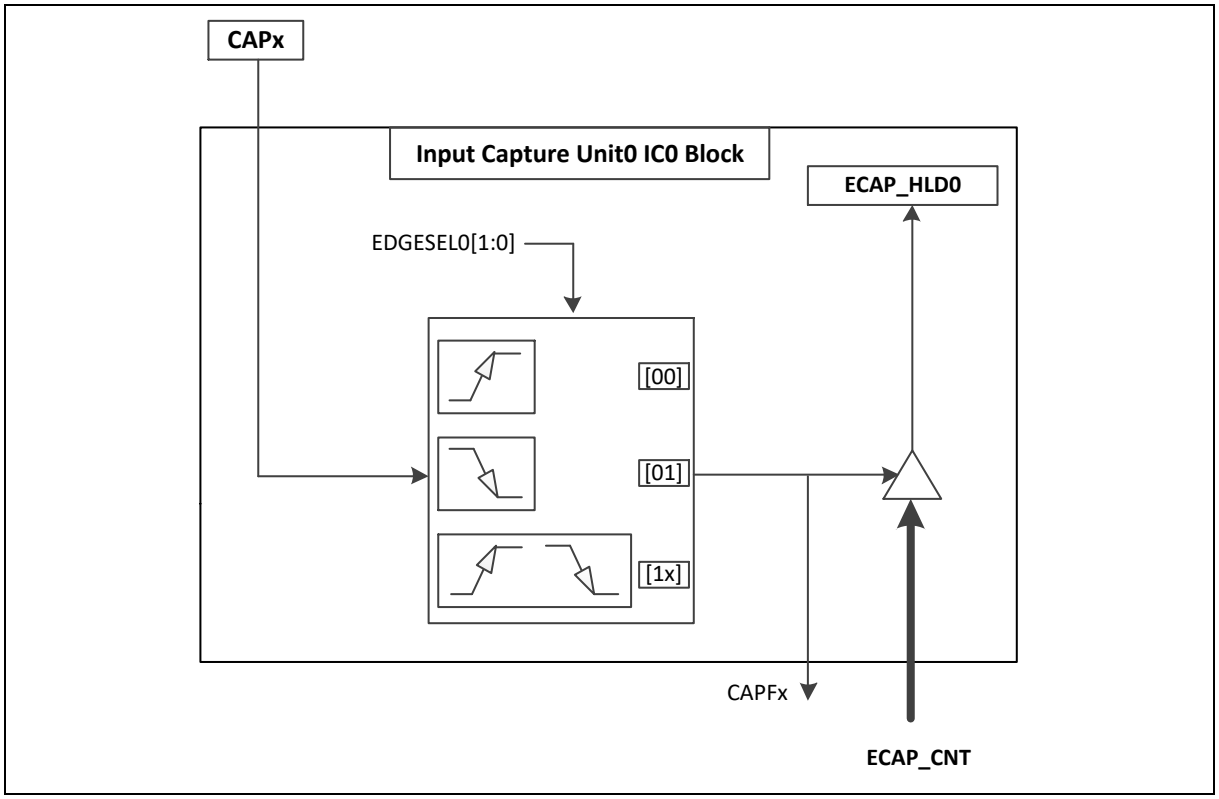


Figure 6–68 Input Capture Units

6.9.5.2 Compare Mode

The compare function is enabled by setting the CMPEN (ECAP_CTL0[28]) bit to 1. ECAP_CNTCMP will serves as a compare register. As ECAP_CNT counting up, upon matching with ECAP_CNTCMP value, CMPF (ECAP_STATUS[4]) will be set, which will generate an interrupt request if capture compare interrupt enable bit, CMPIEN (ECAP_CTL0[21]), is set. And then the timer reload from 0 and starts counting again.

Setting the CMPCLR (ECAP_CTL0[25]), will allow hardware to reset capture counter automatically after a match has occurred.

.

6.9.5.3 Reload Mode

Input Capture Timer/Counter can be also be configured for reload mode. The reload function is enabled by setting the RLDEN (ECAP_CTL0[27]) to 1. In this mode, ECAP_CNTCMP serves as a reload register. When ECAP_CNT overflows, a reload is generated that causes the contents of the ECAP_CNTCMP register to be reloaded into the ECAP_CNT register, if RLDEN is set. However, if RLDEN = 0, ECAP_CNT will be reload with 0, and count up again.

Alternatively, other reload source is also possible by the capture inputs by configuring the RLDSEL(ECAP_CTL1[10:8]). This action also sets the CAPFx flag bits in ECAP_STATUS register.

6.9.5.4 Input Capture Timer/Counter Interrupt Architecture

There are five interrupt sources for one input capture unit, each one has an interrupt flag and enable control bit, which can trigger Input Capture Timer/Counter Interrupt. Note that all the interrupt flags are set by hardware and must be cleared by software.

The Figure 6–69 demonstrates the architecture of Input Capture Timer/Counter interrupts.

.

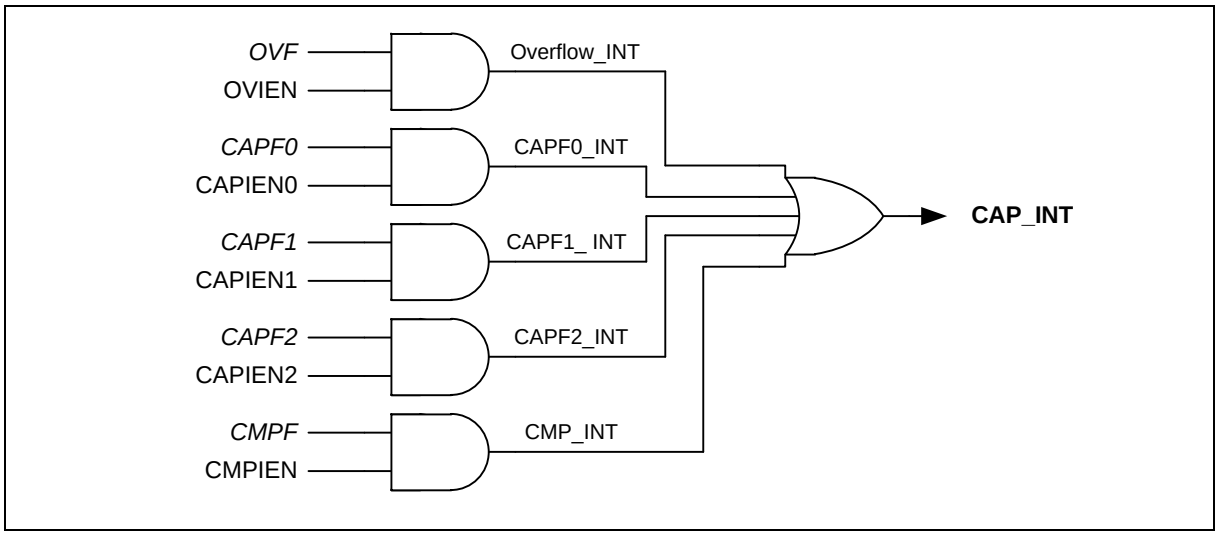


Figure 6–69 Input Capture Timer/Counter Interrupt Architecture Diagram

6.9.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
ECAP Base Address: **ECAPn_BA = 0x401B_0000 + n*0x4000** **n=0,1**				
ECAP_CNT	ECAPn_BA+0x00	R/W	Input Capture Counter (24-bit up counter)	0x0000_0000
ECAP_HOLD0	ECAPn_BA+0x04	R/W	Input Capture Counter Hold Register 0	0x0000_0000
ECAP_HOLD1	ECAPn_BA+0x08	R/W	Input Capture Counter Hold Register 1	0x0000_0000
ECAP_HOLD2	ECAPn_BA+0x0C	R/W	Input Capture Counter Hold Register 2	0x0000_0000
ECAP_CNTCMP	ECAPn_BA+0x10	R/W	Input Capture Counter Compare Register	0x0000_0000
ECAP_CTL0	ECAPn_BA+0x14	R/W	Input Capture Control Register 0	0x0000_0000
ECAP_CTL1	ECAPn_BA+0x18	R/W	Input Capture Control Register 1	0x0000_0000
ECAP_STATUS	ECAPn_BA+0x1C	R/W	Input Capture Status Register	0x0000_0000

6.9.7 Register Description

Input Capture Counter (CAP_CNT)

Register	Offset	R/W	Description	Reset Value
ECAP_CNT	ECAPn_BA+0x00	R/W	Input Capture Counter(24-bit up counter)	0x0000_0000

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

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**VAL**	**Input Capture Timer/Counter** The input Capture Timer/Counter is a 24-bit up-counting counter. The clock source for the counter is from the clock divider output which the CAP_CLK is software optionally divided by 1,4,16 or 32.

Input Capture Counter Hold Register (ECAP_HOLD0~2)

Register	Offset	R/W	Description	Reset Value
ECAP_HOLD0	ECAPn_BA+0x04	R/W	Input Capture Counter Hold Register 0	0x0000_0000
ECAP_HOLD1	ECAPn_BA+0x08	R/W	Input Capture Counter Hold Register 1	0x0000_0000
ECAP_HOLD2	ECAPn_BA+0x0C	R/W	Input Capture Counter Hold Register 2	0x0000_0000

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

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**VAL**	**Input Capture Counter Hold Register** When an active input capture channel detects a valid edge signal change, the ECAP_CNT value is latched into the corresponding holding register. Each input channel has itself holding register named by ECAP_HOLDx where x is from 0 to 2 to indicate inputs from ECAPx_IC0 to ECAPx_IC2, respectively.

Input Capture Counter Compare Register (ECAP_CNTCMP)

Register	Offset	R/W	Description	Reset Value
ECAP_CNTCMP	ECAPn_BA+0x10	R/W	Input Capture Counter Compare Register	0x0000_0000

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

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:0]	**VAL**	**Input Capture Counter Compare Register** If the compare function is enabled (CMPEN(ECAP_CTL0[28]) = 1), the compare register is loaded with the value that the compare function compares the capture counter (ECAP_CNT) with. If the reload control is enabled (RLDEN = 1), an overflow event or capture events will trigger the hardware to reload ECAP_CNTCMP into ECAP_CNT.

Input Capture Timer/Counter Control Register (ECAP_CTL0)

Register	Offset	R/W	Description	Reset Value
ECAP_CTL0	ECAPn_BA+0x14	R/W	Input Capture Control Register 0	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		CAPEN	CMPEN	RLDEN	CPTCLR	CMPCLR	CNTEN
23	22	21	20	19	18	17	16
Reserved		CMPIEN	OVIEN	Reserved	CAPIEN2	CAPIEN1	CAPIEN0
15	14	13	12	11	10	9	8
Reserved		CAPSEL2		CAPSEL1		CAPSEL0	
7	6	5	4	3	2	1	0
Reserved	CAPEN2	CAPEN1	CAPEN0	CAPNF_DIS	Reserved	NFDIS	

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**CAPEN**	**Input Capture Timer/Counter Enable Bit** 0 = Input Capture function Disabled. 1 = Input Capture function Enabled.
[28]	**CMPEN**	**Compare Function Enable Bit** The compare function in input capture timer/counter is to compare the dynamic counting ECAP_CNT with the compare register ECAP_CNTCMP, if ECAP_CNT value reaches ECAP_CNTCMP, the flag CMPF will be set. 0 = The compare function Disabled. 1 = The compare function Enabled.
[27]	**RLDEN**	**Reload Function Enable Bit** Setting this bit to enable the reload function. If the reload control is enabled, an overflow event (OVF) or capture events (CAPFx) will trigger the hardware to reload ECAP_CNTCMP into ECAP_CNT. 0 = The reload function Disabled. 1 = The reload function Enabled.
[26]	**CPTCLR**	**Input Capture Counter Cleared by Capture Events Control** If this bit is set to 1, the capture counter (ECAP_CNT) will be cleared to zero when any one of capture events (CAPF0~3) occurs. 0 = Capture events (CAPF0~3) can clear capture counter (ECAP_CNT) Disabled. 1 = Capture events (CAPF0~3) can clear capture counter (ECAP_CNT) Enabled.
[25]	**CMPCLR**	**Input Capture Counter Cleared by Compare-match Control** If this bit is set to 1, the capture counter (ECAP_CNT) will be cleared to 0 when the compare-match event (CAMCMPF = 1) occurs. 0 = Compare-match event (CAMCMPF) can clear capture counter (ECAP_CNT) Disabled. 1 = Compare-match event (CAMCMPF) can clear capture counter (ECAP_CNT) Enabled.

Bits	Description	
[24]	**CNTEN**	**Input Capture Counter Start** Setting this bit to 1, the capture counter (ECAP_CNT) starts up-counting synchronously with capture clock input (CAP_CLK). 0 = ECAP_CNT stop counting. 1 = ECAP_CNT starts up-counting.
[23:22]	**Reserved**	Reserved.
[21]	**CMPIEN**	**CMPF Trigger Input Capture Interrupt Enable Bit** 0 = The flag CMPF can trigger Input Capture interrupt Disabled. 1 = The flag CMPF can trigger Input Capture interrupt Enabled.
[20]	**OVIEN**	**OVF Trigger Input Capture Interrupt Enable Bit** 0 = The flag OVUNF can trigger Input Capture interrupt Disabled. 1 = The flag OVUNF can trigger Input Capture interrupt Enabled.
[19]	**Reserved**	Reserved.
[18]	**CAPIEN2**	**Input Capture Channel 2 Interrupt Enable Bit** 0 = The flag CAPF2 can trigger Input Capture interrupt Disabled. 1 = The flag CAPF2 can trigger Input Capture interrupt Enabled.
[17]	**CAPIEN1**	**Input Capture Channel 1 Interrupt Enable Bit** 0 = The flag CAPF1 can trigger Input Capture interrupt Disabled. 1 = The flag CAPF1 can trigger Input Capture interrupt Enabled.
[16]	**CAPIEN0**	**Input Capture Channel 0 Interrupt Enable Bit** 0 = The flag CAPF0 can trigger Input Capture interrupt Disabled. 1 = The flag CAPF0 can trigger Input Capture interrupt Enabled.
[15:14]	**Reserved**	Reserved.
[13:12]	**CAPSEL2**	**CAP2 Input Source Selection** 00 = CAP2 input is from pin RCAPx_IC2. 01 = CAP2 input is from CO2(ACMPSR[10]). 10 = CAP2 input is from signal CHX of QEI controller unit x. 10 = Reserved. 11 = CAP2 input is from signal ADCMPOx (ADC compare output x). **Note:** Input capture unit n matches QEI or comparator unit x, where x = 0~1.
[11:10]	**CAPSEL1**	**CAP1 Input Source Selection** 00 = CAP1 input is from pin ECAPx_IC1. 01 = CAP1 input is from CO1(ACMPSR[9]). 10 = CAP1 input is from signal CHB of QEI controller unit x. 10 = Reserved. 11 = CAP1 input is from OPDO1 (OPASR[1]). **Note:** Input capture unit n matches QEI or comparator unit x, where x = 0~1.

Bits	Description	
[9:8]	**CAPSEL0**	**CAP0 Input Source Selection** 00 = CAP0 input is from pin ECAPx_IC0. 01 = CAP0 input is from CO0(ACMPSR[8]). 10 = CAP0 input is from signal CHA of QEI controller unit x. 10 = Reserved. 11 = CAP0 input is from OPDO0 (OPASR[0]). **Note:** Input capture unit n matches QEI or comparator unit x, where x = 0~1.
[7]	**Reserved**	Reserved.
[6]	**CAPEN2**	**Pin ECAPx_IC2 Input to Input Capture Unit Enable Bit** 0 = ECAPx_IC2 input to Input Capture Unit Disabled. 1 = ECAPx_IC2 input to Input Capture Unit Enabled.
[5]	**CAPEN1**	**Pin ECAPx_IC1 Input to Input Capture Unit Enable Bit** 0 = ECAPx_IC1 input to Input Capture Unit Disabled. 1 = ECAPx_IC1 input to Input Capture Unit Enabled.
[4]	**CAPEN0**	**Pin ECAPx_IC0 Input to Input Capture Unit Enable Bit** 0 = ECAPx_IC0 input to Input Capture Unit Disabled. 1 = ECAPx_IC0 input to Input Capture Unit Enabled.
[3]	**CAPNF_DIS**	**Input Capture Noise Filter Disable Bit** 0 = Noise filter of Input Capture Enabled. 1 = Noise filter of Input Capture Disabled.
[2]	**Reserved**	Reserved.
[1:0]	**NFDIS**	**Noise Filter Clock Pre-divide Selection** To determine the sampling frequency of the Noise Filter clock 00 = CAP_CLK. 01 = CAP_CLK/2. 10 = CAP_CLK/4. 11 = CAP_CLK/16.

Input Capture Timer/Counter Control Register (ECAP_CTL1)

Register	Offset	R/W	Description	Reset Value
ECAP_CTL1	ECAPn_BA+0x18	R/W	Input Capture Control Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved						SRCSEL	
15	14	13	12	11	10	9	8
Reserved	CLKSEL			Reserved	RLDSEL		
7	6	5	4	3	2	1	0
Reserved		EDGESEL2		EDGESEL1		EDGESEL0	

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17:16]	**SRCSEL**	**Capture Timer/Counter Clock Source Selection** Select the capture timer/counter clock source. 00 = CAP_CLK (default). 01 = CAP0. 10 = CAP1. 11 = CAP2.
[15]	**Reserved**	Reserved.
[14:12]	**CLKSEL**	**Capture Timer Clock Divide Selection** The capture timer clock has a pre-divider with eight divided options controlled by CLKSEL[1:0]. 000 = CAP_CLK/1. 001 = CAP_CLK/4. 010 = CAP_CLK/16. 011 = CAP_CLK/32. 100 = CAP_CLK/64. 101 = CAP_CLK/96. 110 = CAP_CLK/112. 111 = CAP_CLK/128.
[11]	**Reserved**	Reserved.

Bits	Description	
[10:8]	**RLDSEL**	**ECAP_CNT Reload Trigger Source Selection** If the reload function is enabled RLDEN (ECAP_CTL0[27]) = 1, when a reload trigger event comes, the ECAP_CNT is reloaded withECAP_CNTCMP. RLDSEL[2:0] determines the ECAP_CNT reload trigger source 000 = CAPF0. 001 = CAPF1. 010 = CAPF2. 100 = OVF. Others = Reserved.
[7:6]	**Reserved**	Reserved.
[5:4]	**EDGESEL2**	**Channel 2 Captured Edge Selection** Input capture can detect falling edge change or rising edge change only, or one of both edge changes. 00 = Detect rising edge. 01 = Detect falling edge. 1x = Detect either rising or falling edge.
[3:2]	**EDGESEL1**	**Channel 1 Captured Edge Selection** Input capture can detect falling edge change only, rising edge change only or one of both edge change 00 = Detect rising edge. 01 = Detect falling edge. 1x = Detect either rising or falling edge.
[1:0]	**EDGESEL0**	**Channel 0 Captured Edge Selection** Input capture can detect falling edge change only, rising edge change only or one of both edge change 00 = Detect rising edge. 01 = Detect falling edge. 1x = Detect either rising or falling edge.

Input Capture Timer/Counter Status Register (ECAP_STATUS)

Register	Offset	R/W	Description	Reset Value
ECAP_STATUS	ECAPn_BA+0x1C	R/W	Input Capture Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved		OVF	CMPF	Reserved	CAPF2	CAPF1	CAPF0

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**OVF**	**Input Capture Counter Overflow Flag** Flag is set by hardware when input capture up counter (ECAP_CNT) overflows from 0x00FF_FFFF to zero. 0 = No overflow occurs in ECAP_CNT. 1 = ECAP_CNT overflows. **Note:** This bit is only cleared by writing 1 to it.
[4]	**CMPF**	**Input Capture Compare-match Flag** If the input capture compare function is enabled, the flag is set by hardware while capture counter (ECAP_CNT) up counts and reach to the ECAP_CNTCMP value. 0 = ECAP_CNT does not match with ECAP_CNTCMP value. 1 = ECAP_CNT counts to the same as ECAP_CNTCMP value. **Note:** This bit is only cleared by writing 1 to it.
[3]	**Reserved**	Reserved.
[2]	**CAPF2**	**Input Capture Channel 2 Captured Flag** When the input capture channel 2 detects a valid edge change at CAP2 input, it will set flag CAPF2 to high. 0 = No valid edge change is detected at CAP2 input. 1 = A valid edge change is detected at CAP2 input. **Note:** This bit is only cleared by writing 1 to it.
[1]	**CAPF1**	**Input Capture Channel 1 Captured Flag** When the input capture channel 1 detects a valid edge change at CAP1 input, it will set flag CAPF1 to high. 0 = No valid edge change is detected at CAP1 input. 1 = A valid edge change is detected at CAP1 input. **Note:** This bit is only cleared by writing 1 to it.

Bits	Description	
[0]	**CAPF0**	**Input Capture Channel 0 Captured Flag** When the input capture channel 0 detects a valid edge change at CAP0 input, it will set flag CAPF0 to high. 0 = No valid edge change is detected at CAP0 input. 1 = A valid edge change is detected at CAP0 input. **Note:** This bit is only cleared by writing 1 to it.

6.10 Quadrature Encoder Interface (QEI)

6.10.1 Overview

There are two 32-bit QEI controllers in this device. The Quadrature Encoder Interface (QEI) decodes speed of rotation and motion sensor information. It can be used inany application that uses a quadrature encoder forfeedback.

6.10.2 Features

- Up to two QEI controllers, QEI0 and QEI1.
- Two QEI phase inputs, QEA and QEB; One Index input.
- One QEI control register (QEI_CTL) and one QEI Status Register (QEI_STATUS)
- Four Quadrature encoder pulse counter operation modes:
  - Mode0: x4 free-counting mode
  - Mode1: x2 free-counting mode
  - Mode2: x4 compare-counting mode
  - Mode3: x2 compare-counting mode
- Encoder Pulse Width measurement mode

6.10.3 QEI Architecture

The QEI controller inputs, QEA and QEB, accept the outputs from a quadrature encoded source, such as incremental optical shaft encoder. Two channels, A and B, nominally 90 degrees out of phase, are required. A quadrature encoder usually provides an index signal (to pin IDX) which can be used to indicate an absolute position. There is a noise filter and polarity control for each signal before QEI control unit.

The Figure 6–71 llustrate the architecture of Quadrature Encoder Interface Controller

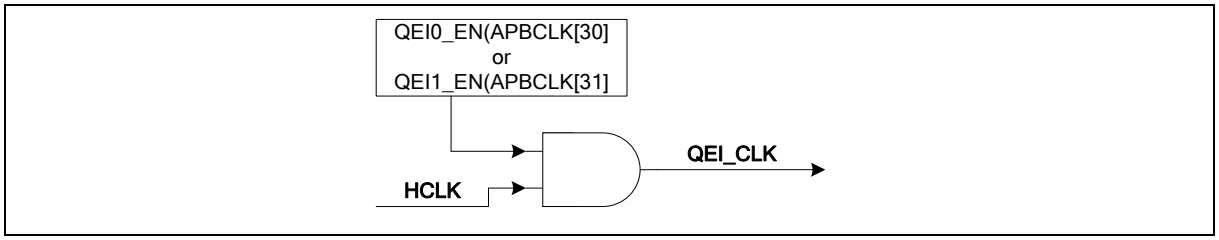


Figure 6–70 QEI Clock Source Control

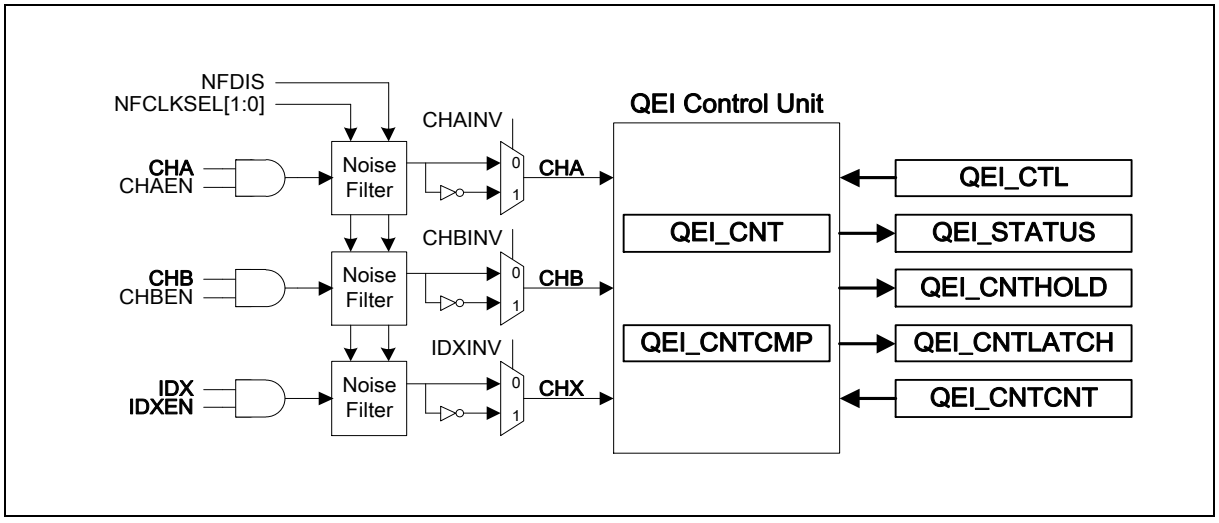


Figure 6–71 QEI Block Diagram

The QEI control logic detects the relation of phase lead/lag between the filtered signals CHA and CHB and CHX to produce direction indication bit (DIRSTS) and clock (QCLK) to control pulse counter. The comparator/reload logic compares the pulse counter and maximum count and control the function of reloading pulse counter in compare-counting mode. In Free-counting mode the pulse counter (QEI_CNT) will count until the 0xFFFF_FFFF value; while in Compare-counting mode the pulse counter will counts until the QEI_CNTMAX value and the pulse counter will be reset to zero to restart the next cyclic counting.

6.10.4 Input Noise Filter

Each pin of QEI inputs is equipped a noise filter which can filter the unwanted noise . The QEA, QEB and IDX noise filters can be disabled through bitsQEINF_DIS. If enabled, the capture logic required to sample 4 consecutive same capture input value in order to recognize an edge as a capture event. A possible implementation of digital noise filter is as follow; the interval between pulses requirement for input capture is 4 QEI_CLK clocks width. Any pulse width less than or equal to 3 QEI_CLK clockswill not have any trigger.CHA, CHB and CHX are the outputs of QEA, QEB and IDX respectively after going through noise filter and polarity control to theFigure 6–72. If the noise filter is disabled the input signals QEA, QEB and IDX are passed to the internal signals CHA, CHB and CHX respectively without any delay.

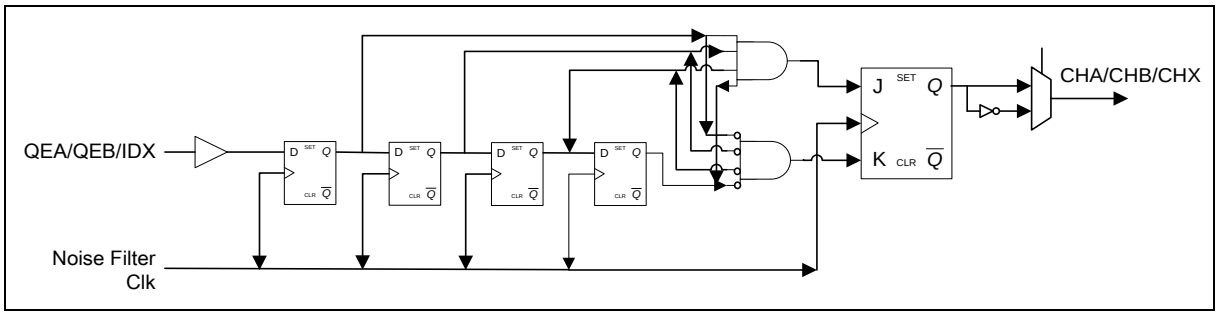


Figure 6–72QEI input Noise Filter

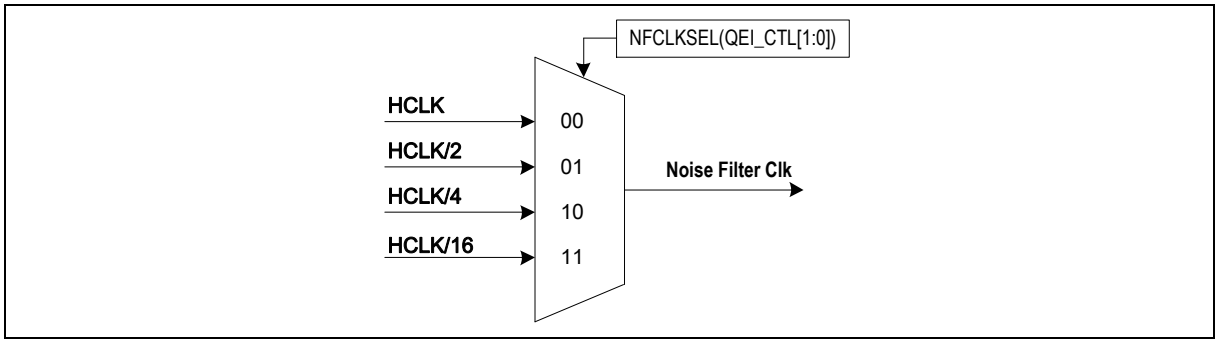


Figure 6–73 Noise Filter Sampling Clock Selection

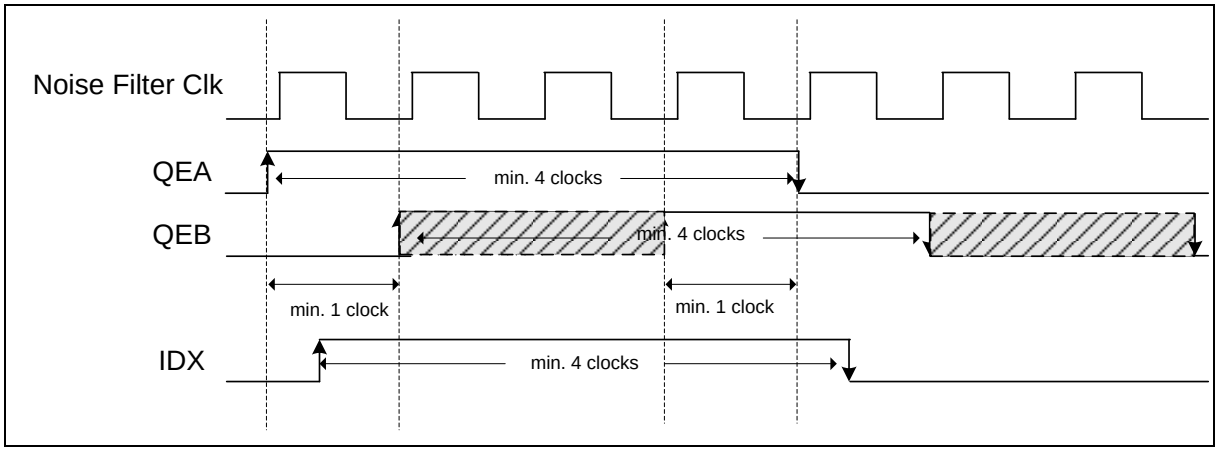


Figure 6–74 QEA/QEB/IDX Timing Requirement through Noise Filter

6.10.5 Operation of Quadrature Encoder Interface

There are four Quadrature encoder pulse counter operation modes

- Mode0: x4 free-counting mode
- Mode1: x2 free-counting mode
- Mode2: x4 compare-counting mode
- Mode3: x2 compare-counting mode

6.10.5.1 Free-counting mode

The quadrature encoder pulse counter (QEI_CNT) up or down counts according direction indication bit DIRSTS (QEI_STATUS [8]). When overflow or underflow occurs, it sets flag OVUNF (QEI_STATUS[2]). Refer to Figure 6–75 and Figure 6–76for detailed timing.

.

6.10.5.2 Compare-counting mode

Pulse counter up or down counts according to direction indication bit DIRSTS (QEI_STATUS[8]). On up counting, flag OVUNF (QEI_STATUS[2]) will be asserted when QEI_CNT overflows from QEI_CNTMAX to zero I next CHA edge for x2 counting mode, and on CHA/CHB edge for x4 counting mode. On down counting, flag OVUNF (QEI_STATUS[2]) will be asserted when QEI_CNT underflows from zero to QEI_CNTMAX on the next CHA edge for x2 counting mode, and on CHA/CHB edge for x4 counting mode. This mode provides the position of a rotor to user. If a quadrature encoder output 1024 pulses to CHA per round, user can write QEI_CNTMAX and QEI_CNTCMP with 4095 in x4 mode or 2047 in x2 mode and reset QEI_CNT at initial before compare-counting mode is active. When the QEI_CNT overflows from QEI_CNTCMP, here QEI_CNTCMP should be preset the same value as

QEI_CNTMAX, it means rotor runs one round on next CHA/CHB edge. Refer to Figure 6–75and Figure 6–76for detailed timing.

6.10.5.3 X4/X2 counting modes

In x4 counting mode, the pulse counter increases or decreases one on every CHA and CHB edge based on the phase relationship of CHA and CHB signals.

QEI x4 Counting mode provides for a finer resolution I rotor position, since the counter increments ordecrements more frequently for each QEA/QEB inputpulse pair than in QEI x2 mode. This mode is selectedby setting the QEI Counting Mode Selection bitsMODE(QEI_CTL[9:8]) to 00b or 01b. Inthis mode, the QEI logic detects every edge on every QEA and QEB input edges.

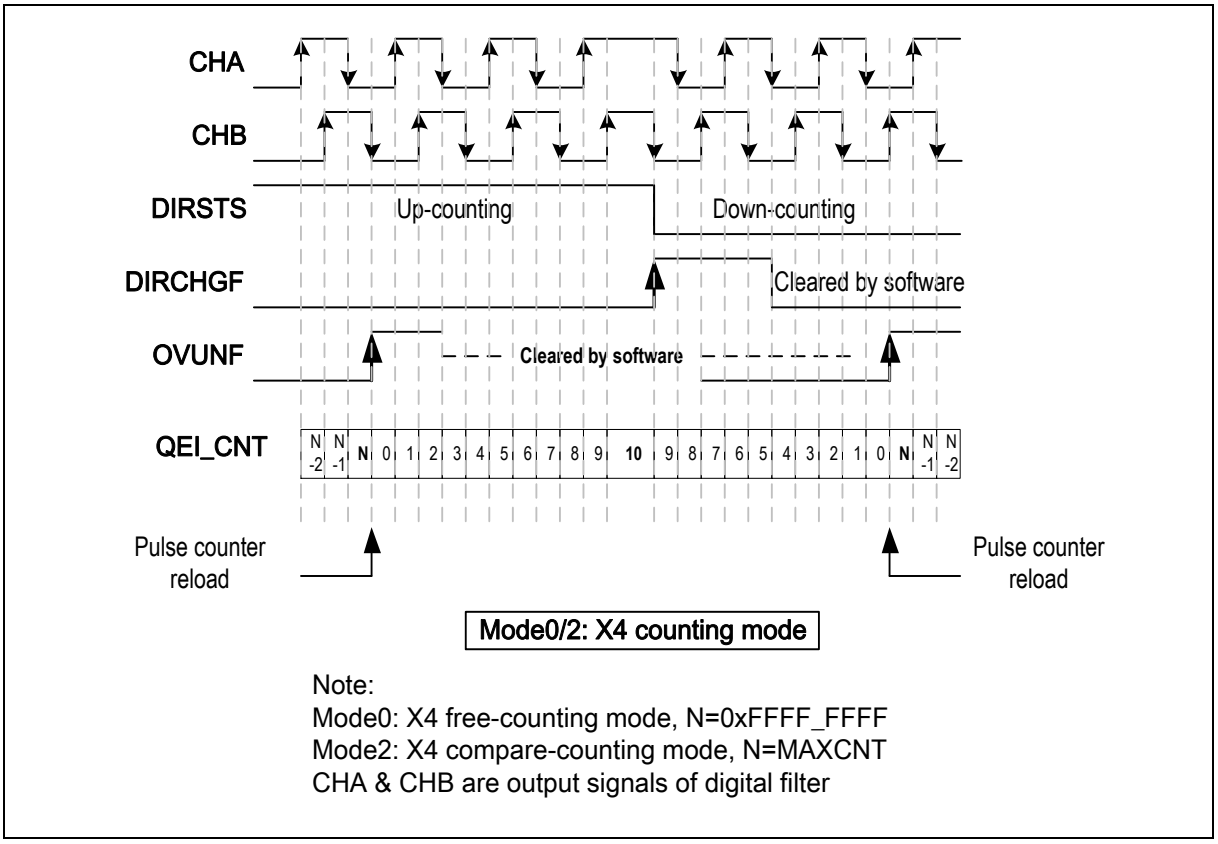


Figure 6–75 X4 Counting Mode

In **x2 counting mode**, the pulse counter increases or decreases one on every CHA edge based on the phase relationship of CHA and CHB signals.

QEI x2 Counting mode is selected by setting the QEI Counting Mode Selection bits (QEI_CTR[9:8]) to [0:1] or [1:1]. Inthis mode, the QEI logic detects every edge on theQEA input only. Every rising and falling edge on theQEA signal clocks the pulse counter.

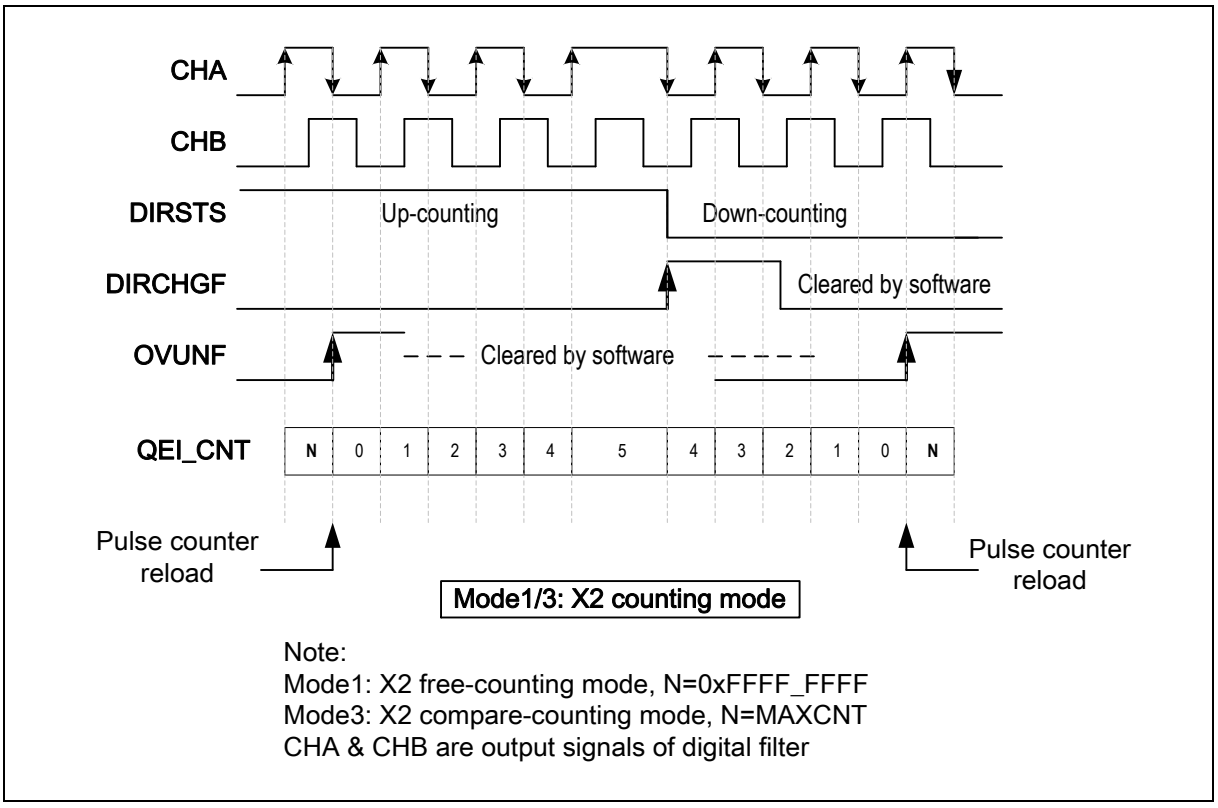


Figure 6–76 X2 Counting Mode

6.10.5.4 Direction of Count

If CHA lead CHB, the pulse counter is increased by 1. If CHA lags CHB, the pulse counter is decreased by 1. The QEI control logic generates a signal that setsthe DIRSTS (QEI_STATUS[8]); this in turndetermines the direction of the count. When CHAleads CHB, DIRSTS is set as 1, and the positioncounter increments on every active edge. WhenCHAlags CHB, DIRSTS is cleared, and theposition counter decrements on every active edge. Refer toTable 6-12.

Current Signal Detected	**Previous Signal Detected**				**DIR (Counting Direction)**
	Rising		**Falling**		
	CHA	**CHB**	**CHA**	**CHB**	
CHA rising				✓	1 (Increment)
		✓			0 (Decrement)
			✓		Toggle (direction change)
CHA falling				✓	0 (Decrement)
		✓			1 (Increment)

	✓				Toggle (direction change)
CHB rising	✓				1 (Increment)
			✓		0 (Decrement)
				✓	Toggle (direction change)
CHB falling			✓		1 (Increment)
	✓				0 (Decrement)
		✓			Toggle (direction change)

Table 6-12 Direction of Count

6.10.5.5 Up-Counting

Under the forward direction the DIRSTS bit is 1 when up-counting. Software needs to clear the OVUNF (QEI_STATUS[2]). For the free-counting mode the QEI_CNT counter will count until it matches 0xFFFF_FFFF and next edges on the forward direction will set the bit OVUNF (QEI_STATUS[2]) high and reset QEI_CNT to zero. For compare-counting mode the QEI_CNT counter counts until the QEI_CNTMAX value and next edges on the forward direction will set the bit OVUNF (QEI_STATUS[2]) high and reset QEI_CNT to zero. Changes of direction trigger a down-count and QEI_CNT decreasing in counter value. For X2 mode, only CHA edge will set OVUNF (QEI_STATUS[2]) while for X4 mode both CHA and CHB edges will set OVUNF (QEI_STATUS[2]).

6.10.5.6 Down-Counting

A change of direction will cause the counter to downcount for X2/X4 counting mode. It is indicated with the DIRSTS bit as 0 and DIRCHGF (QEI_STATUS[3]) flag is set to 1. At this stage the QEI_CNT will start to down-count. In free-counting mode the pulse counter will reload with 0xFFFF_FFFF when it down counts to zero and sets OVUNF (QEI_STATUS[2]) to high in the next edge. The pulse counter will reload with QEI_CNTMAX when it down counts to zero in compare-counting mode and sets OVUNF (QEI_STATUS[2]) to high in the next edge. For X2 mode, only CHA edge will set OVUNF (QEI_STATUS[2]) while for X4 mode both CHA and CHB edges will set OVUNF(QEI_STATUS[2]).

6.10.6 Compare Function

The compare function in QEI controller is to compare the dynamic counting QEI_CNT with the compare register QEI_CNTCMP. When QEI_CNT up or down counts and reaches QEI_CNTCMP, the flag CMPF will be set. Set bit CMP_EN (QEI_CTR[28]) to one to enable the compare function otherwise disable it.

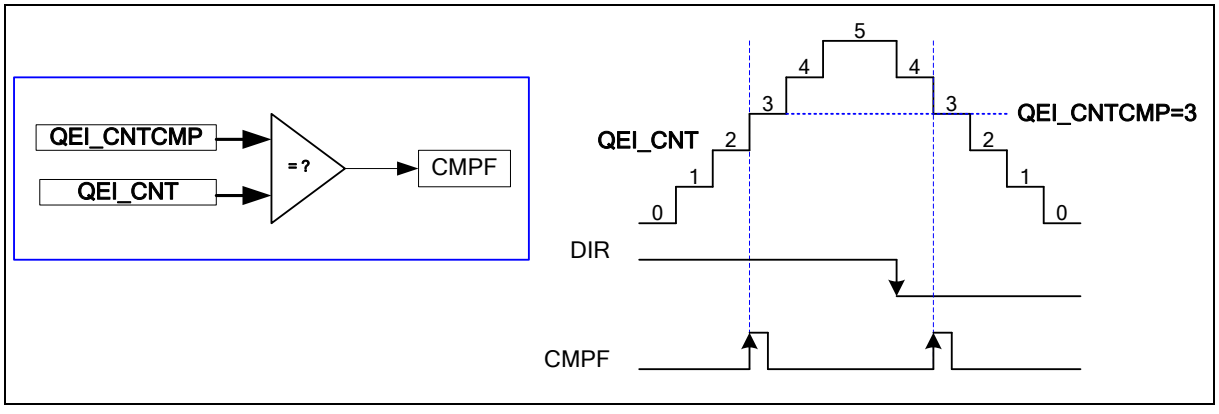


Figure 6–77 Compare Operation

6.10.7 Reload Counter by Pin IDX

The QEI_CNT counter can be reset to 0 or reload with the content of QEI_MAXCNT by the signal CHX (the filtered and polarity-set output of pin IDX) trigger. When the IDX Reload bit **IDXRLD_EN** (QEI_CTR[27]) is set, a rising edge of CHX causes QEI controller to reset the QEI_CNT to 0 if the counter is in up-counting; if the counter is in down-counting the rising edge of CHX causes the QEI controller reload the QEI_CNT with the content of QEI_MAXCNT. Refer to Figure 6–78 for the detail.

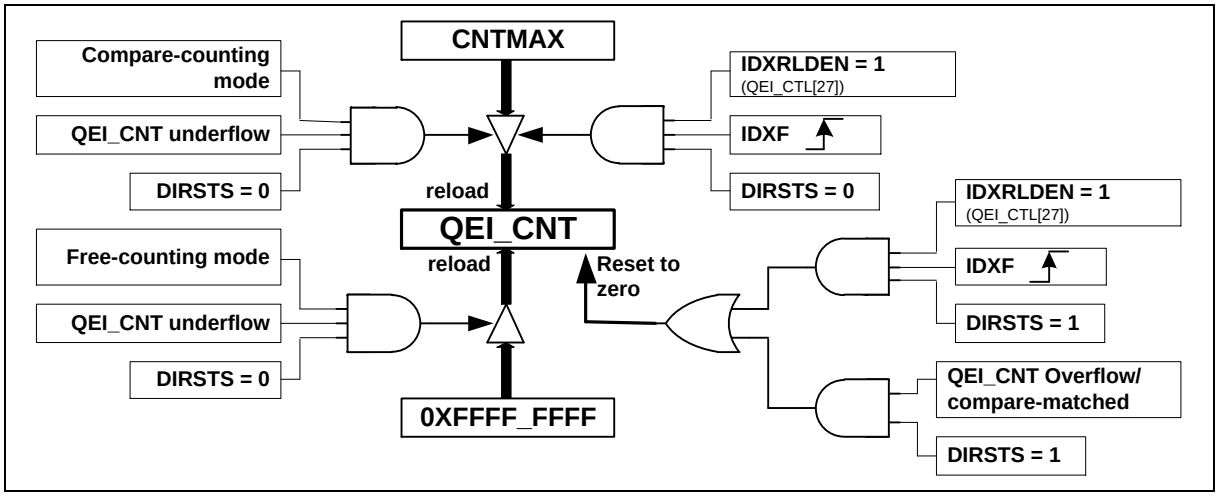


Figure 6–78 QEI_CNT Reload/Reset Control

6.10.8 Capture Qei Counter

If the bit HOLDCNT (QEI_CTL[24]) is set, the QEI_CNT content will be captured into QEI Counter Hold Register (QEI_CNTHOLD), the data will be held until the next HOLDCNT(QEI_CTL[24]) trigger comes. The bit HOLDCNT can be set by writing 1 to it or the rising edge of timers interrupt flags TIF (TIMERx_INTSTS[0]).

Note: The bit HOLDCNT is automatically cleared by hardware after QEI_CNTHOLD captures the content of QEI counter.

If the bit IDXLATEN (QEI_CTL[25]) is set, the QEI_CNT content will be latched into QEI Counter Index Latch Register (QEI_CNTLATCH) at every rising edge of CHX signal.

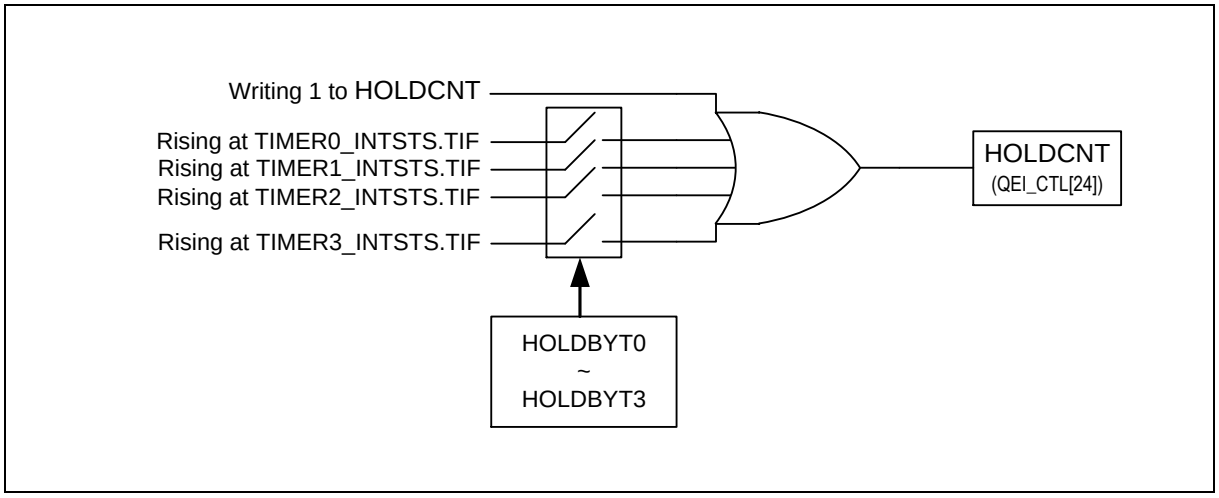


Figure 6–79 Trigger Control of Capturing QEP Counter

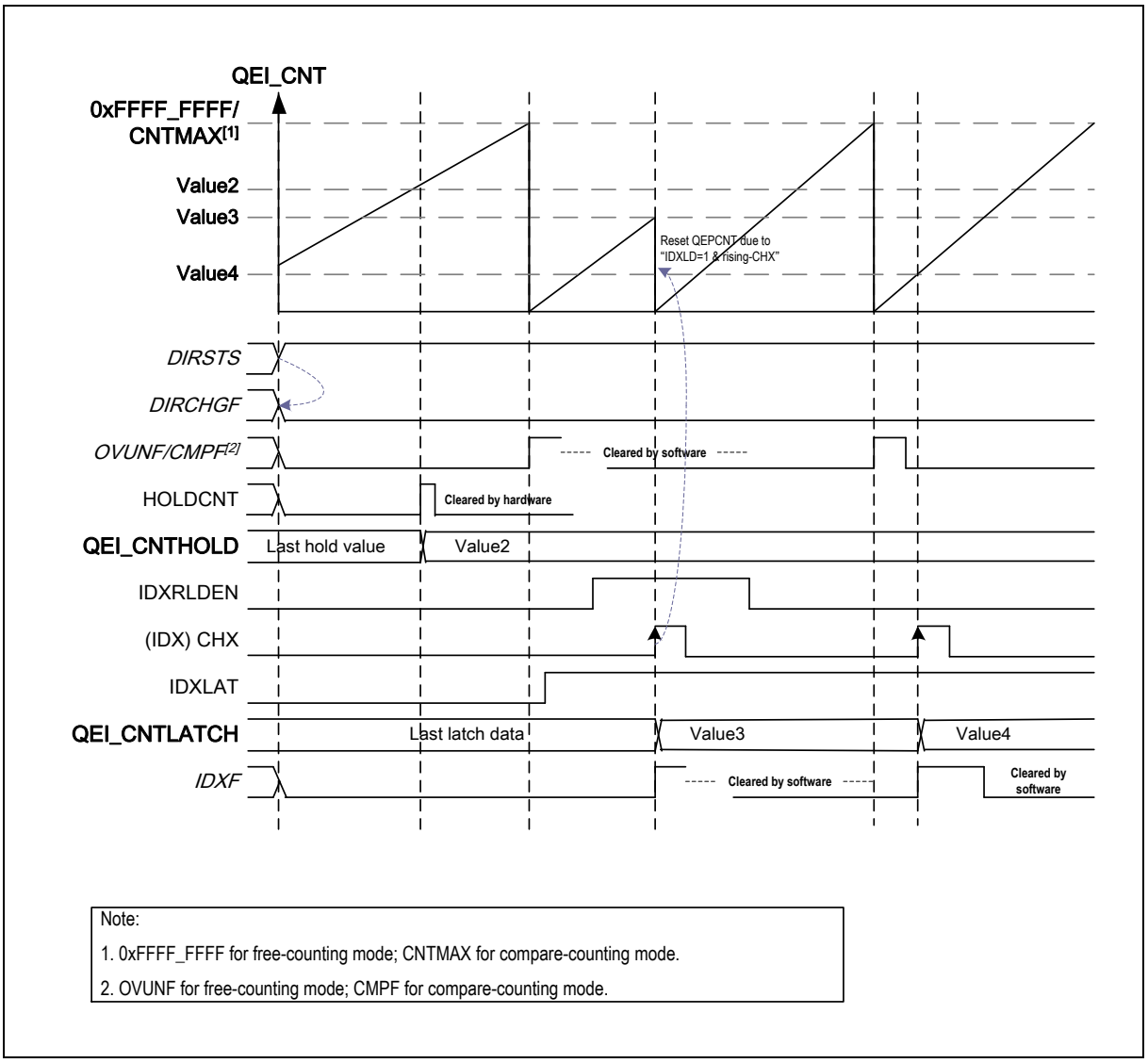


Figure 6–80 Capture and Latch QEP Counter

6.10.9 QEI Interrupt Architecture

There are four interrupt sources, each one of them has an interrupt flag and enable control bit, can trigger QEI Interrupt. When QEI counter is up-counting and QEI_CNT overflows or down-counting and underflows, the Overflow/Underflow flag (OVUNF in QEI_STS[2]) will be set by hardware and it will trigger QEI Interrupt request if bit OVUNIEN (QEI_CTR[16]) is high. When QEI controller detects the encoder rotation change, it toggles the direction indication bit DIRSTS (QEI_STS[8]) and the flag DIRCHGF (QEI_STS[3]) will be set by hardware that requests the QEI interrupt if bit DIRIEN (QEI_CTR[17]) is set. When the QEI counter counting value is equal to the value of QEI Counter Compare Register (QEI_CNTCMP), the flag CMPF (QEI_STS[1]) will be set by hardware and the QEI Interrupt will be requested if bit CMPIEN (QEI_CTR[18]) is high. When QEI controller detects a rising edge at signal CHX (the filtered and polarity-set output of pin IDX), the flag IDXF will set by hardware and the QEI interrupt will be requested if bit IDXIEN (QEI_CTR[19]) is set. **Note that the four flags, OVUNF, DIRCHGF, CMPF and IDXF are set by hardware and must be cleared by software.** Figure 6–81 demonstrates the architecture of Quadrature Encoder Interface Controller interrupts.

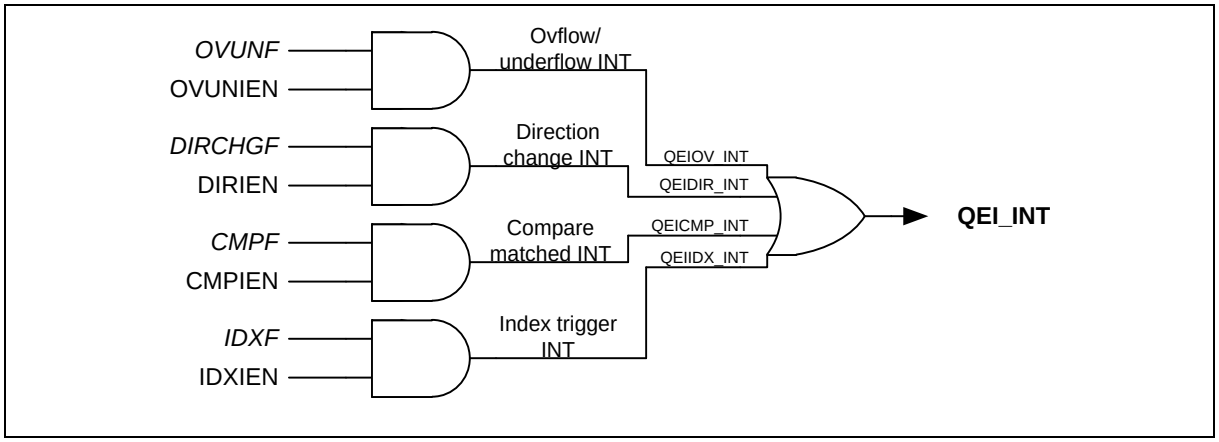


Figure 6–81 Quadrature Encoder Interface Interrupt Architecture Diagram

6.10.10 Register Map

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

Register	Offset	R/W	Description	Reset Value
QEI Base Address: **QEIn_BA = 0x401C_0000 + n*0x4000** **n = 0, 1**				
QEI_CNT	QEIn_BA+0x00	R/W	QEI Pulse Counter	0x0000_0000
QEI_CNTHOLD	QEIn_BA+0x04	R/W	QEI Pulse Counter Hold Register	0x0000_0000
QEI_CNTLATCH	QEIn_BA+0x08	R/W	QEI Pulse Counter Index Latch Register	0x0000_0000
QEI_CNTCMP	QEIn_BA+0x0C	R/W	QEI Pulse Counter Compare Register	0x0000_0000
QEI_CNTMAX	QEIn_BA+0x14	R/W	QEI Pre-set Maximum Count Register	0x0000_0000
QEI_CTL	QEIn_BA+0x18	R/W	QEI Controller Control Register	0x0000_0000
QEI_STATUS	QEIn_BA+0x2C	R/W	QEI Controller Status Register	0x0000_0000

6.10.11 Register Description

QEI Pulse Counter Register (QEI_CNT)

Register	Offset	R/W	Description	Reset Value
QEI_CNT	QEIn_BA+0x00	R/W	QEI Pulse CounterRegister	0x0000_0000

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

Bits	Description	
[31:0]	**VAL**	**Quadrature Encoder Pulse Counter** A 32-bit up/down counter. When an effective phase pulse is detected, this counter is increased by one if the bit DIRSTS (QEI_STATUS[8]) is one or decreased by one if the bit DIRSTS is zero. This register performs an integrator which count value is proportional to the encoder position. The pulse counter may be initialized to a predetermined value by one of three events occurs: 1. Software is written if QEIEN (QEI_CTL[29]) = 0. 2. Compare-match event if QEIEN=1 and QEI is in compare-counting mode. 3. Index signal change if QEIEN=1 and IDXRLDEN (QEI_CTL[27])=1.

QEI Pulse Counter Hold Register (QEI_CNTHOLD)

Register	Offset	R/W	Description	Reset Value
QEI_CNTHOLD	QEIn_BA+0x04	R/W	QEI Pulse Counter Hold Register	0x0000_0000

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

Bits	Description	
[31:0]	VAL	**Quadrature Encoder Pulse Counter Hold Register** When bit HOLDCNT (QEIx_CTL[24]) goes from low to high, the QEI_CNT value is copied into QEI_CNTHOLD register.

QEI Pulse Counter Index Latch Register (QEI_CNTLATCH)

Register	Offset	R/W	Description	Reset Value
QEI_CNTLATCH	QEIn_BA+0x08	R/W	QEI Pulse Counter Index Latch Register	0x0000_0000

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

Bits	Description	
[31:0]	VAL	**Quadrature Encoder Pulse Counter Index Latch** When the IDXF (QEI_STATUS[0]) bit is set, the QEI_CNT value is copied into QEI_CNTLATCH register.

QEI Pulse Counter Compare-Match Register (QEI_CNTCMP)

Register	Offset	R/W	Description	Reset Value
QEI_CNTCMP	QEIn_BA+0x0C	R/W	QEI Pulse Counter Compare Register	0x0000_0000

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

Bits	Description	
[31:0]	VAL	**Quadrature Encoder Pulse Counter Compare** If the QEI controller is in the compare-counting mode CMPEN (QEI_CTL[28]) =1, when the value of QEI_CNT matches the value of VAL the bit CMPF will be set. This register is software writable.

QEI Preset Maximum Count Register (QEI_CNTMAX)

Register	Offset	R/W	Description	Reset Value
QEI_CNTMAX	QEIn_BA+0x14	R/W	QEI Pre-set Maximum Count Register	0x0000_0000

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

Bits	Description	
[31:0]	**VAL**	**Quadrature Encoder Preset Maximum Count** This register value determined by user stores the maximum value which may be the number of the quadrature encoder pulses in a revolution for the QEI controller compare-counting mode.

Quadrature Encoder Interface Control Register (QEI_CTL)

Register	Offset	R/W	Description	Reset Value
QEI_CTL	QEIn_BA+0x18	R/W	QEI Controller Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved		QEIEN	CMPEN	IDXRLDEN	Reserved	IDXLATEN	HOLDCNT
23	22	21	20	19	18	17	16
HOLDTMR3	HOLDTMR2	HOLDTMR1	HOLDTMR0	IDXIEN	CMPIEN	DIRIEN	OVUNIEN
15	14	13	12	11	10	9	8
Reserved	IDXINV	CHBINV	CHAINV	Reserved		MODE	
7	6	5	4	3	2	1	0
Reserved	IDXEN	CHBEN	CHAEN	NFDIS	Reserved	NFCLKSEL	

Bits	Description	
[31:30]	**Reserved**	Reserved.
[29]	**QEIEN**	**Quadrature Encoder Interface Controller Enable Bit** 0 = QEI controller function Disabled. 1 = QEI controller function Enabled.
[28]	**CMPEN**	**the Compare Function Enable Bit** The compare function in QEI controller is to compare the dynamic counting QEI_CNT with the compare register QEI_CNTCMP, if QEI_CNT value reaches QEI_CNTCMP, the flag CMPF will be set. 0 = Compare function Disabled. 1 = Compare function Enabled.
[27]	**IDXRLDEN**	**Index Trigger QEI_CNT Reload Enable Bit** When this bit is high and a rising edge comes on signal CHX, the QEI_CNT will be reset to zero if the counter is in up-counting type (DIRSTS = 1); while the QEI_CNT will be reloaded with QEI_CNTMAX content if the counter is in down-counting type (DIRSTS = 0). 0 = Reload function Disabled. 1 = QEI_CNT re-initialized by Index signal Enabled.
[26]	**Reserved**	Reserved.
[25]	**IDXLATEN**	**Index Latch QEI_CNT Enable Bit** If this bit is set to high, the QEI_CNT content will be latched into QEI_CNTLATCH at every rising on signal CHX. 0 = The index signal latch QEI counter function Disabled. 1 = The index signal latch QEI counter function Enabled.
[24]	**HOLDCNT**	**Hold QEI_CNT Control** When this bit is set from low to high, the QEI_CNT value is copied into QEI_CNTHOLD. This bit may be set by writing 1 to it or Timer0~Timer3 interrupt flag TIF (TIMERx_INTSTS[0]). 0 = No operation. 1 = QEI_CNT content is captured and stored in QEI_CNTHOLD. **Note:** This bit is automatically cleared after QEI_CNTHOLD holds QEI_CNT value.

Bits	Description	
[23]	**HOLDTMR3**	**Hold QEI_CNT by Timer 3** 0 = TIF (TIMER3_INTSTS[0]) has no effect on HOLDCNT. 1 = A rising edge of bit TIF(TIMER3_INTSTS[0]) in timer 3 sets HOLDCNT to 1.
[22]	**HOLDTMR2**	**Hold QEI_CNT by Timer 2** 0 = TIF(TIMER2_INTSTS[0]) has no effect on HOLDCNT. 1 = A rising edge of bit TIF(TIMER2_INTSTS[0]) in timer 2 sets HOLDCNT to 1.
[21]	**HOLDTMR1**	**Hold QEI_CNT by Timer 1** 0 = TIF(TIMER1_INTSTS[0]) has no effect on HOLDCNT. 1 = A rising edge of bit TIF (TIMER1_INTSTS[0]) in timer 1 sets HOLDCNT to 1.
[20]	**HOLDTMR0**	**Hold QEI_CNT by Timer 0** 0 = TIF (TIMER0_INTSTS[0]) has no effect on HOLDCNT. 1 = A rising edge of bit TIF(TIMER0_INTSTS[0]) in timer 0 sets HOLDCNT to 1.
[19]	**IDXIEN**	**IDXF Trigger QEI Interrupt Enable Bit** 0 = The IDXF can trigger QEI interrupt Disabled. 1 = The IDXF can trigger QEI interrupt Enabled.
[18]	**CMPIEN**	**CMPF Trigger QEI Interrupt Enable Bit** 0 = CMPF can trigger QEI controller interrupt Disabled. 1 = CMPF can trigger QEI controller interrupt Enabled.
[17]	**DIRIEN**	**DIRCHGF Trigger QEI Interrupt Enable Bit** 0 = DIRCHGF can trigger QEI controller interrupt Disabled. 1 = DIRCHGF can trigger QEI controller interrupt Enabled.
[16]	**OVUNIEN**	**OVUNF Trigger QEI Interrupt Enable Bit** 0 = OVUNF can trigger QEI controller interrupt Disabled. 1 = OVUNF can trigger QEI controller interrupt Enabled.
[15]	**Reserved**	Reserved.
[14]	**IDXINV**	**Inverse IDX Input Polarity** 0 = Not inverse IDX input polarity. 1 = IDX input polarity is inversed to QEI controller.
[13]	**CHBINV**	**Inverse QEB Input Polarity** 0 = Not inverse QEB input polarity. 1 = QEB input polarity is inversed to QEI controller.
[12]	**CHAINV**	**Inverse QEA Input Polarity** 0 = Not inverse QEA input polarity. 1 = QEA input polarity is inversed to QEI controller.
[11:10]	**Reserved**	Reserved.
[9:8]	**MODE**	**QEI Counting Mode Selection** There are four quadrature encoder pulse counter operation modes. 00 = X4 Free-counting Mode. 01 = X2 Free-counting Mode. 10 = X4 Compare-counting Mode. 11 = X2 Compare-counting Mode.
[7]	**Reserved**	Reserved.

Bits	Description	
[6]	**IDXEN**	**IDX Input to QEI Controller Enable Bit** 0 = IDX input to QEI Controller Disabled. 1 = IDX input to QEI Controller Enabled.
[5]	**CHBEN**	**QEB Input to QEI Controller Enable Bit** 0 = QEB input to QEI Controller Disabled. 1 = QEB input to QEI Controller Enabled.
[4]	**CHAEN**	**QEA Input to QEI Controller Enable Bit** 0 = QEA input to QEI Controller Disabled. 1 = QEA input to QEI Controller Enabled.
[3]	**NFDIS**	**QEI Controller Input Noise Filter Disable Bit** 0 = The noise filter of QEI controller Enabled. 1 = The noise filter of QEI controller Disabled.
[2]	**Reserved**	Reserved.
[1:0]	**NFCLKSEL**	**Noise Filter Clock Pre-divide Selection** To determine the sampling frequency of the Noise Filter clock . 00 = QEI_CLK. 01 = QEI_CLK/2. 10 = QEI_CLK/4. 11 = QEI_CLK/16.

Quadrature Encoder Interface Status Register (QEI_STATUS)

Register	Offset	R/W	Description	Reset Value
QEI_STATUS	QEIn_BA+0x2C	R/W	QEI Controller 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							DIRSTS
7	6	5	4	3	2	1	0
Reserved				DIRCHGF	OVUNF	CMPF	IDXF

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**DIRSTS**	**QEI Counter Counting Direction Indication** 0 = QEI Counter is in down-counting. 1 = QEI Counter is in up-counting. **Note:** This bit is set/reset by hardware according to the phase detection between CHA and CHB.
[7:4]	**Reserved**	Reserved.
[3]	**DIRCHGF**	**Direction Change Flag** Flag is set by hardware while QEI counter counting direction is changed. Software can clear this bit by writing 1 to it. 0 = No change in QEI counter counting direction. 1 = QEI counter counting direction is changed. **Note:** This bit is only cleared by writing 1 to it.
[2]	**OVUNF**	**QEI Counter Overflow or Underflow Flag** Flag is set by hardware while QEI_CNT overflows from 0xFFFF_FFFF to zero in free-counting mode or from the QEI_CNTMAX value to zero in compare-counting mode. Similarly, the flag is set wile QEI counter underflows from zero to 0xFFFF_FFFF or QEI_CNTMAX. 0 = No overflow or underflow occurs in QEI counter. 1 = QEI counter occurs counting overflow or underflow. **Note:** This bit is only cleared by writing 1 to it.
[1]	**CMPF**	**Compare-match Flag** If the QEI compare function is enabled, the flag is set by hardware while QEI counter up or down counts and reach to the QEI_CNTCMP value. 0 = QEI counter does not match with QEI_CNTCMP value. 1 = QEI counter counts to the same as QEI_CNTCMP value. **Note:** This bit is only cleared by writing 1 to it.

Bits	Description	
[0]	**IDXF**	**IDX Detected Flag** When the QEI controller detects a rising edge on signal CHX it will set flag IDXF to high. 0 = No rising edge detected on signal CHX. 1 = A rising edge occurs on signal CHX. **Note:** This bit is only cleared by writing 1 to it.

6.11 Watchdog Timer (WDT)

6.11.1 Overview

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

6.11.2 Features

- 18-bit free running up counter forWDT time-out interval
- Selectable time out interval (2^4 ~ 2^{18}) and the time out interval is 104 ms ~ 26.3168 s (if WDT_CLK = 10 kHz and reset delay period selects (1024+2))
- System kept in reset state for a period of (1 / WDT_CLK) * 63
- Supports selectable WDTreset delay period, including 1026、130、18 or 3 WDT_CLK reset delay period
- Supports to force WDTenabled after chip powered on or reset by setting CWDTEN in Config0 register
- Supports WDTtime-out wake-up function only if WDT clock source is selected as 10 kHz

6.11.3 Block Diagram

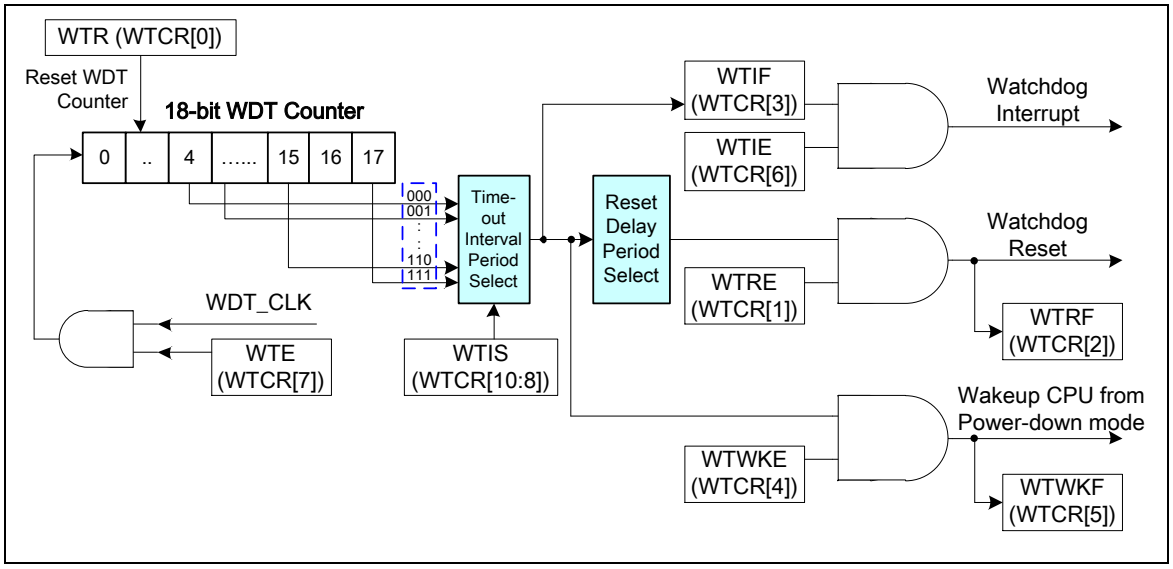


Figure 6–82 Watchdog Timer Block Diagram

Note1:WDT resets CPU and lasts 63 WDT_CLK.

Note2: Chip can be woken-up by WDT time-out interrupt signal generated only, if WDT clock source is selected to 10kHz oscillator.

Note3: The WDT reset delay periodcan be selected as 3/18/130/1026 WDT_CLK.

6.11.4 Clock Control

The WDT clock control are shown asFigure 6–83.

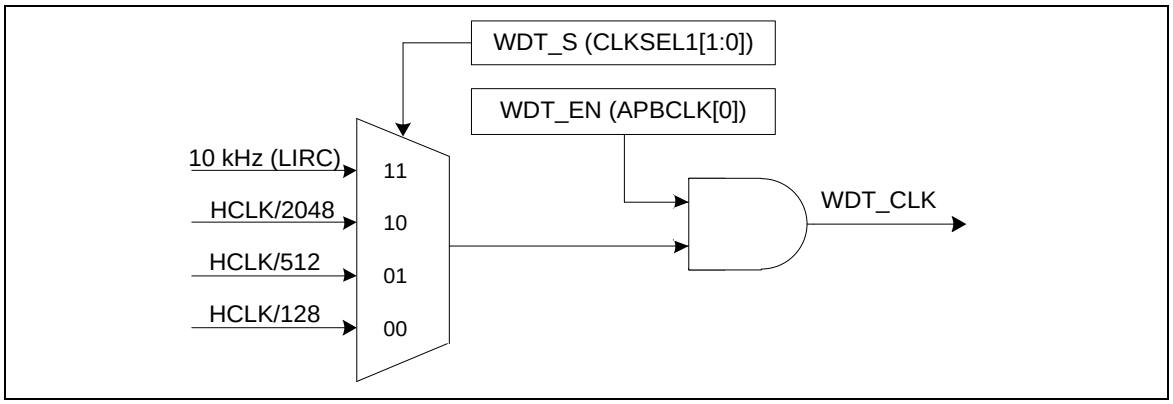


Figure 6–83 Watchdog Timer Clock Control

6.11.5 Basic Configuration

The WDT peripheral clock is enabled in WDT_EN (APBCLK[0]) and clock source can be selected in WDT_S (CLKSEL1[1:0]).

WDT controller also can be forced enabled and active in 10 kHz after chip powered on or reset while CWDTEN (Config0[31]) is configure to 0.

6.11.6 Functional Description

The WDT includes an 18-bit free running up counter with programmable time-out intervals.Table 6-13shows the WDT time-out intervalperiod selection and Figure 6-84 shows the WDT time-out interval and reset period timing.

6.11.6.1 WDT Time-out Interrupt

Setting WTE (WTCR[7]) to 1 will enable the WDT function and the WDT counter to start counting up. There are eight time-out interval period can be selected by setting WTIS (WTCR[10:8]). When the WDT up counter reaches the WTIS (WTCR[10:8])settings, WDT time-out interrupt will occur then WDT time-out interrupt flag WTIF (WTCR[3]) will be set to 1 immediately.

6.11.6.2 WDT Reset Delay Period and Reset System

A specified T_{RSTD} reset delay period occurs when the WTIF (WTCR[3]) is setto 1. User should setWTR (WTCR[0]) to reset the 18-bit WDT up counter value to avoid generate WDT time-out reset signal before the T_{RSTD}reset delay period expires.Moreover, user should set WTRDSEL (WTCRALT[1:0]) to select reset delay period to clear WDT counter. If the WDT up counter value has not been cleared after the specific T_{RSTD} delay period expires, the WDT control will set WTRF (WTCR[2]) to 1 if WTRE (WTCR[1]) bit is enabled, then chip enters to reset state immediately. Refer toFigure 6-84, T_{RST} reset period will keep last 63 WDT clocks then chip restart executing program from reset vector (0x0000_0000). The WTRF (WTCR[2]) will keep 1 after WDT time-out reset the chip, user can check WTRF (WTCR[2]) by software to recognize the system has been reset by WDT time-out reset or not.

WTIS	Time-Out Interval Period T_{TIS}	Reset Delay Period T_{RSTD}
000	$2^4 * T_{WDT}$	(3/18/130/1026) * T_{WDT}
001	$2^6 * T_{WDT}$	(3/18/130/1026) * T_{WDT}
010	$2^8 * T_{WDT}$	(3/18/130/1026) * T_{WDT}
011	$2^{10} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
100	$2^{12} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
101	$2^{14} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
110	$2^{16} * T_{WDT}$	(3/18/130/1026) * T_{WDT}
111	$2^{18} * T_{WDT}$	(3/18/130/1026) * T_{WDT}

Table 6-13Watchdog Timer Time-out Interval Period Selection

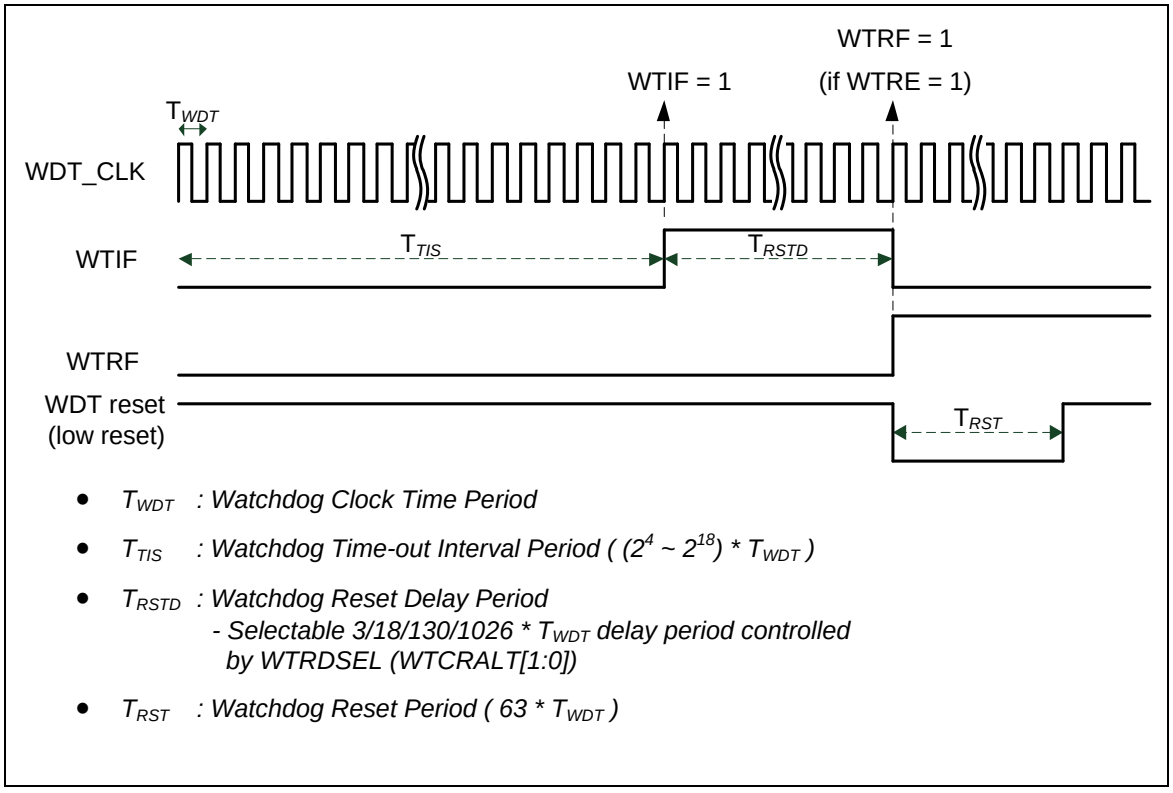


Figure 6-84Watchdog Timer Time-out Interval and Reset Period Timing

6.11.6.3 WDT Wake-up

If WDT clock source is selected to 10 kHz, system can be waken-up from Power-down mode while WDT time-out interrupt signal is generated and WTWKE (WTCR[4]) enabled. Notice that user should set OSC10K_EN (PWRCON[3])before system entries power down mode I system peripheral clock are disabled when system is power down mode. In the meanwhile, the WTWKF (WTCR[5]) will set to 1 automatically, user can check WTWKF(WTCR[5]) status by software to recognize the system has been waken-up by WDT time-out interrupt or not.

6.11.7 Register Map

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

Register	Offset	R/W	Description	Reset Value
WDT Base Address: **WDT_BA = 0x4000_4000**				
WTCR	WDT_BA+0x00	R/W	WDTControl Register	0x0000_0700
WTCRALT	WDT_BA+0x04	R/W	WDTAlternative Control Register	0x0000_0000

6.11.8 Register Description

WDT Control Register (WTCR)

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

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**

Bits	Description	
[31]	**DBGACK_WDT**	**ICE Debug Mode Acknowledge Disable (Write Protect)** 0 = ICE debug mode acknowledgement affects WDT counting. WDTup counter will be held while CPU is held by ICE. 1 = ICE debug mode acknowledgement Disabled. WDTup counter will keep going no matter CPU is held by ICE or not. **Note:** This bit is write protected. Refer to the REGWRPROT register.
[30:11]	**Reserved**	Reserved.
[10:8]	**WTIS**	**WDTTime-out Interval Selection(Write Protect)** These three bits select the time-out interval period for the WDT. 000 = 2^4 * WDT_CLK. 001 = 2^6 * WDT_CLK. 010 = 2^8 * WDT_CLK. 011 = 2^{10} * WDT_CLK. 100 = 2^{12} * WDT_CLK. 101 = 2^{14} * WDT_CLK. 110 = 2^{16} * WDT_CLK. 111 = 2^{18} * WDT_CLK. **Note:** This bit is write protected. Refer to the REGWRPROT register.
[7]	**WTE**	**WDT Enable Control (Write Protect)** 0 = WDT Disabled (This action will reset the internal up counter value). 1 = WDT Enabled. **Note1:** This bit is write protected. Refer to the REGWRPROT register. **Note2:** If CWDTEN(Config0[31]) bits is configure to 0, this bit is forced as 1 and user cannot change this bit to 0.
[6]	**WTIE**	**WDT Time-out Interrupt Enable Control (Write Protect)**

		If this bit is enabled, the WDT time-out interrupt signal is generated and inform to CPU. 0 = WDTtime-out interrupt Disabled. 1 = WDTtime-out interrupt Enabled. **Note:** This bit is write protected. Refer to the REGWRPROT register.
[5]	**WTWKF**	**WDT Time-out Wake-up Flag(Write Protect)** This bit indicates the interrupt wake-up flag status of WDT 0 = WDT does not cause chip wake-up. 1 = Chip wake-up from Idle or Power-down mode if WDT time-out interrupt signal generated. **Note1:** This bit is write protected. Refer to the REGWRPROT register. **Note2:** This bit is cleared by writing 1 to it.
[4]	**WTWKE**	**WDT Time-out Wake-up Function Control (Write Protect)** If this bit is set to 1, while WDT time-out interrupt flag IF (WTCR[3]) is generated to 1 and interrupt enable bit WTIE (WTCR[6]) is enabled, the WDT time-out interrupt signal will generate a wake-up trigger event to chip. 0 = Wake-up trigger event Disabled if WDT time-out interrupt signal generated. 1 = Wake-up trigger event Enabled if WDT time-out interrupt signal generated. **Note1:** This bit is write protected. Refer to the REGWRPROT register. **Note2:** Chip can be woken-up by WDT time-out interrupt signal generated only if WDT clock source is selected to 10 kHz internal low speed RC oscillator (LIRC).
[3]	**WTIF**	**WDT Time-out Interrupt Flag** This bit will set to 1 while WDT up counter value reaches the selected WDT time-out interval 0 = WDTtime-out interrupt did not occur. 1 = WDTtime-out interrupt occurred. **Note:** This bit is cleared by writing 1 to it.
[2]	**WTRF**	**WDTTime-out Reset Flag** This bit indicates the system has been reset by WDT time-out reset or not. 0 = WDT time-out reset did not occur. 1 = WDT time-out reset occurred. **Note:** This bit is cleared by writing 1 to it.
[1]	**WTRE**	**WDTTime-out Reset EnableControl (Write Protect)** Setting this bit will enable the WDT time-out reset function If the WDT up counter value has not been cleared after the specific WDT reset delay period expires. 0 = WDT time-out reset function Disabled. 1 = WDT time-out reset function Enabled. **Note:** This bit is write protected. Refer to the REGWRPROT register.
[0]	**WTR**	**ResetWDTUp Counter (Write Protect)** 0 = No effect. 1 = Reset the internal 18-bit WDT up counter value. **Note1:** This bit is write protected. Refer to the REGWRPROT register. **Note2:** This bit will be automatically cleared by hardware.

WDT Alternative Control Register (WTCRALT)

Register	Offset	R/W	Description	Reset Value
WTCRALT	WDT_BA+0x04	R/W	WDT AlternativeControl 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**	**WDT Reset Delay Selection (Write Protect)** When WDT time-out happened, user has a time named WDT Reset Delay Period to clear WDT counter by setting WTR (WTCR[0]) to prevent WDT time-out reset happened. User can select a suitable setting of WTRDSEL for different WDT Reset Delay Period. 00 = WDTReset Delay Period is 1026 * WDT_CLK. 01 = WDTReset Delay Period is 130 * WDT_CLK. 10 = WDTReset Delay Period is 18 * WDT_CLK. 11 = WDTReset Delay Period is 3 * WDT_CLK. **Note1:**This bit is write protected. Refer to the REGWRPROT register. **Note2:**This register will be reset to 0 if WDT time-out reset happened.

6.12 Window Watchdog Timer (WWDT)

6.12.1 Overview

The Window Watchdog Timer is used to perform a system reset within a specified window period to prevent software from running to uncontrollable state by any unpredictable condition usually generated by external interferences or unexpected logical conditions.

When the window function is used to trim the watchdog behavior to match the application perfectly, software must refresh the counter before time-out.

6.12.2 Features

- 6-bit down counter value WWDTCVAL (WWDTCVR[5:0]) and 6-bit compare value WINCMP (WWDTCR[21:16]) to make the WWDT time-out window period flexible
- Supports 4-bit value PERIODSEL (WWDTCR[11:8]) to programmable maximum 11-bit prescale counter period of WWDT counter
- WWDT counter suspends in Idle/Power-down mode

6.12.3 Block Diagram

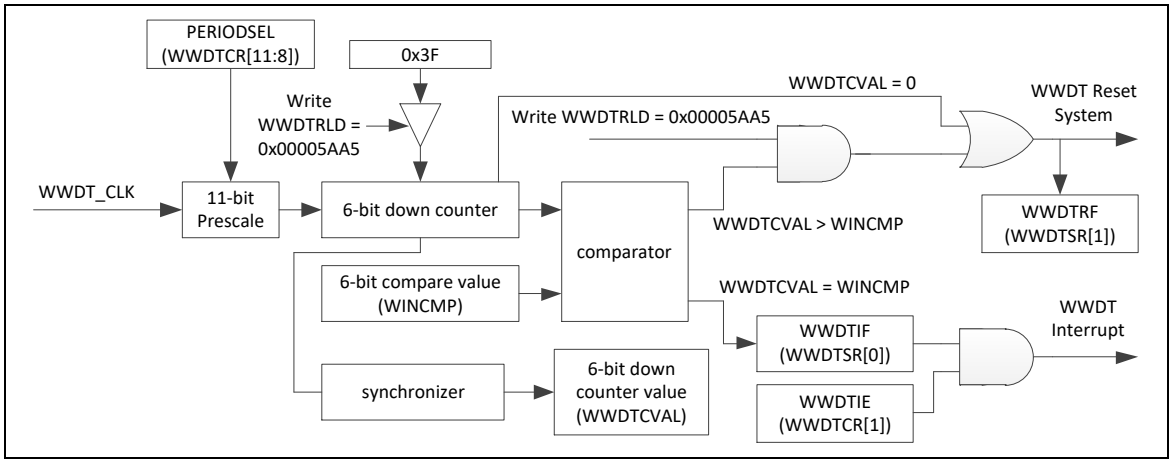


Figure 6-85 Window Watchdog Timer Block Diagram

6.12.4 Clock Diagram

The Window Watchdog Timer block diagram is shown as follows.

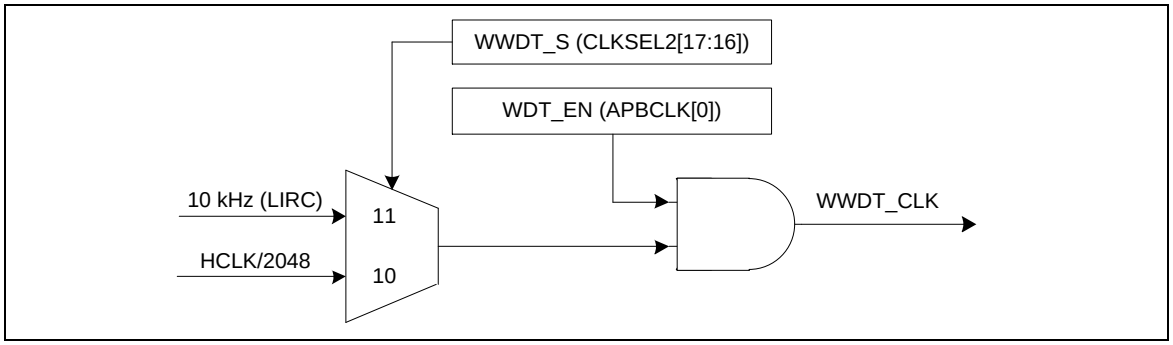


Figure 6-86 Window Watchdog Timer Clock Control

6.12.5 Basic Configuration

The WWDT peripheral clock is enabled in WDT_EN (APBCLK[0]) and clock source can be selected in WWDT_S (CLKSEL2[17:16]).

6.12.6 Functional Description

The WWDT includes a 6-bit down counter with programmable prescaler to define different time-out intervals.The clock source of 6-bit WWDT 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 6-14.

PERIODSEL	PrescalerValue	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 6-14WWDT Prescale Value Selection

6.12.6.1 WWDT Counting

When the WWDTEN (WWDTCR[0]) is set, WWDT down counter will start counting from 0x3F to 0. To prevent program runs to disable WWDT counter counting unexpected, the WWDTCRregister can only be written once after chip is powered on or reset. User cannot disable WWDT counter counting WWDTEN(WWDTCR[0]), change counter prescale period (PERIODSEL) or change window compare value WINCMP(WWDTCR[21:16]) while WWDTEN (WWDTCR[0])has been enabled by user unless chip is reset.

6.12.6.2 WWDT Compare Match Interrupt

During down counting by the WWDT counter, the WWDTIF (WWDTSR[0]) is set to 1 while the WWDT counter value WWDTCVAL (WWDTCVR[5:0]) is equal to window compare value WINCMP(WWDTCR[21:16])and WWDTIFcan be cleared by user; if WWDTIE (WWDTCR[1]) is also set to 1 by user, the WWDT compare match interrupt signal is generated also while WWDTIFis set to 1 by hardware.

6.12.6.3 WWDT Reset System

When WWDTIF (WWDTSR[0]) is generated, user must write 0x00005AA5 in WWDTRLDregister to reload WWDTCVAL(WWDTCVR[5:0])to 0x3F, and also to preventWWDT reset system signal occurred while WWDTCVAL reached to 0 and internal prescale counter value down count to 0.

If current WWDTCVAL is larger than WINCMP(WWDTCR[21:16]) and user writes 0x00005AA5 to the WWDTRLD register, the WWDT reset system signal will be generated immediately to cause chip reset also.

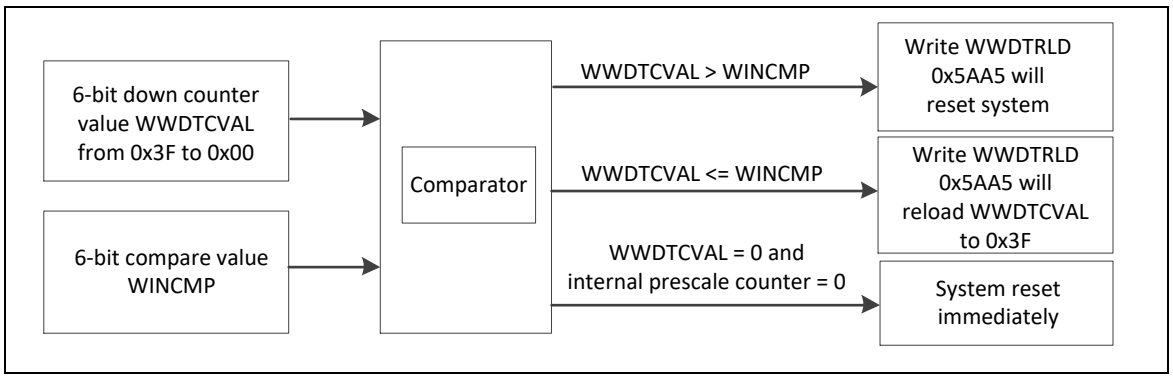


Figure 6-87 WWDTReset and Reload Behavior

6.12.6.4 WWDT Window Setting Limitation

When user writes 0x00005AA5 to WWDTRLDregister to reload WWDT counter value to 0x3F, it needs 3 WWDT clocks to sync the reload command to actually perform reload action.Notice that if user set PERIODSEL (WWDTCR[11:8]) to 0000, the counter prescale value should be as 1, and the WINCMP (WWDTCR[21:16]) must be larger than 2.Otherwise, writing WWDTRLDregister to reload WWDT counter value to 0x3F is unavailable,WWDTIF(WWDTSR[0]) is generated, and WWDT reset system event always happened.

PERIODSEL	Prescale Value	Valid WINCMP Value
0000	1	0x3 ~ 0x3F
0001	2	0x2 ~ 0x3F
Others	Others	0x0 ~ 0x3F

Table 6-15WWDT WINCMP Setting Limitation

6.12.7 Register Map

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

Register	Offset	R/W	Description	Reset Value
WWDT Base Address: **WWDT_BA = 0x4000_4100**				
WWDTRLD	WWDT_BA+0x00	W	WWDT Reload Counter Register	0x0000_0000
WWDTCR	WWDT_BA+0x04	R/W	WWDT Control Register	0x003F_0800
WWDTSR	WWDT_BA+0x08	R/W	WWDT Status Register	0x0000_0000
WWDTCVR	WWDT_BA+0x0C	R	WWDT Counter Value Register	0x0000_003F

6.12.8 Register Description

WWDT Reload Counter Register (WWDTRLD)

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

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

Bits	Description	
[31:0]	**WWDTRLD**	**WWDT Reload Counter Register** Writing 0x00005AA5 to this register will reload the WWDT counter value to 0x3F. **Note:** Usercan only write WWDTRLD to reload WWDT counter value when current WWDTcounter value between 0 and WINCMP(WWDTCR[21:16]). If user writes WWDTRLD when current WWDTcounter value is larger than WINCMP, WWDT reset signal will generate immediately.

WWDT Control Register (WWDTCR)

Register	Offset	R/W	Description	Reset Value
WWDTCR	WWDT_BA+0x04	R/W	WWDT Control Register	0x003F_0800

Note: This register can be written only one time 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 Control** 0 = ICE debug mode acknowledgement effects WWDT counting. WWDT down counter will be held while CPU is held by ICE. 1 = ICE debug mode acknowledgement Disabled. WWDT down counter will keep going no matter CPU is held by ICE or not.
[30:22]	**Reserved**	Reserved.
[21:16]	**WINCMP**	**WWDT Window Compare Register** Set this register to adjust the valid reload window. **Note:** User can only write WWDTRLD to reload WWDT counter value when current WWDT counter value between 0 and WINCMP. If user writes WWDTRLD when current WWDT counter value larger than WINCMP, WWDT reset signal will generate immediately.
[15:12]	**Reserved**	Reserved.
[11:8]	**PERIODSEL**	**WWDT Counter Prescale Period Selection** 0000 = Pre-scale is 1; Max time-out period is 1 * 64 * WWDT_CLK. 0001 = Pre-scale is 2; Max time-out period is 2 * 64 * WWDT_CLK. 0010 = Pre-scale is 4; Max time-out period is 4 * 64 * WWDT_CLK. 0011 = Pre-scale is 8; Max time-out period is 8 * 64 * WWDT_CLK. 0100 = Pre-scale is 16; Max time-out period is 16 * 64 * WWDT_CLK. 0101 = Pre-scale is 32; Max time-out period is 32 * 64 * WWDT_CLK. 0110 = Pre-scale is 64; Max time-out period is 64 * 64 * WWDT_CLK. 0111 = Pre-scale is 128; Max time-out period is 128 * 64 * WWDT_CLK. 1000 = Pre-scale is 192; Max time-out period is 192 * 64 * WWDT_CLK. 1001 = Pre-scale is 256; Max time-out period is 256 * 64 * WWDT_CLK. 1010 = Pre-scale is 384; Max time-out period is 384 * 64 * WWDT_CLK. 1011 = Pre-scale is 512; Max time-out period is 512 * 64 * WWDT_CLK. 1100 = Pre-scale is 768; Max time-out period is 768 * 64 * WWDT_CLK.

		1101 = Pre-scale is 1024; Max time-out period is 1024 * 64 * WWDT_CLK. 1110 = Pre-scale is 1536; Max time-out period is 1536 * 64 * WWDT_CLK. 1111 = Pre-scale is 2048; Max time-out period is 2048 * 64 * WWDT_CLK.
[7:2]	**Reserved**	Reserved.
[1]	**WWDTIE**	**WWDT Interrupt Enable Bit** If this bit is enabled, the WWDT counter compare match interrupt signal is generated and inform to CPU. 0 = WWDT counter compare match interrupt Disabled. 1 = WWDT counter compare match interrupt Enabled.
[0]	**WWDTEN**	**WWDT Enable Bit** Set this bit to enable WWDT counter counting 0 = WWDT counter is stopped. 1 = WWDT counter is starting counting.

WWDT Status Register (WWDTSR)

Register	Offset	R/W	Description	Reset Value
WWDTSR	WWDT_BA+0x08	R/W	WWDT Status Register	0x0000_0000

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

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1]	**WWDTRF**	**WWDT Time-out Reset Flag** This bit indicates the system has been reset by WWDT time-out reset or not. 0 = WWDT time-out reset did not occur. 1 = WWDT time-out reset occurred. **Note:**This bit is cleared by writing 1 to it.
[0]	**WWDTIF**	**WWDT Compare Match Interrupt Flag** This bit indicates the interrupt flag status of WWDT while WWDT counter value matches WINCMP(WWDTCR[21:16]). 0 = No effect. 1 = WWDT counter value matches WINCMP value. **Note:** This bit is cleared by writing 1 to it.

WWDT Counter Value Register (WWDTCVR)

Register	Offset	R/W	Description	Reset Value
WWDTCVR	WWDT_BA+0x0C	R	WWDT Counter Value Register	0x0000_003F

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

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5:0]	**WWDTCVAL**	**WWDT Counter Value** WWDTCVAL will be updated continuously to monitor 6-bit down counter value.

6.13 Universal Asynchronous Receiver Transmitter (UART)

6.13.1 Overview

TheNuMicro®NM1530 series providestwo channels of Universal Asynchronous Receiver/Transmitters (UART). UART Controller performs Normal Speed UARTand supports flow control function. The UART Controller performs a serial-to-parallel conversion on data received from the peripheraland a parallel-to-serial conversion on data transmitted from the CPU.Each UART Controller channel supports seventypes of interrupts. The UART controller also supports IrDA SIR,RS-485 and LIN.

6.13.2 Features

- Full duplex, asynchronous communications
- Separates receive / transmit 16 bytes entry FIFO for data payloads
- Supports hardware auto-flow control/flow control function (nCTS, nRTS) and programmable nRTS flow control trigger level
- Programmable receiver buffer trigger level
- Supports programmable baud-rate generator for each channel individually
- Supports nCTS wake-up function
- Supports 8-bit receiver buffer time out detection function
- Programmable transmitting data delay time between the last stop and the next start bit by setting DLY (UA_TOR [15:8]) register
- Supports break error, frame error, parity error and receive / transmit buffer overflow detect function
- Fully programmable serial-interface characteristics
  - Programmable data bit length, 5-, 6-, 7-, 8-bit character
  - Programmable parity bit, even, odd, no parity or stick parity bit generation and detection
  - Programmable stop bit length, 1, 1.5, or 2 stop bit generation
- IrDA SIR function mode
  - Supports 3-/16-bit duration for normal mode
- LIN function mode
  - Supports LIN master/slave mode
  - Supports programmable break generation function for transmitter
  - Supports break detect function for receiver
- RS-485 function mode
  - Supports RS-485 9-bit mode
  - Supports hardware or software direct enable control provided by nRTS pin

6.13.3 Block Diagram

The UART clock control and block diagram are shown in Figure 6-82 and Figure 6-89respectively.

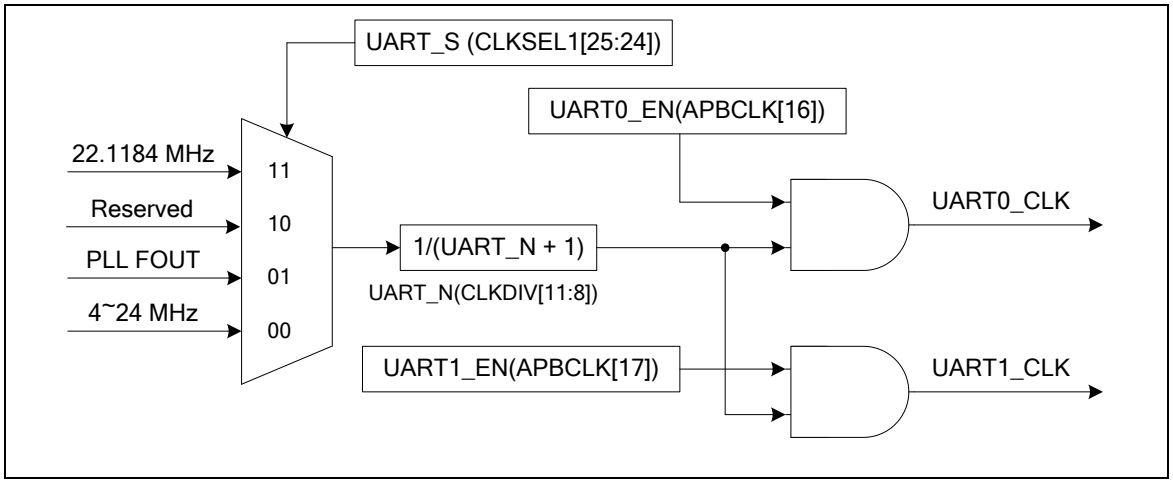


Figure 6-88UART Clock Control Diagram

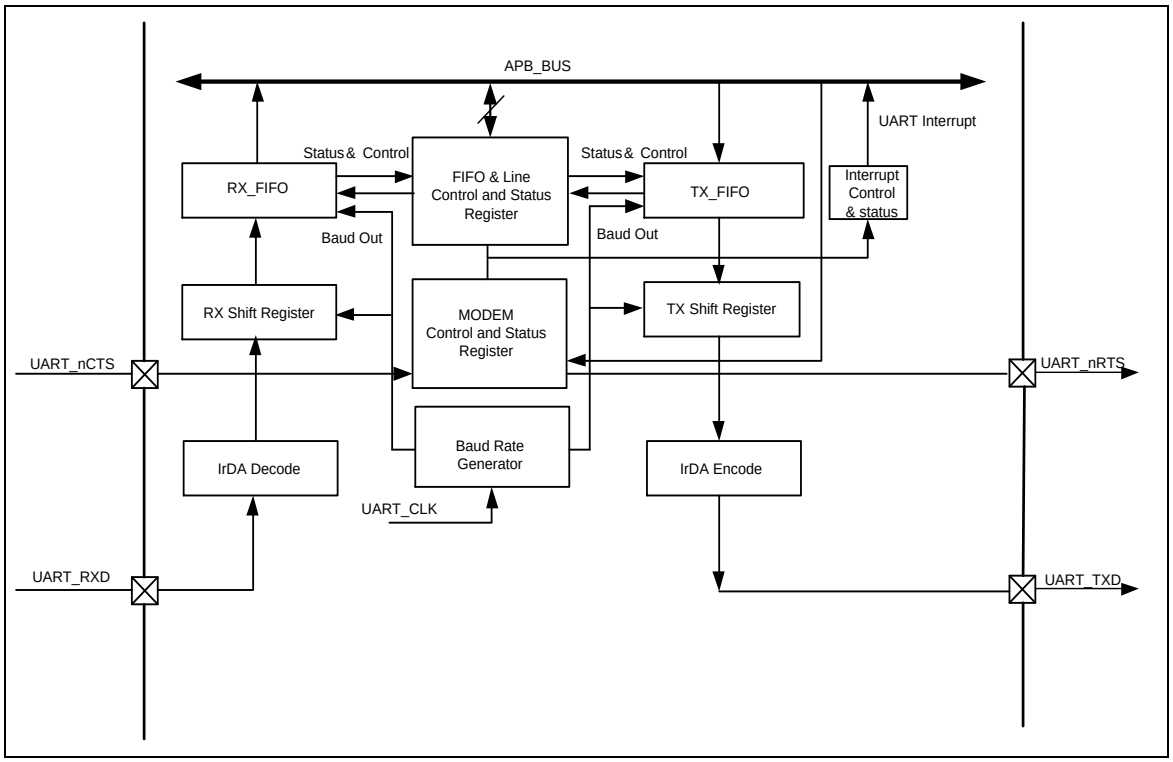


Figure 6-89 UART Block Diagram

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 BIF (UA_FSR[6]), FEF (UA_FSR[5]), PEF (UA_FSR[4])per byte) to reduce the number of interrupts presented to the CPU.

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

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

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

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

IrDA Encode
This block is IrDA encode control block.

IrDA Decode
This block is IrDA decode control block.

FIFO & Line Control and Status Register
This field is register set that including the FIFO Control Register (UA_FCR), FIFO Status Register (UA_FSR), and Line Control Register (UA_LCR) for transmitter and receiver. The Time OutRegister (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.

Interrupt Control and Status Register
There are seven types of interrupts, transmitter FIFO empty interrupt(THRE_INT), receiver threshold level reached interrupt (RDA_INT), line status interrupt (parity error or frame error or break error) (RLS_INT) , receiver buffer time-out interrupt (TOUT_INT), MODEM status interrupt (MODEM_INT), Buffer error interrupt (BUF_ERR_INT) and LIN bus interrupt (LIN_INT).

6.13.4 Basic Configuration

The basic configurations of UART0 are as follows:

- UART0 pins are configured in P3_MFP registers.
- Enable UART0 peripheral clock in UART0_EN (APBCLK[16]).
- Reset UART0 controller in UART0_RST (IPRSTC2[16]).

The basic configurations of UART1 are as follows:

- UART1 pins are configured in PA_MFP register.
- Enable UART1 peripheral clock in UART1_EN (APBCLK[17]).
- Reset UART1 controller in UART1_RST (IPRSTC2[17]).

6.13.5 Functional Description

The UART Controller supports four function modes including UART, IrDA, RS-485 and LIN mode. User can select a function by setting the UA_FUN_SEL register. The four function modes will be described in following section.

6.13.5.1 UART Controller Baud Rate Generator

The UART Controller includes a programmable baud rate generator capable of dividing clock input by divisors to produce the serial clock that transmitter and receiver need. The Table 6-16 and Table 6-17list the UART baud rate equations in the various conditionsand UART baud rate parameter settings. There is no error for the baud rate results calculated through the baud rate parameter and register setting below. In IrDA function mode, the baud rate generator must be set in mode 0. More detail register description is shown in UA_BAUD register. There are three setting mode. Mode 0 is set by UA_BAUD[29:28] with 00. Mode 1 is set by UA_BAUD[29:28] with 10. Mode 2 is set by UA_BAUD[29:28] with 11.

Mode	DIV_X_EN	DIV_X_ONE	Baud Rate Equation
Mode 0	0	0	UART_CLK / [16 * (BRD+2)]
Mode 1	1	0	UART_CLK / [(DIVIDER _X+1) * (BRD+2)], DIVIDER_X must >= 8
Mode 2	1	1	UART_CLK / (BRD+2), If UART_CLK <= HCLK, BRD must >=9. If HCLK < UART_CLK <= 2*HCLK, BRD must >=15. If 2*HCLK < UART_CLK <= 3*HCLK, BRD must >=21. If UART_CLK > 3*HCLK, it is unsupported.

Table 6-16Baud Rate Equation Table

UART PeripheralClock =22.1184 MHz			
Baud Rate	**Mode 0**	**Mode 1**	**Mode 2**
921600	Not support	BRD=0, DIVIDER_X=11	BRD=22
460800	BRD=1	BRD=1, DIVIDER_X =15 BRD=2, DIVIDER_X =11	BRD=46

230400	BRD =4	BRD =4, DIVIDER_X =15 BRD =6, DIVIDER_X =11	BRD =94
115200	BRD =10	BRD =10, DIVIDER_X =15 BRD =14, DIVIDER_X =11	BRD =190
57600	BRD =22	BRD =22, DIVIDER_X =15 BRD =30, DIVIDER_X =11	BRD =382
38400	BRD =34	BRD =62, DIVIDER_X =8 BRD =46, DIVIDER_X =11 BRD =34, E DIVIDER_X =15	BRD =574
19200	BRD =70	BRD =126, DIVIDER_X =8 BRD =94, DIVIDER_X =11 BRD =70, DIVIDER_X =15	BRD =1150
9600	BRD =142	BRD =254, DIVIDER_X =8 BRD =190, DIVIDER_X =11 BRD =142, DIVIDER_X =15	BRD =2302
4800	BRD =286	BRD =510, DIVIDER_X =8 BRD =382, DIVIDER_X =11 BRD =286, DIVIDER_X =15	BRD =4606

Table 6-17UART Controller Baud Rate Parameter SettingExample Table

UART Peripheral Clock =22.1184 MHz			
Baud Rate	**UA_BAUD Value**		
	Mode 0	**Mode 1**	**Mode 2**
921600	Not support	0x2B00_0000	0x3000_0016
460800	0x0000_0001	0x2F00_0001 0x2B00_0002	0x3000_002E
230400	0x0000_0004	0x2F00_0004 0x2B00_0006	0x3000_005E
115200	0x0000_000A	0x2F00_000A 0x2B00_000E	0x3000_00BE
57600	0x0000_0016	0x2F00_0016 0x2B00_001E	0x3000_017E
38400	0x0000_0022	0x2800_003E 0x2B00_002E 0x2F00_0022	0x3000_023E
19200	0x0000_0046	0x2800_007E 0x2B00_005E 0x2F00_0046	0x3000_047E
9600	0x0000_008E	0x2800_00FE 0x2B00_00BE 0x2F00_008E	0x3000_08FE
4800	0x0000_011E	0x2800_01FE 0x2B00_017E	0x3000_11FE

		0x2F00_011E	

Table 6-18UART Controller Baud Rate Register Setting Example Table

6.13.5.2 UART Controller Transmit Delay Time Value

The UART Controller programs DLY (UA_TOR[15:8]) to control the transfer delay time between the last stop bit and next start bit in transmission. The unit is baud.The operation is shown in below.

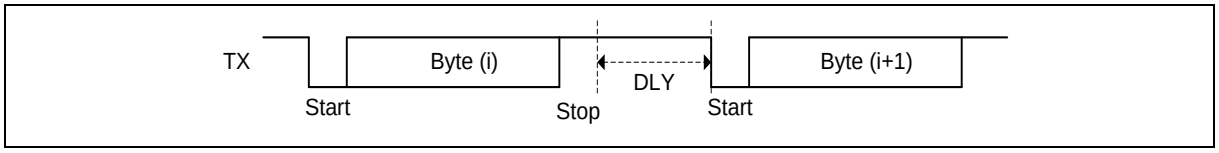


Figure 6-90Transmit Delay Time Operation

6.13.5.3 UART Controller FIFO Control and Status

The UART Controller is built-in with a 16 bytes transmitter FIFO (TX_FIFO) and a 16 bytes receiver FIFO (RX_FIFO) that reduces the number of interrupts presented to the CPU. The CPU can read the status of the UART at any time during operation. The reported status information includes condition of the transfer operations being performed by the UART, as well as 3 error conditions (parity error, frame error, break error)occur if receiving data has parity, frame or break error. UART, IrDA, LIN and RS-485 modesupportFIFO control and status function.

If there is any overrun event in transmitter or receiver FIFO, the buffer error flag BUF_ERR_IF (UA_ISR[5]) will be set automatically.

6.13.5.4 UART Controller Wake-up Function

The UART controller supports wake-up system function. The wake-up function includes nCTS. When the system is in Power-down, the UART can wake-up system by nCTS pin. The following diagram demonstrates the wake-up function.

nCTS Wake-up Case 1 (nCTS transition from low to high)

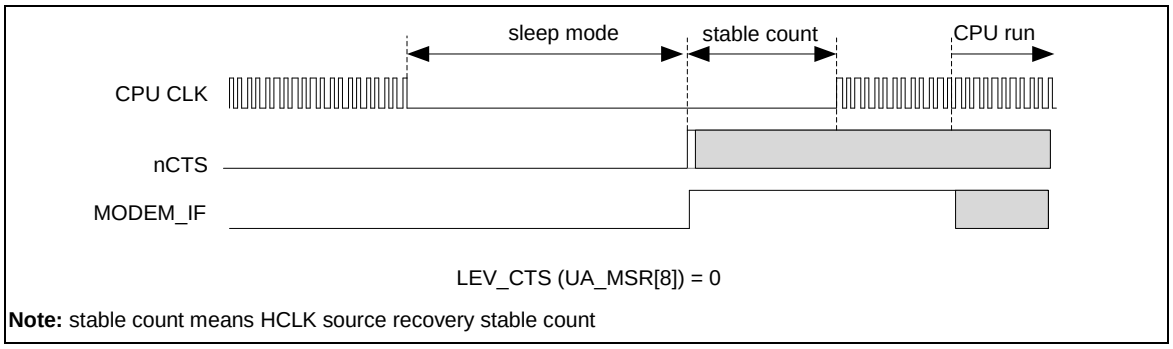
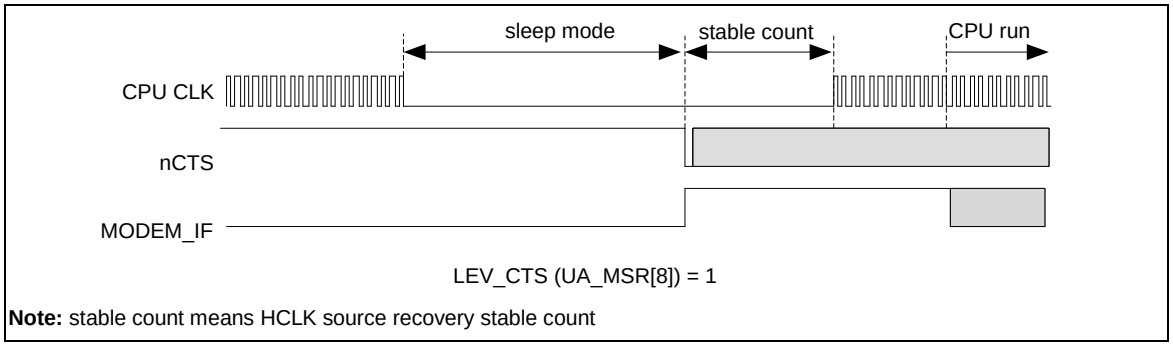


Figure 6-91 UART nCTS Wake-up Case1

nCTS Wake-up Case 2 (nCTS transition from high to low)



Note: stable count means HCLK source recovery stable count

Figure 6-92 UART nCTS Wake-up Case2

6.13.5.5 UART Controller Interrupt and Status

Each UART Controller supportsseven types of interrupts including:

- Receiver threshold level reached interrupt (RDA_INT)
- Transmitter FIFO empty interrupt (THRE_INT)
- Line status interrupt (parity error or frame error or break error) (RLS_INT)
- MODEM status interrupt (MODEM_INT)
- Receiver buffer time-out interrupt (TOUT_INT)
- Buffer error interrupt (BUF_ERR_INT)
- LIN bus interrupt (LIN_INT)

The Table 6-19describes the interrupt sources and flags. The interrupt is generated when the interrupt flag is generated and the interrupt enable bit is set. User must clear the interrupt flag after the interrupt is generated.

UART Interrupt Source	Interrupt Enable Bit	Interrupt Indicator To Interrupt Controller	Interrupt Flag	Flag Cleared By
LIN interrupt	LIN_IEN	LIN_INT	**LIN_IF**	Write ‘1’ to LINS_HDET_F/LIN_BKDET_F/BIT_ERR_F/LINS_IDPERR_F/LINS_HERR_F
Buffer Error Interrupt	BUF_ERR_IEN	BUF_ERR_INT	**BUF_ERR_IF** = TX_OVER_IF	Write ‘1’ to TX_OVER_IF
			BUF_ERR_IF = RX_OVER_IF	Write ‘1’ to RX_OVER_IF
Receiver Buffer Time-out Interrupt	TOUT_IEN	TOUT_INT	**TOUT_IF**	Read UA_RBR
Modem Status Interrupt	MODEM_IEN	MODEM_INT	**MODEM_IF** = DCTSF	Write ‘1’ to DCTSF
Receive Line Status Interrupt	RLS_IEN	RLS_INT	**RLS_IF** = BIF	Write ‘1’ to BIF
			RLS_IF = FEF	Write ‘1’ to FEF
			RLS_IF = PEF	Write ‘1’ toPEF
			RLS_IF = RS485_ADD_DETF	Write ‘1’ to RS485_ADD_DETF

Transmit Holding Register Empty Interrupt	THRE_IEN	THRE_INT	**THRE_IF**	Write UA_THR
Receive Data Available Interrupt	RDA_IEN	RDA_INT	**RDA_IF**	Read UA_RBR

Table 6-19 UART Interrupt Sources and Flags Table in Software Mode

6.13.5.6 UART Function Mode

The UART Controller provides UART function (SettingFUN_SEL (UA_FUN_SEL [1:0]) to 00 to enable UART function mode). The UART baud rate is up to 1 Mbps.

The UART provides full-duplex and asynchronous communications. The transmitter and receiver contain 16 bytes FIFO for payloads. User can program receiver buffer trigger level and receiver buffer time-out detection for receiver.

The UART supports hardware auto-flow control that provides programmable nRTS flow control trigger level. When number of data bytes in RX FIFO is equal to or greater than RTS_TRI_LEV (UA_FCR[19:16]), the nRTS is de-asserted.

UART Line Control Function

The UART Controller supportsfully programmable serial-interface characteristics by setting the UA_LCR register. User can program UA_LCR register for the word length, stop bit and parity bit setting. The Table 6-20 and Table 6-21list the UART word, stop bit length and the parity bit settings.

NSB (UA_LCR[2])	WLS (UA_LCR[1:0])	Word Length (Bit)	Stop Length (Bit)
0	00	5	1
0	01	6	1
0	10	7	1
0	11	8	1
1	00	5	1.5
1	01	6	2
1	10	7	2
1	11	8	2

Table 6-20UART Line Control of Word and Stop Length Setting

Parity Type	SPE (UA_LCR[5])	EPE (UA_LCR[4])	PBE (UA_LCR[3])	Description
No Parity	x	x	0	No parity bit output.
Odd Parity	0	0	1	Odd Parity is calculated by adding all the “1’s” in a data stream and adding a parity bit to the total bits, to make the total count an odd number.
Even Parity	0	1	1	Even Parity is calculated by adding all the “1’s” in a data stream and adding a parity bit to the total bits, to make the total count an even number.

Forced Mark Parity	1	0	1	Parity bit always logic 1. Parity bit on the serial byte is set to “1” regardless of total number of “1’s” (even or odd counts).
Forced Space Parity	1	1	1	Parity bit always logic 0. Parity bit on the serial byte is set to “0” regardless of total number of “1’s” (even or odd counts).

Table 6-21UART Line Control of Parity Bit Setting

UART Auto-Flow Control Function

The UART supports auto-flow control function that uses two signals, nCTS (clear-to-send) and nRTS (request-to-send), to control the flow of data transfer between the UART and external devices (e.g. Modem). When auto-flow is enabled, the UART is not allowed to receive data until the UART asserts nRTS to external device. When the number of bytes stored in the RX FIFO equals the value of RTS_TRI_LEV(UA_FCR[19:16]), the nRTS is de-asserted. The UART sends data out when UART detects nCTS is asserted from external device. If the valid asserted nCTS is not detected, the UART will not send data out.

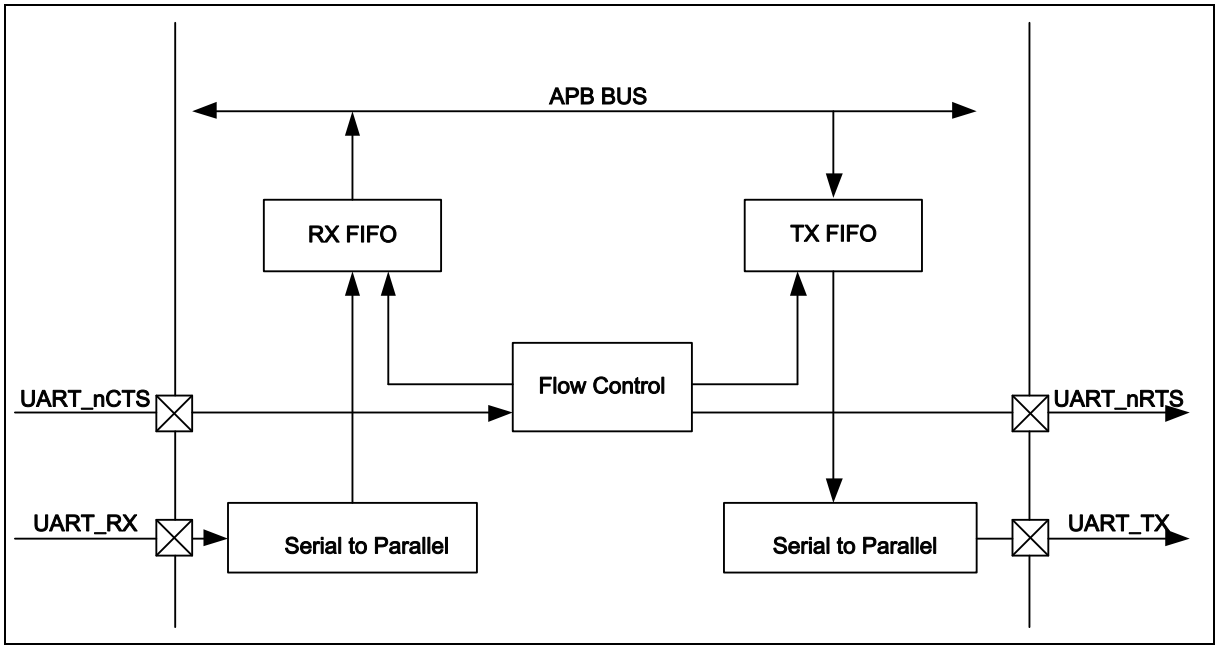


Figure 6-93Auto-Flow Control Block Diagram

The following diagram demonstrates the nCTS auto-flow control of UART function mode. User must set AUTO_CTS_EN(UA_IER[13]) to enable nCTS auto-flow control function. The LEV_CTS (UA_MSR[8]) can set nCTS pin input active state. The DCTSF (UA_MSR[0]) is set when any state change of nCTS pin input has occurred, and then TX data will be automatically transmitted from TX FIFO.

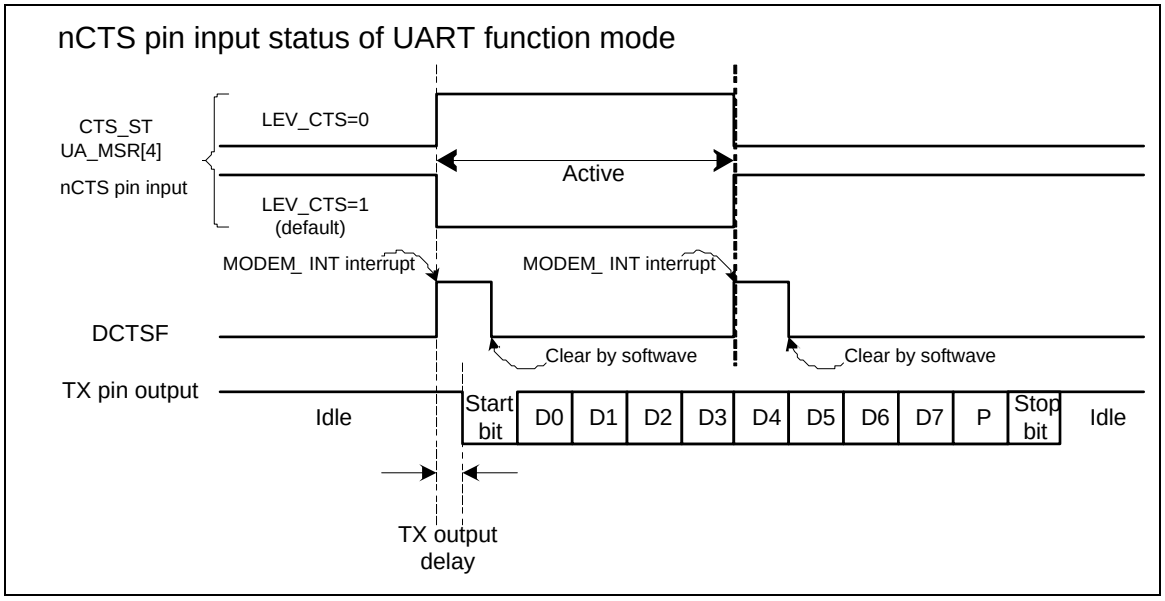


Figure 6-94UART nCTS Auto-Flow Control Enabled

As shown in theFigure 6-95, in UART nRTS auto-flow control mode (AUTO_RTS_EN(UA_IER[12])=1), the nRTS internal signal is controlled by UA_FCR controller with RTS_TRI_LEV(UA_FCR[19:16]) trigger level.

Setting LEV_RTS(UA_MCR[9]) can control the nRTS pin output is inverse or non-inversefrom nRTS signal. User can read the RTS_ST(UA_MCR[13]) bit to get real nRTS pin output voltage logic status.

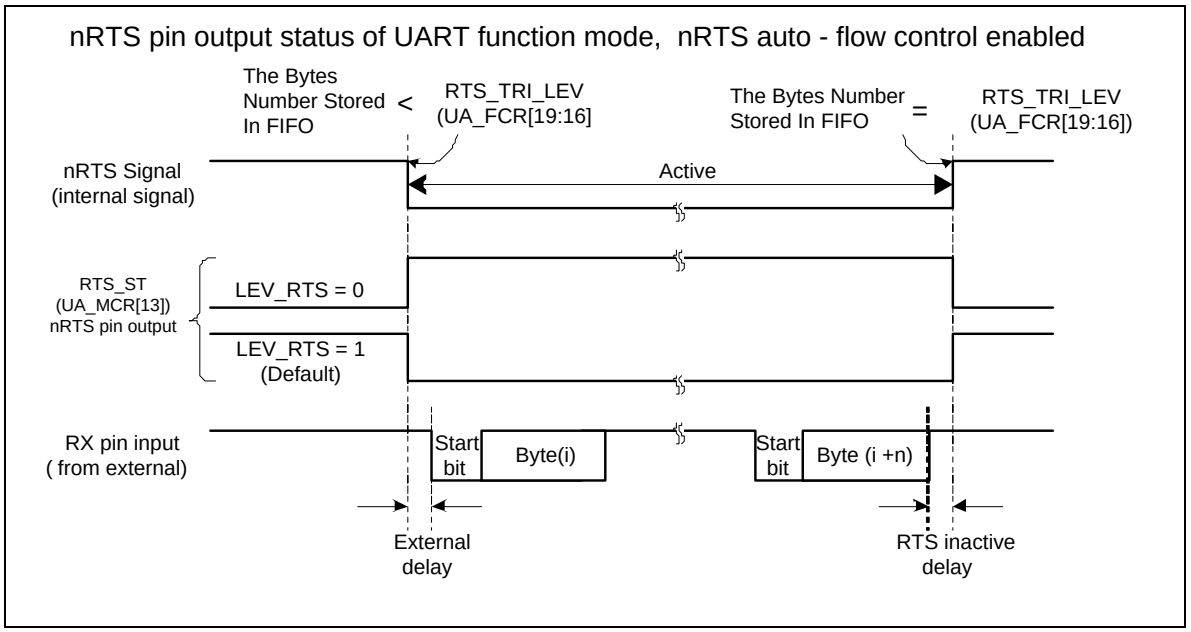


Figure 6-95UART nRTS Auto-Flow Control Enabled

As shown in theFigure 6-96, in software mode (AUTO_RTS_EN (UA_IER[12])=0), the nRTS flow is directly controlled by software programming of RTS(UA_MCR[1]) control bit.

Setting LEV_RTS(UA_MCR[9]) can control the nRTS pin output is inverse or non-inverse from RTS(UA_MCR[1]) control bit. User can read the RTS_ST(UA_MCR[13]) bit to get real nRTS pin output voltage logic status.

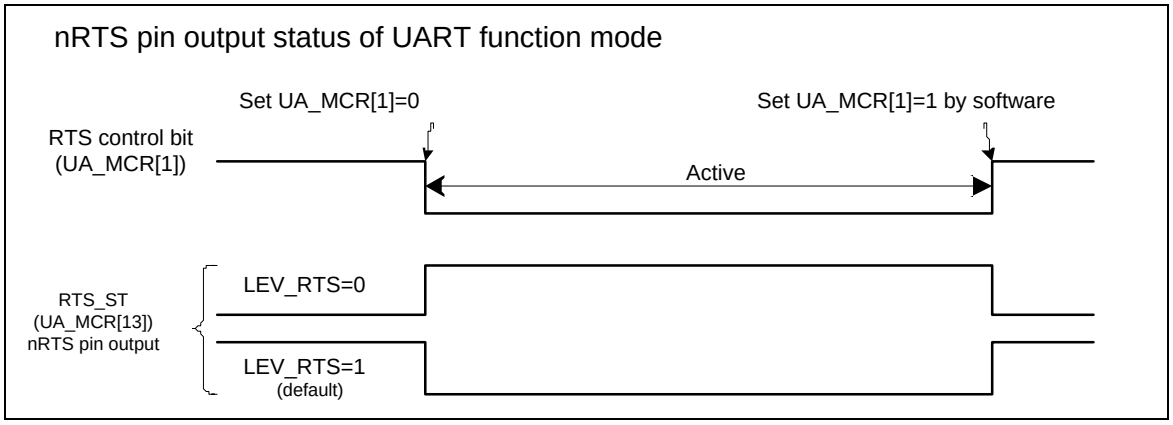


Figure 6-96UART nRTS Auto-Flow with Software Control

6.13.5.7 IrDA Function Mode

The UART Controller also provides Serial IrDA (SIR, Serial Infrared) function (SettingFUN_SEL (UA_FUN_SEL[1:0]) to 10 to enable the IrDA function). The SIR specification defines a short-range infrared asynchronous serial transmission mode with one start bit, 8 data bits, and 1 stop bit. The maximum data rate is 115.2 kbps. The IrDA SIR block contains an IrDA SIR protocol encoder/decoder. The IrDA SIR protocol is half-duplex only. So, it cannot transmit and receive data at the same time. The IrDA SIR physical layer specifies a minimum 10 ms transfer delay between transmission and reception, and this delay feature must be implemented by software.

In IrDA mode, the DIV_X_EN (UA_BAUD[29]) register must be cleared.

Baud Rate = Clock / (16 * (BRD+2)), where BRD is Baud Rate Divider in UA_BAUD register.

The IrDA control block diagram is shown as follows

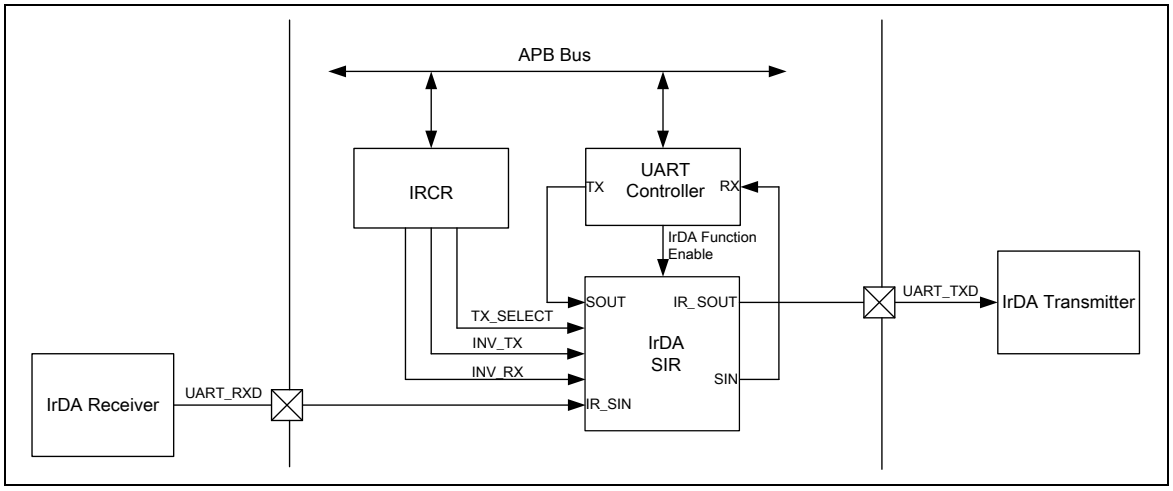


Figure 6-97 IrDA Block Diagram

6.13.5.7.1 IrDA SIR Transmit Encoder

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

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

6.13.5.7.2 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 and the start bit is detected when the decoder input is LOW. (Because of this,INV_RX(UA_IRCR[6]) should be set as 1 by default and the INV_TX (UA_IRCR[5]) is set to ‘0’).

6.13.5.7.3 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:

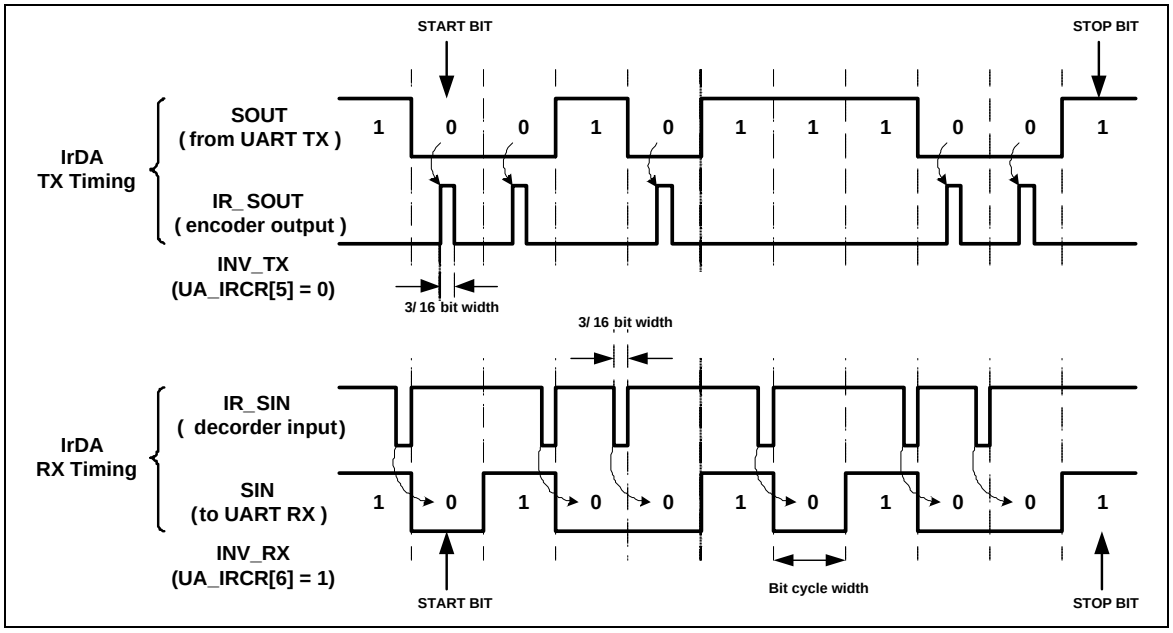


Figure 6-98 IrDA Timing Diagram

6.13.5.8 RS-485 function mode

Another alternate function of UART Controller is RS-485 function(user must set FUN_SEL (UA_FUN_SEL[1:0]) to 11 to enable RS-485 function), and direction control provided by nRTS pin from an asynchronous serial port. The RS-485 transceiver control is implemented by using the nRTS control signal to enable the RS-485 driver, many characteristics of the RX and TX are same as UART in RS-485 mode.

The UART controller can be configured as an RS-485 addressable slave and the RS-485 master transmitter will identify an address character by setting the parity (9-th bit) to 1. For data characters, the parity is set to 0. Software can use UA_LCR register to control the 9-th bit (When the PBE (UA_LCR[3]), EPE (UA_LCR[4]) and SPE(UA_LCR[5]) are set, the 9-th bit is transmitted 0 and when PBE (UA_LCR[3]) and SPE(UA_LCR[5]) are set and EPE (UA_LCR[4]) is cleared, the 9-th bit is transmitted 1).

The controller supports three operation modes: RS-485 Normal Multidrop Operation Mode (NMM), RS-485 Auto Address Detection Operation Mode (AAD) and RS-485 Auto Direction Control Operation Mode (AUD). Software can choose any operation mode by programming the UA_ALT_CSR register, and drive the transfer delay time between the last stop bit leaving the TX-FIFO and the de-assertion of by setting DLY (UA_TOR[15:8]) register.

The Controller support three operation mode that is

RS-485 Normal Multidrop Operation Mode (NMM)

In RS-485 Normal Multidrop operation mode(RS485_NMM (UA_ALT_CSR[8]) = 1), in first, software must decide the data which before the address byte be detected will be stored in RX-FIFO or not. If software want to ignore any data before address byte detected, the flow is set RX_DIS (UA_FCR[8]), then enable RS485_NMM (UA_ALT_CSR[8])and the receiver will ignore any data until an address byte is detected (bit9 =1) and the address byte data will be stored in the RX-FIFO. If software wants to receive any data before address byte detected, the flow is clearRX_DIS (UA_FCR[8]), then enables RS485_NMM (UA_ALT_CSR[8]) and the receiver will received any data.

If an address byte is detected (bit9 =1), it will generate an interrupt to CPU and RX_DIS (UA_FCR[8]) can decide whether accepting the following data bytes are stored in the RX-FIFO. If software disables receiver by setting RX_DIS (UA_FCR[8])register, when the next address byte be detected, the controller will clear the RX_DIS (UA_FCR[8]) bit and the address byte data will be stored in the RX-FIFO.

RS-485 Auto Address Detection Operation Mode (AAD)

In RS-485 Auto Address Detection Operation mode(RS485_AAD (UA_ALT_CSR[9]) = 1), the receiver will ignore any data until an address byte is detected (bit9 =1) and the address byte data match the ADDR_MATCH (UA_ALT_CSR [31:24])value. The address byte data will be stored in the RX-FIFO. The all received byte data will be accepted and stored in the RX-FIFO until an address byte data not match the ADDR_MATCH (UA_ALT_CSR[31:24])value.

RS-485 Auto Direction Mode (AUD)

Another option function of RS-485 controllers is RS-485 auto direction control function(RS485_AUD (UA_ALT_CSR[10] = 1). The RS-485 transceiver control is implemented by using the nRTS control signal from an asynchronous serial port. The nRTS line is connected to the RS-485 transceiver enable pin such that setting the nRTS line to high (logic 1) enables the RS-485 transceiver. Setting the nRTS line to low (logic 0) puts the transceiver into the tri-state condition to disabled. User can setting LEV_RTS (UA_MCR[9]) to change the nRTS driving level.

The following diagram demonstrates the RS-485 nRTS driving level in AUD mode. The nRTS pin will be automatically driven during TX data transmission.

Setting LEV_RTS(UA_MCR[9]) can control nRTS pin output driving level. User can read the

RTS_ST(UA_MCR[13]) bit to get real nRTS pin output voltage logic status.

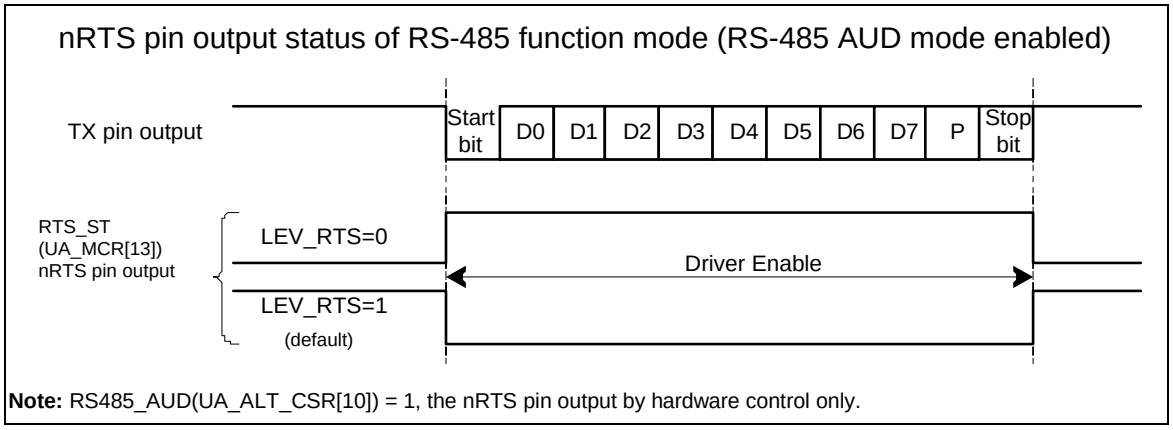


Figure 6-99RS-485 nRTS Driving Level in Auto Direction Mode

The following demonstrates the RS-485 nRTS driving level in software control (RS485_AUD(UA_ALT_CSR[10])=0). The nRTS driving level is controlled by programing the RTS(UA_MCR[1]) control bit.

Setting LEV_RTS(UA_MCR[9]) can control the nRTS pin output is inverse or non-inverse from RTS(UA_MCR[1]) control bit. User can read the RTS_ST(UA_MCR[13]) bit to get real nRTS pin output voltage logic status.

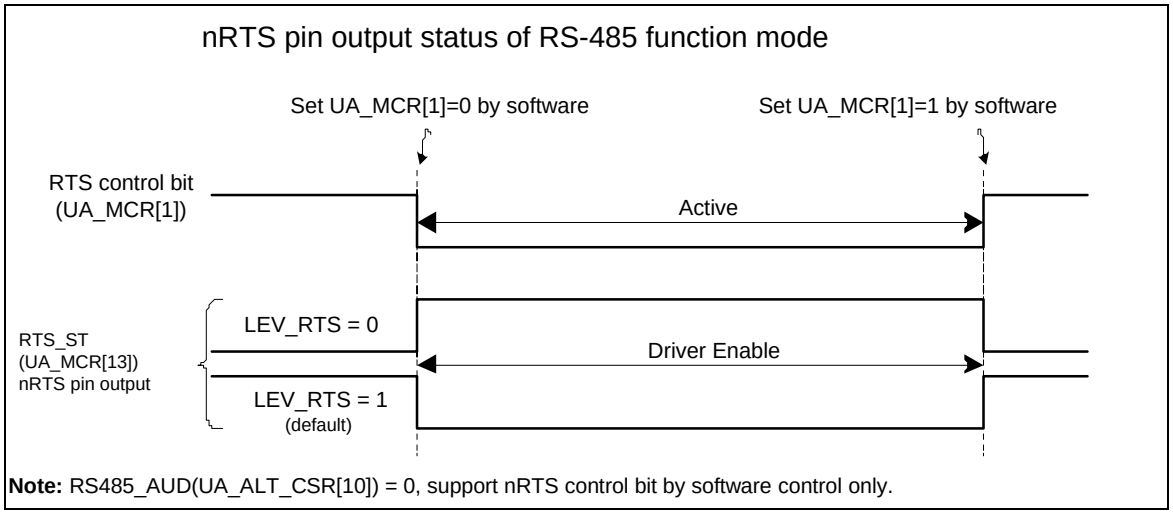
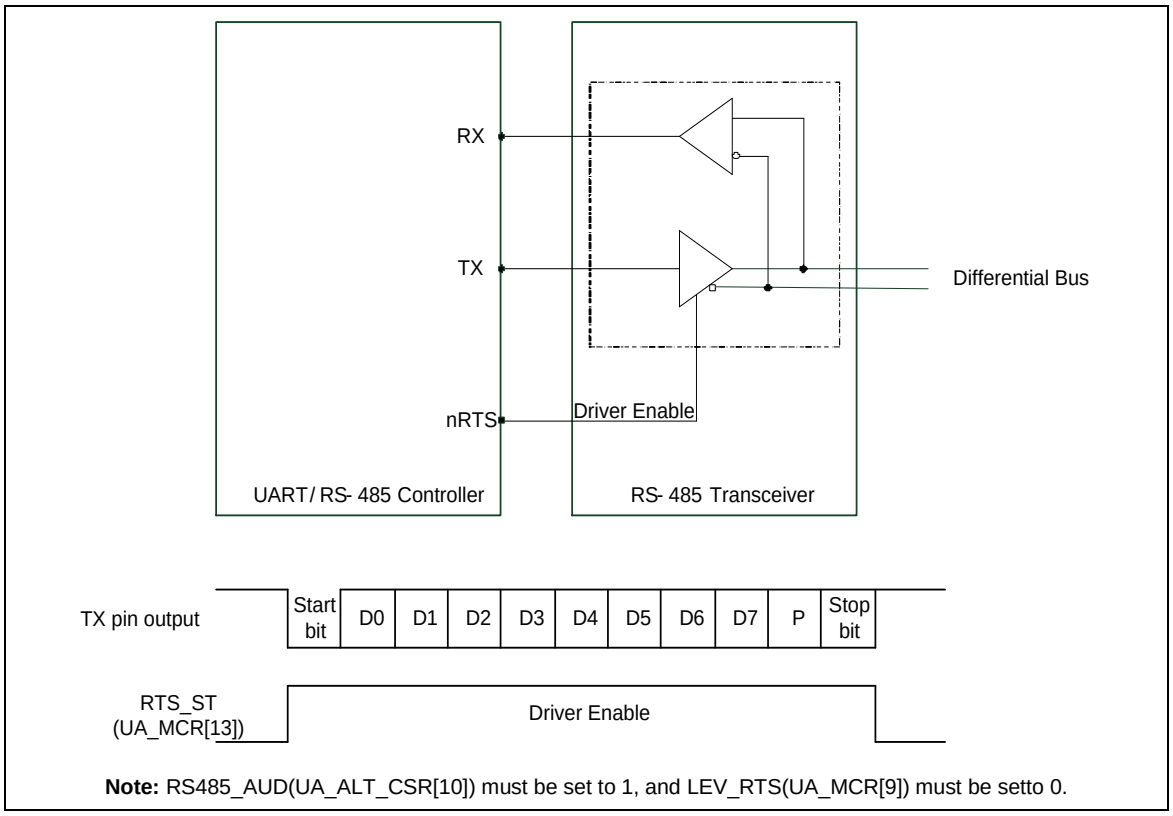


Figure 6-100RS-485 nRTS Driving Level with Software Control

Program Sequence Example:

1. Program FUN_SEL in UA_FUN_SEL to select RS-485 function.
2. Program the RX_DIS (UA_FCR[8])to determine enable or disable RS-485 receiver.
3. Program the RS485_NMM (UA_ALT_CSR[8])or RS485_AAD (UA_ALT_CSR[9])mode.
4. If the RS485_AAD (UA_ALT_CSR[9])mode is selected, the ADDR_MATCH (UA_ALT_CSR[31:24])is programmed for auto address match value.
5. Determine auto direction control by programming RS485_AUD(UA_ALT_CSR[10]).



Note: RS485_AUD(UA_ALT_CSR[10]) must be set to 1, and LEV_RTS(UA_MCR[9]) must be setto 0.

Figure 6-101 Structure of RS-485 Frame

6.13.5.9 LIN (Local Interconnection Network) mode

The UART supports LIN function, and LIN mode is selected by setting the FUN_SEL (UA_FUN_SEL[1:0]) to 01. The UART support LIN break/delimiter generation and break/delimiter detection in LIN master mode, support header detection and automatic resynchronization in LIN slave mode.

6.13.5.9.1 Structure of LIN Frame

According to the LIN protocol, all information is transmitted packed as frames; a frame consist a header(provided by the master task) and a response (provided by a slave task). That is any communication on the LIN bus is started by the master sending a header, followed by the response. The header (provided by the master task) consists of a break field and sync field followed by a frame identifier (frame ID). The frame identifier uniquely defines the purpose of the frame. The slave task appointed for providing the response associated with the frame ID and the response consists of a data field and a checksum field. The following diagram is the structure of LIN Frame.

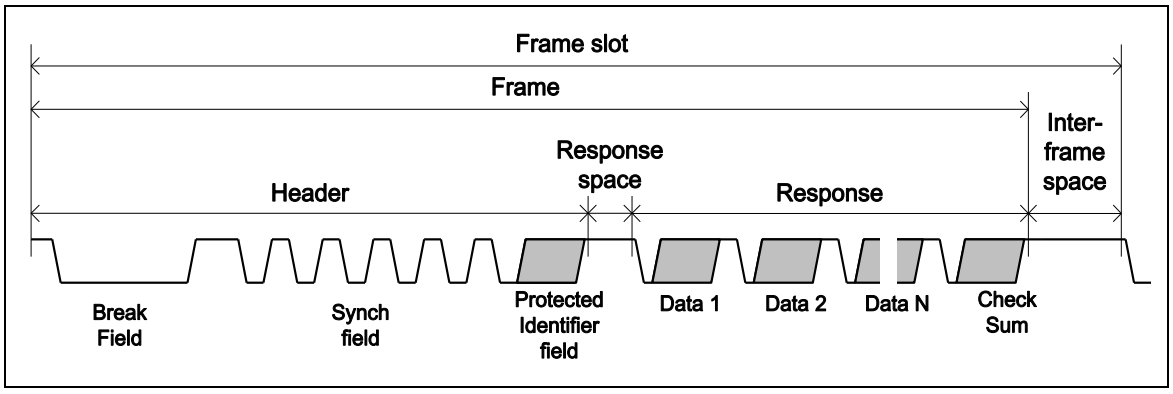


Figure 6-102 Structure of LIN Frame

6.13.5.9.2 Structure of LIN Byte

In LIN mode, each byte field is initiated by a START bit with value zero (dominant), followed by 8 data bits WLS (UA_LCR[1:0]) = 11 and no parity bit, LSB is first and ended by 1 stop bit NSB (UA_LCR[2]) = 0 with value one (recessive) in accordance with the LIN standard. Structure of Byte is shown as follows.

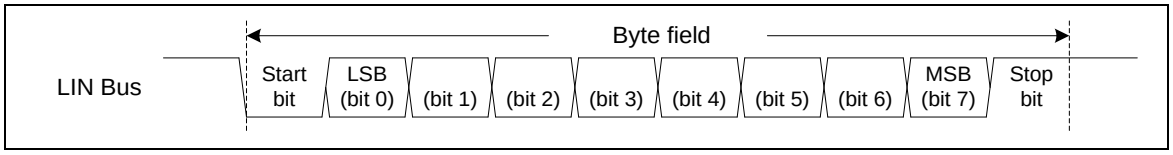


Figure 6-103 Structure of LIN Byte

6.13.5.9.3 LIN Master Mode

The UART controller support LIN master mode by setting the UA_FUN_SEL register. To enable and initialize the LIN master mode, the following steps are necessary.

1. Select the desired baud-rate by setting the UA_BAUD register.
2. Configure the data length to 8 bits by setting WLS (UA_LCR[1:0]) = 11 and disable parity check by clearing PBE (UA_LCR[3]) bit and configure the stop bit to 1 by clearing NSB (UA_LCR[2]) bit.

3. Select LIN function mode by setting UA_FUN_SEL register.

A complete header consists of a break field and sync field followed by a frame identifier (frame ID). The UART controller can be selected header sending by three header selected mode. The header selected mode can be “break field” or “break field and sync field” or “break field, sync field and frame ID field” by setting LIN_HEAD_SEL(UA_LIN_CTL[23:22]) in UA_LIN_CTL register. If the header selected is “break field”, software must handle the following sequence to sending a complete header to bus by filled sync data (0x55) and frame ID data to UA_THR register. If the header selected is “break field and sync field”, software must handle the sequence to sending a complete header to bus by filled frame ID data to UA_THR register, and if the header selected is “break field, sync field and frame ID field”, hardware will control the header sending sequence automatically but software must filled frame ID data to LIN_PID(UA_LIN_CTL[31:24]) register. When operating in header selected is “break field, sync field and frame ID field” mode, the frame ID parity bit can be calculated by software or hardware that depended on LIN_IDPEN (UA_LIN_CTL[9]) bit setting or not.

LIN_HEAD_SEL	Break Field	Sync Field	ID Field
0	Generated by H/W	Handled by S/W	Handled by S/W
1	Generated by H/W	Generated by H/W	Handled by S/W
2	Generated by H/W	Generated by H/W	Generated by H/W (But S/W needs to fill ID to LIN_PID (UA_LIN_CTL[31:24]) first)

Table 6-22 LIN Header selection in master mode

When operating in LIN data transmission, software can monitor the LIN bus transfer state by hardware or software. User can enable hardware monitoring by setting BIT_ERR_EN (UA_LIN_CTL[12]) to “1”, if the input pin (SIN) state is not equal to the output pin (SOUT) state in LIN transmitter state that the hardware will generator an interrupt to CPU. Software also can monitor the LIN bus transfer state by checking the read back data in UA_RBR register. The following sequence is a program sequence example:

Procedure without software error monitoring in master mode

1. Fill Protected Identifier toLIN_PID (UA_LIN_CTL[31:24]).
2. Choose the hardware transmission header field include “break field + sync field + Protected identifier field” by setting LIN_HEAD_SEL (UA_LIN_CTL[23:22]) = 10.
3. Request header transmission by setting the LIN_SHD (UA_LIN_CTL[8]).
4. Wait until LIN_SHD (UA_LIN_CTL[8]) be cleared by hardware.
5. Wait until TE_FLAG (UA_FSR[28]) set to “1” by hardware.

Note1: The break field + break/sync delimiter default setting is 13 dominant bits(break field) and 1 recessive bit(break/sync delimiter), software can change it by setting LIN_BKFL(UA_LIN_CTL[19:16]) and LIN_BS_LEN(UA_LIN_CTL[21:20]), to change the dominant bits.

Note2: The break/sync delimiter length default setting is 1 bit time and the inter-byte spaces default setting is also 1 bit time, software can change them by

settingLIN_BS_LEN(UA_LIN_CTL[21:20]) and DLY (UA_TOR[15:8]).

Note3: If the header includes “break field, sync field and frame ID field”, software must fill frame ID in LIN_PID (UA_LIN_CTL[31:24]) register before trigger header transmission (setting the LIN_SHD (UA_LIN_CTL[8])). The frame ID parity can be generated by software or hardware depends on LIN_IDPEN(UA_LIN_CTL[9]). If the parity generated by software (LIN_IDPEN(UA_LIN_CTL[9]) = 0), software must fill 8 bit data (include 2 bit parity) in this field, and if the parity generated by hardware (LIN_IDPEN(UA_LIN_CTL[9]) = 1), software fill ID0~ID5, hardware will calculate P0 and P1.

Procedure with software error monitoring in master mode

1. Choose the hardware transmission header field only include “break field” by setting LIN_HEAD_SEL(UA_LIN_CTL[23:22]) = 0x00.
2. Enable break detection function by setting LIN_BKDET_EN (UA_LIN_CTL[10]).
3. Request break + break/sync delimiter transmission by setting the LIN_SHD(UA_LIN_CTL[8]).
4. Wait until the LIN_BKDET_F (UA_LIN_SR[8]) be set to “1” by hardware.
5. Request sync field transmission by writing 0x55 into UA_THR register.
6. Wait until the RDA_IF (UA_ISR[0]) set to “1” by hardware and then read back the UA_RBR register.
7. Request header frame ID transmission by writing the protected identifier value to UA_THR register.
8. Wait until the RDA_IF (UA_ISR[0]) set to “1” by hardware and then read back the UA_RBR register.

LIN break and delimiter detection

When software enable the break detection function by setting LIN_BKDET_EN (UA_LIN_CTL[10]), the break detection circuit is activated. The break detection circuit is totally independent from the UART receiver.

When enable break detection function, the circuit looks at the input SIN pin for a start signal. If circuit detected consecutive dominant greater than 11 bits dominant followed by a recessive bit(delimiter), the LIN_BKDET_F (UA_LIN_SR[8]) flag is set at the end of break field. If the LIN_IEN (UA_IER[8]) =1, an interrupt will be generated. The behavior of the break detection and break flag is shown inFigure 6-104.

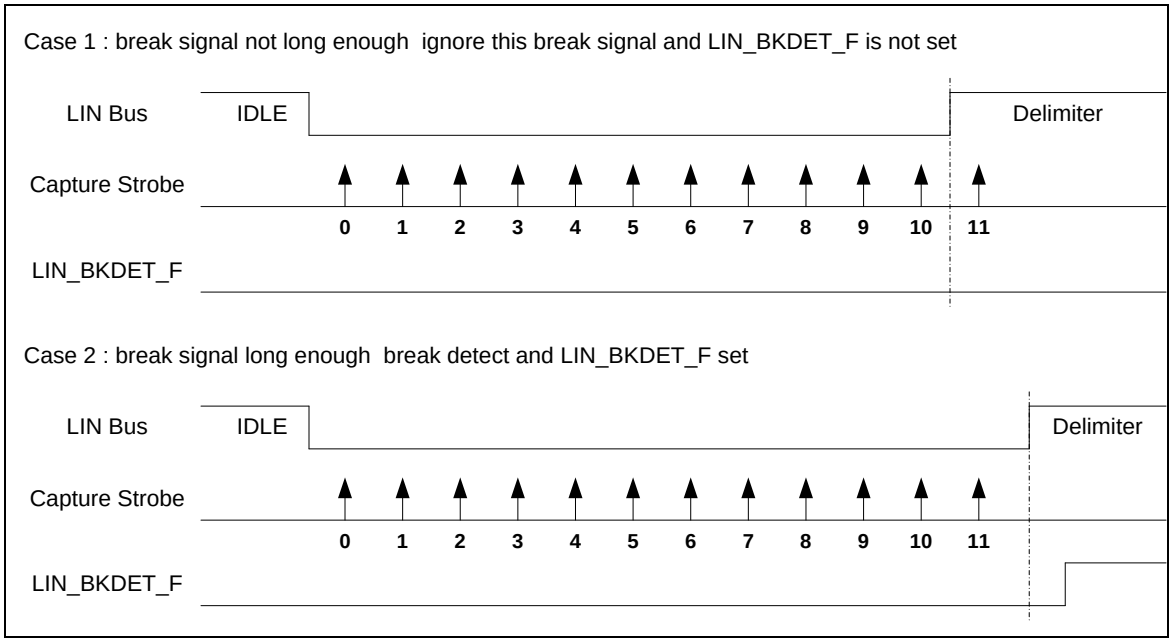


Figure 6-104 Break detection in LIN mode

LIN Frame ID and Parity Format

The LIN frame ID value in LIN function mode is shown, the frame ID parity can be generated by software or hardware depends on LIN_IDPEN (UA_LIN_CTL[9]) = 1.

If the parity generated by hardware, user fill ID0~ID5, (LIN_PID [29:24])hardware will calculate P0 (LIN_PID[30]) and P1 (LIN_PID[31]), otherwise user must filled frame ID and parity in this field.

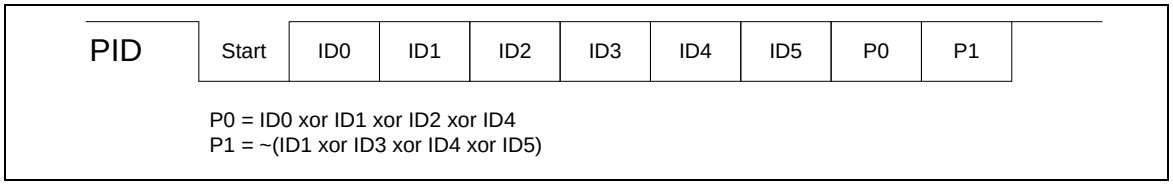


Figure 6-105LIN Frame ID and Parity Format

6.13.5.9.4 LIN Slave Mode

The UART controller support LIN slave mode by setting the LINS_EN (UA_LIN_CTL[0]). To enable and initialize the LIN slave mode, the following steps are necessary:

1. Select the desired baud-rate by setting the UA_BAUD register.
2. Configure the data length to 8 bits by setting WLS (UA_LCR[1:0]) = 11and disable parity check by clearing PBE (UA_LCR[3]) bit and configure the stop bit to 1 by clearing NSB (UA_LCR[2])] bit.
3. Select LIN function mode by setting UA_FUN_SEL register.
4. Enable LIN slave mode by setting the LINS_EN (UA_LIN_CTL[0]).

LIN header reception

According to the LIN protocol, a slave node must wait for a valid header which came from the master node. Then application will take one of following actions (depend on the master header frame ID value)

- Receive the response.
- Transmit the response.
- Ignore the response and wait for next header.

In LIN slave mode, user can enable slave header detection function by setting LINS_HDET_EN (UA_LIN_CTL[1]) register to detect complete frame header (receive “break field”, “sync field” and “frame ID field”). When a LIN header is received, the LINS_HDET_F (UA_LIN_SR[0]) register will be set (If the LIN_IEN (UA_IER[8]) bit =1, an interrupt will be generated). User can enable frame ID parity check function by setting LIN_IDPEN (UA_LIN_CTL[9]). If only received frame ID parity is not correct(break and sync filed are correct), the LIN_IDPERR_F (UA_LIN_SR[2]) flag (If the LIN_IEN (UA_IER[8]) =1, an interrupt will be generated) and LINS_HDET_F (UA_LIN_SR[0]) both will be set. User also can put LIN in mute mode by setting LIN_MUTE_EN (UA_LIN_CTL[4]) to 1. This mode allows detection of headers only (break + sync + frame ID) and prevents the reception of any other characters. In order to avoid bit rate tolerance, the controller support automatic resynchronization function to avoid clock deviation error, user can enable this feature by setting LINS_ARS_EN (UA_LIN_CTL[2]) register.

LIN response transmission

LIN slave node can transmit response and receive response. When slave node is the publisher of the response, the slave node send response by filling data to UA_THR register, and if the slave node is the subscriber of the response, the slave node received data from LIN bus.

LIN header time-out error

The LIN slave controller contains a header time-out counter. If the entire header is not received within the maximum time limit of 57 bit times, the header error flag LINS_HERR_F (UR_LIN_SR[1]) will be set. The time-out counter is enabled at each break detect edge and stopped in the following conditions.

- A LIN frame ID field has been received.
- The header error flag assert.
- Software write 1 to LINS_SYNC_F(UR_LIN_SR[3]) to re-search new frame header.

Mute mode and LIN exit from mute mode condition

In mute mode, LIN slave node will not receive any data until specified condition occurred. It allows detection of headers only and prevents the reception of any other characters. User can enable mute mode by setting LIN_MUTE_EN (UA_LIN_CTL[4]) and exit from mute mode condition can be selected by LIN_HEAD_SEL (UA_LIN_CTL[23:22]).

Note: It is recommended to set LIN slave node to mute mode after checksum transmission.

LIN slave controller exit from mute mode conditions is shown as follows:If LIN_HEAD_SEL (UA_LIN_CTL[23:22]) is set to “break field”, when LIN slave controller detects a valid LIN break + delimiter, the controller will enable the receiver (exit from mute mode) and subsequent data(sync data, frame ID data, response data) are received in RX-FIFO.

IfLIN_HEAD_SEL (UA_LIN_CTL[23:22]) is set to “break field and sync field”, when LIN slave controller detects a valid LIN break + delimiter followed by a valid sync field without frame error, the controller will enable the receiver (exit from mute mode) and subsequent data(ID data, response data) are received in RX-FIFO.

If LIN_HEAD_SEL (UA_LIN_CTL[23:22]) is set to “break field, sync field and ID field”, when LIN slave controller detects a valid LIN break + delimiter and valid sync field without frame error followed by ID data without frame error and received ID data matched LIN_PID (UA_LIN_CTL[31:24]) value. The controller will enable the receiver (exit from mute mode) and subsequent data(response data) are received in RX-FIFO.

Slave mode without automatic resynchronization

User can disable automatic resynchronization function to fix the communication baud rate. When operating in without automatic resynchronization mode, software need some initial process, and the initialization process flow of without automatic resynchronization mode is shown as follows:

1. Select the desired baud-rate by setting the UA_BAUD register.
2. Select LIN function mode by setting UA_FUN_SEL register.
3. Disable automatic resynchronization function by setting LINS_ARS_EN (UA_LIN_CTL[2]) = 0.
4. Enable LIN slave mode by setting the LINS_EN (UA_LIN_CTL[0]).

Slave mode with automatic resynchronization

User can enable automatic resynchronization function by setting LINS_ARS_EN (UA_LIN_CTL[2]). In automatic resynchronization mode, the controller will adjust the baud rate generator after each sync field reception. The initialization process flow of automatic resynchronization mode is shown as follows:

1. Select the desired baud-rate by setting the UA_BAUD register.
2. Select LIN function mode by setting UA_FUN_SEL register.
3. Enable automatic resynchronization function by setting LINS_ARS_EN (UA_LIN_CTL[2]) = 1.
4. Enable LIN slave mode by setting the LINS_EN (UA_LIN_CTL[0]).

When automatic resynchronization function is enabled, after each LIN break field, the time duration between five falling edges is sampled on UART peripheral clock and the result of this measurement is stored in an internal 13-bit register and the UA_BAUD register value will automatically updated at the end of the fifth falling edge. If the measure timer (13bit) overflow before five falling edges, then the header error flag LINS_HERR_F (UA_LIN_SR[1])will be set.

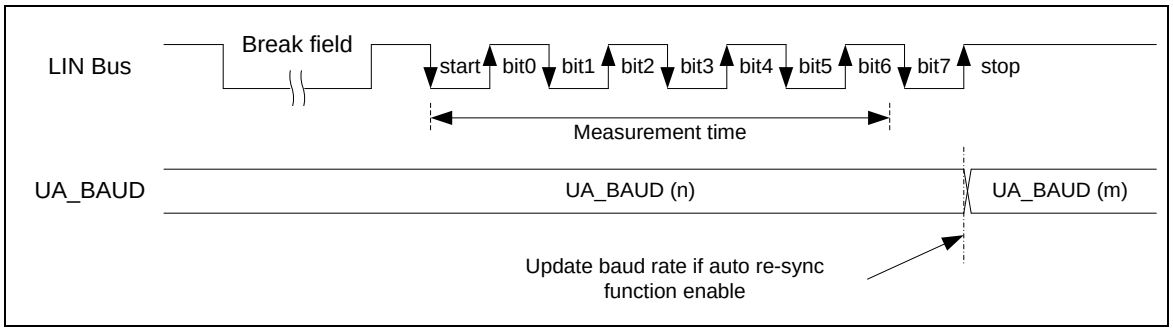


Figure 6-106 LIN sync field measurement

When operating in automatic resynchronization mode, software must select the desired baud rate by setting the UA_BAUD register and hardware will store it at internal TEMP_REG register, after each LIN break field, the time duration between five falling edges is sampled on UART peripheralclock and the result of this measurement is stored in an internal 13-bit register (BAUD_LIN) and the result will be updated to UA_BAUD register automatically.

In order to guarantee the transmission baud rate, the baud rate generator must reload the initial value before each new break reception. The initial value is programmed by the application during initialization (TEMP_REG). User can setting LINS_DUM_EN (UA_LIN_CTL[3]) to enable auto reload initial baud rate value function. If the LINS_DUM_EN (UA_LIN_CTL[3]) is set, when received the next character, hardware will auto reload the initial value to UA_BAUD, and when the UA_BAUD be updated, the LINS_DUM_EN (UA_LIN_CTL[3]) will be cleared automatically. The behavior of LIN updated method as shown asFigure 6-107 and Figure 6-108.

Note1: It is recommended setting the LINS_DUM_EN (UA_LIN_CTL[3]) before every checksum reception.

Note2: When header error been detected, user must writing 1 to LINS_SYNC_F (UA_LIN_SR[3]) register to re-search new frame header. When user writing 1 to it, hardware will reload the initial baud-rate (TEMP_REG) and re-search new frame header.

Note3: When operation in automatic resynchronization mode, the baud rate setting must be mode2 (DIV_X_EN (UA_BAUD[29]) and DIV_X_ONE (UA_BAUD[28]) must be 1).

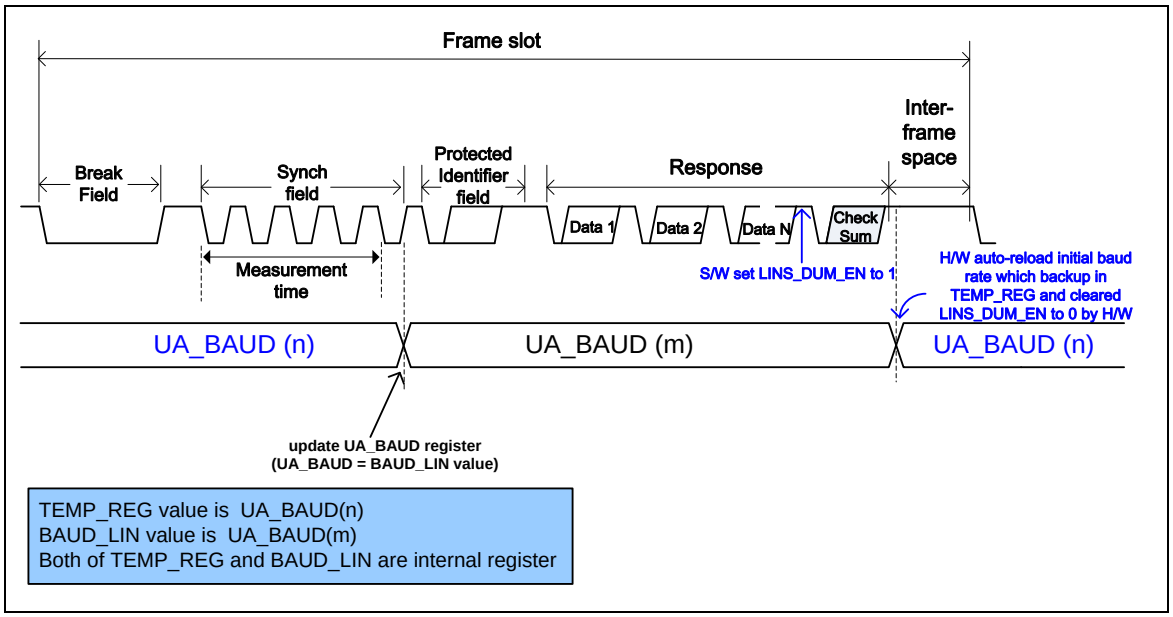


Figure 6-107UA_BAUD update sequence in automatic resynchronization mode when LINS_DUM_EN (UA_LIN_CTL[3]) = 1

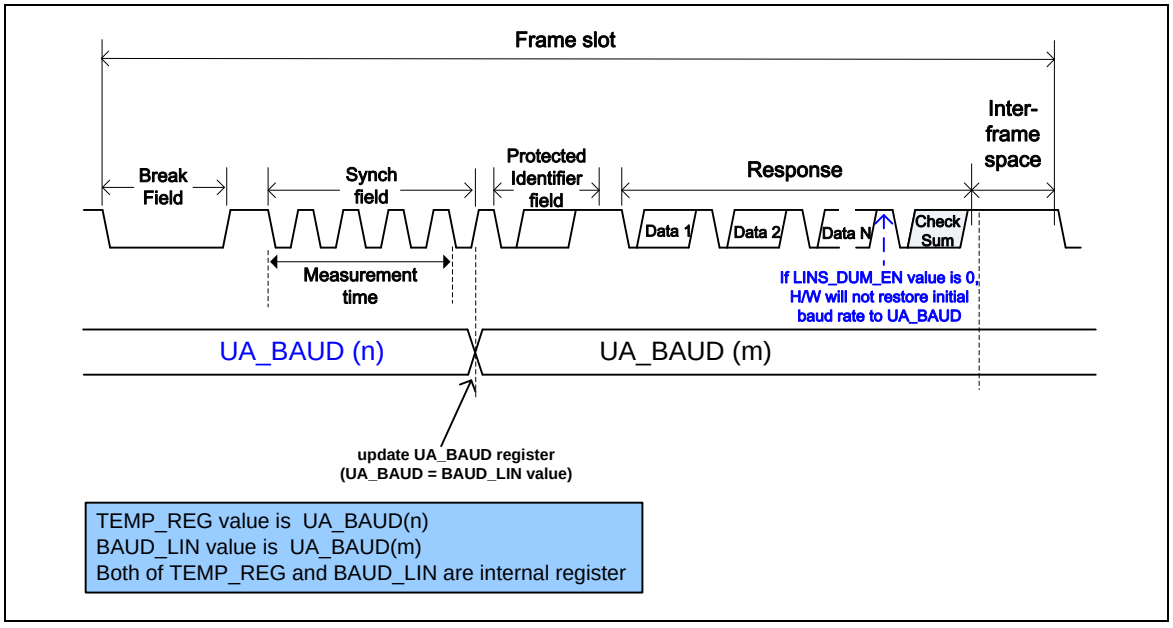


Figure 6-108UA_BAUD update sequence in automatic resynchronization mode when LINS_DUM_EN (UA_LIN_CTL[3]) = 0

Deviation error on the sync field

When operating in automatic resynchronization mode, the controller will check the deviation error on the sync field. The deviation error is checked by comparing the current baud rate with the received sync field. Two checks are performed in parallel.

Check1: Based on measurement of time between the first falling edge and the last falling edge of the sync field.

- If the difference more than 14.84%, the header error flag LINS_HERR_F (UA_LIN_SR[1]) will be set.
- If the difference less than 14.06%, the header error flag LINS_HERR_F (UA_LIN_SR[1]) will not be set.
- If the difference between 14.84% and 14.06%, the header error flag LINS_HERR_F (UA_LIN_SR[1]) may either set or not (it is depending on the data dephasing).

Check2: Based on measurement of time between each falling edge of the sync field.

- If the difference morethan 18.75%, the header error flag LINS_HERR_F (UA_LIN_SR[1]) will be set.
- If the difference less than 15.62%, the header error flag LINS_HERR_F (UA_LIN_SR[1]) will not be set.
- If the difference between 18.75% and 15.62%, the header error flag LINS_HERR_F (UA_LIN_SR[1]) may either set or not (it is depending on the data dephasing).

Note: The deviation check is based on the current baud-rate clock. Therefore, in order to guarantee correct deviation checking, the baud-rate must reload the nominal value before each new break reception by setting LINS_DUM_EN (UA_LIN_CTL[3]) register (It is recommend setting the LINS_DUM_EN (UA_LIN_CTL[3]) before every checksum reception).

LIN header error detection

In LIN slave function mode, when user enables header detection function by setting LINS_HDET_EN (UA_LIN_CTL[1]) register, the hardware will handle the header detect flow. If the header has error, the LIN header error flag LINS_HERR_F (UA_LIN_SR[1]) will be set and an interrupt is generated if the LIN_IEN (UA_IER[8]) is set. When header error isdetected, user must to reset the detect circuit to re-search new frame header by writing 1 to LINS_SYNC_F (UA_LIN_SR[3]) register.

The LIN header error flag LINS_HERR_F (UA_LIN_SR[1]) is set if one of the following conditions occurs:

- Break Delimiter is too short (less than 0.5 bit time).
- Frame error in sync field or Identifier field.
- The sync field data is not 0x55 (without automatic resynchronization mode).
- The sync field deviation error (with automatic resynchronization mode).
- The sync field measure time-out (with automatic resynchronization mode).
- LIN header reception time-out

6.13.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
UART Base Address: **UARTx_BA = 0x4005_0000 + (0x10_0000 * x)** **x = 0, 1**				
UA_RBR	UARTx_BA+0x00	R	UART Receive Buffer Register	Undefined
UA_THR	UARTx_BA+0x00	W	UART Transmit Holding Register	Undefined
UA_IER	UARTx_BA+0x04	R/W	UART Interrupt Enable Register	0x0000_0000
UA_FCR	UARTx_BA+0x08	R/W	UART FIFO Control Register	0x0000_0000
UA_LCR	UARTx_BA+0x0C	R/W	UART Line Control Register	0x0000_0000
UA_MCR	UARTx_BA+0x10	R/W	UART Modem Control Register	0x0000_0200
UA_MSR	UARTx_BA+0x14	R/W	UART Modem Status Register	0x0000_0110
UA_FSR	UARTx_BA+0x18	R/W	UART FIFO Status Register	0x1040_4000
UA_ISR	UARTx_BA+0x1C	R/W	UART Interrupt Status Register	0x0000_0002
UA_TOR	UARTx_BA+0x20	R/W	UART Time Out Register	0x0000_0000
UA_BAUD	UARTx_BA+0x24	R/W	UART Baud Rate Divisor Register	0x0F00_0000
UA_IRCR	UARTx_BA+0x28	R/W	UART IrDA Control Register	0x0000_0040
UA_ALT_CSR	UARTx_BA+0x2C	R/W	UART Alternate Control/Status Register	0x0000_000C
UA_FUN_SEL	UARTx_BA+0x30	R/W	UART Function Select Register	0x0000_0000
UA_LIN_CTL	UARTx_BA+0x34	R/W	UART LIN Control Register	0x000C_0000
UA_LIN_SR	UARTx_BA+0x38	R/W	UART LIN Status Register	0x0000_0000

6.13.7 Register Description

Receive Buffer Register (UA_RBR)

Register	Offset	R/W	Description	Reset Value
UA_RBR	UARTx_BA+0x00	R	UART 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 UART_RXD pin (LSB first).

Transmit Holding Register (UA_THR)

Register	Offset	R/W	Description	Reset Value
UA_THR	UARTx_BA+0x00	W	UART 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 one byte to this register, the data byte will be stored in transmitter FIFO. The UART Controller will send out the data stored in transmitter FIFO top location through the UART_TXD pin. (LSB first)

Interrupt Enable Register (UA_IER)

Register	Offset	R/W	Description	Reset Value
UA_IER	UARTx_BA+0x04	R/W	UART Interrupt Enable Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		AUTO_CTS_EN	AUTO_RTS_EN	TIME_OUT_EN	Reserved		LIN_IEN
7	6	5	4	3	2	1	0
Reserved	WAKE_EN	BUF_ERR_IEN	TOUT_IEN	MODEM_IEN	RLS_IEN	THRE_IEN	RDA_IEN

Bits	Description	
[31:14]	**Reserved**	Reserved.
[13]	**AUTO_CTS_EN**	**nCTS Auto Flow Control Enable Bit** 0 = nCTS auto-flow control Disabled. 1 = nCTS auto-flow control Enabled. When nCTS auto-flow is enabled, the UART will send data to external device when nCTS input assert (UART will not send data to device until nCTS is asserted).
[12]	**AUTO_RTS_EN**	**nRTS Auto Flow Control Enable Bit** 0 = nRTS auto-flow control Disabled. 1 = nRTS auto-flow control Enabled. When nRTS auto-flow is enabled, if the number of bytes in the RX FIFO equals the RTS_TRI_LEV (UA_FCR[19:16]), the UART will de-assert nRTS signal.
[11]	**TIME_OUT_EN**	**Receive Buffer Time Out Counter Enable Bit** 0 = Receive Buffer time out counter Disabled. 1 = Receive Buffer time out counter Enabled.
[10:9]	**Reserved**	Reserved.
[8]	**LIN_IEN**	**LIN Bus Interrupt Enable Bit** 0 = Lin bus interrupt Disabled. 1 = Lin bus interrupt Enabled. **Note:**This field is used for LIN function mode.
[7]	**Reserved**	Reserved.
[6]	**WAKE_EN**	**UART Wake-up Function Enable Bit** 0 = UART wake-up function Disabled. 1 = UART wake-up function Enabled, when the chip is in Power-down mode, an external nCTS change will wake-up chip from Power-down mode.
[5]	**BUF_ERR_IEN**	**Buffer Error Interrupt Enable Bit**

		0 = Buffer errorinterrupt Disabled. 1 =Buffer errorinterruptEnabled.
[4]	**TOUT_IEN**	**RX Time Out Interrupt Enable Bit** 0 =RX time-out interrupt Disabled. 1 = RX time-outinterruptEnabled.
[3]	**MODEM_IEN**	**Modem Status Interrupt Enable Bit** 0 = Modem status interrupt Disabled. 1 = Modem statusinterruptEnabled.
[2]	**RLS_IEN**	**Receive Line Status Interrupt Enable Bit** 0 = Receive Line Status interrupt Disabled. 1 = Receive Line Status interrupt Enabled.
[1]	**THRE_IEN**	**Transmit Holding Register Empty Interrupt Enable Bit** 0 = Transmit holding register empty interrupt Disabled. 1 = Transmit holding register empty interrupt Enabled.
[0]	**RDA_IEN**	**Receive Data Available Interrupt Enable Bit** 0 = Receive data available interrupt Disabled. 1 = Receive data available interrupt Enabled.

FIFO Control Register (UA_FCR)

Register	Offset	R/W	Description	Reset Value
UA_FCR	UARTx_BA+0x08	R/W	UART FIFO Control Register	0x0000_0000

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

Bits	Description	
[31:20]	**Reserved**	Reserved.
[19:16]	**RTS_TRI_LEV**	**nRTS Trigger Level for Auto-flow Control Use** 0000 = nRTS Trigger Level is 1 byte. 0001 = nRTS Trigger Level is 4 bytes. 0010 = nRTS Trigger Level is 8 bytes. 0011 = nRTS Trigger Level is 14 bytes. Others = Reserved. **Note:** This field is used for auto nRTS flow control.
[15:9]	**Reserved**	Reserved.
[8]	**RX_DIS**	**Receiver Disable Register** The receiver is disabled or not (set 1 to disable receiver) 0 = Receiver Enabled. 1 = Receiver Disabled. **Note:**This field is used for RS-485 Normal Multi-drop mode.It should be programmed before RS485_NMM (UA_ALT_CSR [8]) is programmed.
[7:4]	**RFITL**	**RX FIFO Interrupt Trigger Level** When the number of bytes in the receive FIFO equals the RFITL, the RDA_IF will be set (if RDA_IEN(UA_IER [0]) enabled, and an interrupt will be generated). 0000 = RX FIFO Interrupt Trigger Level is 1 byte. 0001 = RX FIFO Interrupt Trigger Level is 4 bytes. 0010 = RX FIFO Interrupt Trigger Level is 8 bytes. 0011 = RX FIFO Interrupt Trigger Level is 14 bytes. Others = Reserved.
[3]	**Reserved**	Reserved.
[2]	**TFR**	**TX Field Software Reset** When TFR is set, all the byte in the transmit FIFO and TX internal state machine are cleared.

		0 = No effect. 1 = Reset the TX internal state machine and pointers. **Note:** This bit will automatically clear at least 3 UART engine clock cycles.
[1]	**RFR**	**RX Field Software Reset** When RFR is set, all the byte in the receiver FIFO and RX internal state machine are cleared. 0 = No effect. 1 = Reset the RX internal state machine and pointers. **Note:** This bit will automatically clear at least 3 UART engine clock cycles.
[0]	**Reserved**	Reserved.

Line Control Register (UA_LCR)

Register	Offset	R/W	Description	Reset Value
UA_LCR	UARTx_BA+0x0C	R/W	UART Line Control Register	0x0000_0000

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

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**BCB**	**Break Control Bit** 0 = Break Control Disabled. 1 = Break Control Enabled. **Note:**When this bit is set to logic 1, the serial data output (TX) is forced to the Spacing State (logic 0). This bit acts only on TX and has no effect on the transmitter logic.
[5]	**SPE**	**Stick Parity Enable Bit** 0 = Stick parity Disabled. 1 = Stick parity Enabled. **Note:**If PBE (UA_LCR[3]) and EBE (UA_LCR[4]) are logic 1, the parity bit is transmitted and checked as logic 0. If PBE (UA_LCR[3]) is 1 and EBE (UA_LCR[4]) is 0 then the parity bit is transmitted and checked as 1.
[4]	**EPE**	**Even Parity Enable Bit** 0 = Odd number of logic 1's is transmitted and checked in each word. 1 = Even number of logic 1's is transmitted and checked in each word. **Note:**This bit has effect only when PBE (UA_LCR[3]) is set.
[3]	**PBE**	**Parity Bit Enable Bit** 0 = No parity bit. 1 = Parity bit generated Enabled. **Note:**Parity bit is generated on each outgoing character and is checked on each incoming data.
[2]	**NSB**	**Number of "STOP Bit"** 0 = One " STOP bit" is generated in the transmitted data. 1 = When select 5-bit word length, 1.5 "STOP bit" is generated in the transmitted data. When select 6-,7- and 8-bit word length, 2 "STOP bit" is generated in the transmitted data.
[1:0]	**WLS**	**Word Length Selection** 00 = character length is 5-bit. 01 = character length is 6-bit.

		10 = character length is 7-bit. 11 = character length is 8-bit.

MODEM Control Register (UA_MCR)

Register	Offset	R/W	Description	Reset Value
UA_MCR	UARTx_BA+0x10	R/W	UART Modem Control Register	0x0000_0200

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved		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**	**nRTS Pin State (Read Only)** This bit mirror from nRTS pin output of voltage logic status. 0 = nRTS pin output is low level voltage logic state. 1 = nRTS pin output is high level voltage logic state.
[12:10]	**Reserved**	Reserved.
[9]	**LEV_RTS**	**nRTS Pin Active Level** This bit defines the active level state of nRTS pin output. 0 =nRTS pin output is high level active. 1 = nRTS pin output is low level active. (Default) **Note1:**Refer to Figure 6-95and Figure 6-96 for UART function mode. **Note2:** Refer to Figure 6-99and Figure 6-100 for RS-485 function mode.
[8:2]	**Reserved**	Reserved.
[1]	**RTS**	**nRTS (Request-to-send) Signal** This bit is direct control internal nRTS signal active or not, and then drive the nRTS pin output with LEV_RTS bit configuration. 0 = nRTS signal is active. 1 = nRTS signal is inactive. **Note1:** This nRTS signal control bit is not effective when nRTS auto-flow control is enabled in UART function mode. **Note2:** This nRTS signal control bit is not effective when RS-485 auto direction mode (AUD) is enabled in RS-485 function mode.
[0]	**Reserved**	Reserved.

Modem Status Register (UA_MSR)

Register	Offset	R/W	Description	Reset Value
UA_MSR	UARTx_BA+0x14	R/W	UART Modem Status Register	0x0000_0110

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

Bits	Description	
[31:9]	**Reserved**	Reserved.
[8]	**LEV_CTS**	**nCTS Pin Active Level** This bit defines the active level state of nCTS pin input. 0 = nCTS pin input is high level active. 1 = nCTS pin input is low level active. (Default)
[7:5]	**Reserved**	Reserved.
[4]	**CTS_ST**	**nCTS Pin Status (Read Only)** This bit mirror from nCTS pin input of voltage logic status. 0 = nCTS pin input is low level voltage logic state. 1 = nCTS pin input is low level voltage logic state. **Note:** This bit echoes when UART Controller peripheral clock is enabled, and nCTS multi-function port is selected.
[3:1]	**Reserved**	Reserved.
[0]	**DCTSF**	**Detect nCTS State Change Flag** This bit is set whenever nCTS input has change state, and it will generate Modem interrupt to CPU when MODEM_IEN (UA_IER[3]) is set to 1. Write 1 to clear this bit to 0

FIFO Status Register (UA_FSR)

Register	Offset	R/W	Description	Reset Value
UA_FSR	UARTx_BA+0x18	R/W	UART 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)** This bit is set by hardware when TX FIFO (UA_THR) is empty and the STOP bit of the last byte has been transmitted. 0 = TX FIFO is not empty or the STOP bit of the last byte has not been transmitted. 1 = TX FIFO is empty and the STOP bit of the last byte has been transmitted. **Note:** This bit is cleared automatically when TX FIFO is not empty or the last byte transmission has not completed.
[27:25]	**Reserved**	Reserved.
[24]	**TX_OVER_IF**	**TX Overflow Error Interrupt Flag** If TX FIFO (UA_THR) is full, an additional write to UA_THR will cause this bit to logic 1. 0 = TX FIFO is not overflow. 1 = TX FIFO is overflow. **Note:**This bit can be cleared by writing '1' to it.
[23]	**TX_FULL**	**Transmitter FIFO Full (Read Only)** This bit indicates TX FIFO is full or not. 0 = TX FIFO is not full. 1 = TX FIFO is full. **Note:**This bit is set when the number of usage in TX FIFO Buffer equal to 16, otherwise it is cleared by hardware.
[22]	**TX_EMPTY**	**Transmitter FIFO Empty (Read Only)** This bit indicates TX FIFO is empty or not. 0 = TX FIFO is not empty. 1 = TX FIFO is empty. **Note:**When the last byte of TX FIFO has been transferred to Transmitter Shift Register, hardware sets this bit high. It will be cleared when writing data into UA_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. The Maximum value shown in TX_POINTER is 15. When the using level of TX FIFO Buffer equal to 16, the TX_FULL bit is set to 1 and TX_POINTER will show 0. As one byte of TX FIFO is transferred to Transmitter Shift Register, the TX_FULL bit is cleared to 0 and TX_POINTER will show 15.
[15]	**RX_FULL**	**Receiver FIFO Full (Read Only)** This bit indicatesRX FIFO is full or not. 0 = RX FIFO is not full. 1 = RX FIFO is full. **Note:**This bit is set when the number of usage in RX FIFO Buffer is equal to 16, otherwise is cleared by hardware.
[14]	**RX_EMPTY**	**Receiver FIFO Empty (Read Only)** This bit indicatesRX FIFO empty or not. 0 = RX FIFO is not empty. 1 = RX FIFO is empty. **Note:**When the last byte of RX FIFO has been read by CPU, hardware sets this bit high. It will be cleared when UART receives any new data.
[13:8]	**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. The Maximum value shown in RX_POINTER is 15. When the using level of RX FIFO Buffer equal to 16, the RX_FULL bit is set to 1 and RX_POINTER will show 0. As one byte of RX FIFO is read by CPU, the RX_FULL bit is cleared to 0 and RX_POINTER will show 15.
[7]	**Reserved**	Reserved.
[6]	**BIF**	**Break Error Interrupt Flag** 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). 0 = No Break error is generated. 1 = Break error is generated. **Note:**This bit can be cleared by writing ‘1’ to it.
[5]	**FEF**	**Frame Error Flag** 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). 0 = No frame error is generated. 1 = Frame error is generated. **Note:**This bit can be cleared by writing ‘1’ to it.
[4]	**PEF**	**Parity Error Flag** This bit is set to logic 1 whenever the received character does not have a valid “parity bit”. 0 = No parity error is generated. 1 = Parity error is generated. **Note:**This bit can be cleared by writing ‘1’ to it.
[3]	**RS485_ADD_DETF**	**RS-485 Address Byte Detection Flag** 0 = Receiver detects a data that is not an address bit (bit 9 =’0’).

		1 = Receiver detects a data that is an address bit (bit 9 ='1'). **Note1:** This field is used for RS-485 function mode and RS485_ADD_EN (UA_ALT_CSR[15]) is set to 1to enable Address detection mode. **Note2:**This bit can be cleared by writing '1' to it.
[2:1]	**Reserved**	Reserved.
[0]	**RX_OVER_IF**	**RX Overflow Error Interrupt Flag** This bit is set when RX FIFO overflow. If the number of bytes of received data is greater than RX_FIFO (UA_RBR) size, this bit will be set. 0 = RX FIFO is not overflow. 1 = RX FIFO is overflow. **Note:**This bit can be cleared by writing '1' to it.

Interrupt Status Control Register (UA_ISR)

Register	Offset	R/W	Description	Reset Value
UA_ISR	UARTx_BA+0x1C	R/W	UART Interrupt Status Register	0x0000_0002

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**LIN_INT**	**LIN Bus Interrupt Indicator (Read Only)** This bit is set if LIN_IEN(UA_IER[8]) and LIN_IF(UA_ISR[7]) are both set to 1. 0 = No LIN Bus interrupt is generated. 1 = The LIN Bus interrupt is generated.
[14]	**Reserved**	Reserved.
[13]	**BUF_ERR_INT**	**Buffer Error Interrupt Indicator (Read Only)** This bit is set if BUF_ERR_IEN (UA_IER[5]) and BUF_ERR_IF (UA_ISR[5])are both set to 1. 0 = No buffer error interrupt is generated. 1 = Buffer error interrupt is generated.
[12]	**TOUT_INT**	**Time Out Interrupt Indicator (Read Only)** This bit is set if TOUT_IEN(UA_IER[4])and TOUT_IF(UA_ISR[4]) are both set to 1. 0 = No time-out interrupt is generated. 1 = Time-outinterrupt is generated.
[11]	**MODEM_INT**	**MODEM Status Interrupt Indicator (Read Only)** This bit is set if MODEM_IEN(UA_IER[3]) and MODEM_IF(UA_ISR[3]) are both set to 1. 0 = No Modem interrupt is generated. 1 = Modem interrupt is generated.
[10]	**RLS_INT**	**Receive Line Status Interrupt Indicator (Read Only)** This bit is set if RLS_IEN(UA_IER[2] and RLS_IF(UA_ISR[2]) are both set to 1. 0 = No RLS interrupt is generated. 1 = RLS interrupt is generated.
[9]	**THRE_INT**	**Transmit Holding Register Empty Interrupt Indicator (Read Only)** This bit is set if THRE_IEN (UA_IER[1]) and THRE_IF(UA_ISR[1]) are both set to 1. 0 = No THRE interrupt is generated. 1 = THRE interrupt is generated.

[8]	**RDA_INT**	**Receive Data Available Interrupt Indicator (Read Only)** This bit is set if RDA_IEN (UA_IER[0]) and RDA_IF (UA_ISR[0]) are both set to 1. 0 = No RDA interrupt is generated. 1 = RDA interrupt is generated.
[7]	**LIN_IF**	**LIN Bus Flag (Read Only)** This bit is set when LIN slave header detect (LINS_HDET_F(UA_LIN_SR[0]) =1), LIN break detect (LIN_BKDET_F(UA_LIN_SR[8]) =1), bit error detect (BIT_ERR_F(UA_LIN_SR[9]) =1), LIN slave ID parity error (LINS_IDPERR_F(UA_LIN_SR[2]) = 1) or LIN slave header error detect (LINS_HERR_F(UA_LIN_SR[1]) = 1). If LIN_ IEN(UA_IER [8]) is enabled the LIN interrupt will be generated. 0 = None of LINS_HDET_F, LIN_BKDET_F, BIT_ERR_F, LINS_IDPERR_F and LINS_HERR_F is generated. 1 = At least one of LINS_HDET_F, LIN_BKDET_F, BIT_ERR_F, LINS_IDPERR_F and LINS_HERR_F is generated. **Note:**This bit is read only. This bit is cleared when LINS_HDET_F(UA_LIN_SR[0]), LIN_BKDET_F(UA_LIN_SR[8]), BIT_ERR_F(UA_LIN_SR[9]), LINS_IDPENR_F(UA_LIN_SR[2]) and LINS_HERR_F(UA_LIN_SR[1]) all are cleared.
[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 (UA_FSR[24]) or RX_OVER_IF (UA_FSR[0])is set. When BUF_ERR_IF (UA_ISR[5]) is set, the transfer is not correct. If BUF_ERR_IEN (UA_IER [5]) is enabled, the buffer error interrupt will be generated. 0 = No buffer error interrupt flag is generated. 1 = Buffer error interrupt flag is generated. **Note:**This bit is cleared if both of RX_OVER_IF (UA_FSR[0]) and TX_OVER_IF (UA_FSR[24]) are cleared to 0 by writing 1 to RX_OVER_IF (UA_FSR[0]) and TX_OVER_IF (UA_FSR[24]).
[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 (UA_TOR[7:0]). If TOUT_IEN (UA_IER [4]) is enabled, the time-out interrupt will be generated. 0 = No Time-out interrupt flag is generated. 1 = Time-out interrupt flag is generated. **Note:**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 nCTS pin has state change (DCTSF (UA_MSR[0])=1). If MODEM_IEN(UA_IER [3]) is enabled, the Modem interrupt will be generated. 0 = No Modem interrupt flag is generated. 1 = Modem interrupt flag is generated. **Note:** This bit is reset to 0 when bit DCTSF (UA_MSR[0]) is cleared by a write 1 on DCTSF (UA_MSR[0]).
[2]	**RLS_IF**	**Receive Line Interrupt Flag (Read Only)** This bit is set when the RX receive data have parity error, frame error or break error (at least one of 3 bits, BIF(UA_FSR[6]), FEF(UA_FSR[5]) and PEF(UA_FSR[4]), is set). If RLS_IEN(UA_IER [2]) is enabled, the RLS interrupt will be generated. 0 = No RLS interrupt flag is generated. 1 = RLS interrupt flag is generated. **Note1:** In RS-485 function mode, this field is set include “receiver detect and received address byte character (bit9 = ‘1’) bit”. At the same time, the bit of RS485_ADD_DETF(UA_FSR[3]) is also set. **Note2:** This bit is read only and reset to 0 when all bits of BIF(UA_FSR[6]),

		FEF(UA_FSR[5]) and PEF(UA_FSR[4]) are cleared. **Note3:** In RS-485 function mode, this bit is read only and reset to 0 when all bits of BIF(UA_FSR[6]) , FEF(UA_FSR[5]) and PEF(UA_FSR[4])andRS485_ADD_DETF (UA_FSR[3]) 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 THRE_IEN(UA_IER [1]) is enabled, the THRE interrupt will be generated. 0 = No THRE interrupt flag is generated. 1 = THRE interrupt flag is generated. **Note:** This bit is read only and it will be cleared when writing data into THR (TX FIFO not empty).
[0]	**RDA_IF**	**Receive Data Available Interrupt Flag (Read Only)** When the number of bytes in the RX FIFO equals the RFITL(UA_FCR[7:4]) then the RDA_IF(UA_ISR[0])will be set. If RDA_IEN(UA_IER [0]) is enabled, the RDA interrupt will be generated. 0 = No RDA interrupt flag is generated. 1 = RDA interrupt flag is generated. **Note:** This bit is read only and it will be cleared when the number of unread bytes of RX FIFO drops below the threshold level (RFITL (UA_FCR[7:4])).

Time out Register (UA_TOR)

Register	Offset	R/W	Description	Reset Value
UA_TOR	UARTx_BA+0x20	R/W	UART Time Out Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**DLY**	**TX Delay Time Value** This field is used to programming the transfer delay time between the last stop bit and next start bit. The unit is bit time.
[7:0]	**TOIC**	**Time Out Interrupt Comparator** The time out counter resets and starts counting (the counting clock = baud rate) whenever the RX FIFO receives a new data wordif time out counter is enabled by setting TIME_OUT_EN (UA_IER[11]). Once the content of time out counter is equal to that of time out interrupt comparator (TOIC (UA_TOR[7:0])), a receiver time out interrupt (TOUT_INT (UA_ISR[12])) is generated if TOUT_IEN(UA_IER [4]) enabled. A new incoming data word or RX FIFO empty will clearTOUT_IF (UA_ISR[4]). In order to avoid receiver time out interrupt generation immediately during one character is being received, TOIC value should be set between 40 and 255. So, for example, if TOIC is set with 40, the time out interrupt is generated after four characters are not received when 1 stop bit and no parity check is set for UART transfer.

Baud Rate Divider Register (UA_BAUD)

Register	Offset	R/W	Description	Reset Value
UA_BAUD	UARTx_BA+0x24	R/W	UART 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 Bit** The BRD = Baud Rate Divider, and the baud rate equation is Baud Rate = Clock / [M * (BRD + 2)]; The default value of M is 16. 0 = Divider X Disabled (the equation of M = 16). 1 = Divider X Enabled (the equation of M = X+1, but DIVIDER_X [27:24] must >= 8). **Note1:** The detail description is shown in **UART Controller Baud Rate Generator** section. **Note2:** In IrDA mode, this bit must disable.
[28]	**DIV_X_ONE**	**Divider X Equal to 1** 0 = Divider M = X+1 (the equation of M = X+1, but DIVIDER_X[27:24] must >= 8). 1 = Divider M = 1. **Note:** The detail description is shown in **UART Controller Baud Rate Generator** section.
[27:24]	**DIVIDER_X**	**Divider X** The baud rate divider M = X+1. **Note 1:**This field is used for baud rate calculation in mode 1 and has no effect for baud rate calculation in mode 0 and mode 2. **Note 2:** The detail description is shown in **UART Controller Baud Rate Generator** section.
[23:16]	**Reserved**	Reserved.
[15:0]	**BRD**	**Baud Rate Divider** The field indicates the baud rate divider. This filed is used in baud rate calculation. **Note:** The detail description is shown in **UART Controller Baud Rate Generator** section.

IrDA Control Register (UA_IRCR)

Register	Offset	R/W	Description	Reset Value
UA_IRCR	UARTx_BA+0x28	R/W	UART IrDA Control Register	0x0000_0040

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

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**INV_RX**	**IrDA Inverse Receive Input Signal** 0= None inverse receiving input signal. 1= Inverse receiving input signal. (Default)
[5]	**INV_TX**	**IrDA Inverse Transmitting Output Signal** 0= None inverse transmitting signal. (Default) 1= Inverse transmitting output signal.
[4:2]	**Reserved**	Reserved.
[1]	**TX_SELECT**	**IrDA Receiver/Transmitter Selection Enable Bit** 0= IrDA Transmitter Disabled andReceiver Enabled. (Default) 1= IrDA Transmitter Enabled and Receiver Disabled.
[0]	**Reserved**	Reserved.

UART Alternate Control/Status Register (UA_ALT_CSR)

Register	Offset	R/W	Description	Reset Value
UA_ALT_CSR	UARTx_BA+0x2C	R/W	UART Alternate Control/Status Register	0x0000_000C

31	30	29	28	27	26	25	24
ADDR_MATCH							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
RS485_ADD_EN	Reserved				RS485_AUD	RS485_AAD	RS485_NMM
7	6	5	4	3	2	1	0
LIN_TX_EN	LIN_RX_EN	Reserved		LIN_BKFL			

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 Bit** This bit is used to enable RS-485 Address Detection mode. 0 = Address detection mode Disabled. 1 = Address detection mode Enabled. **Note:** This field is used for RS-485 any operation mode.
[14:11]	**Reserved**	Reserved.
[10]	**RS485_AUD**	**RS-485 Auto Direction Mode (AUD)** 0 = RS-485 Auto Direction Operation mode (AUO) Disabled. 1 = RS-485 Auto Direction Operation mode (AUO) Enabled. **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)** 0 = RS-485 Auto Address Detection Operation mode (AAD) Disabled. 1 = RS-485 Auto Address Detection Operation mode (AAD) Enabled. **Note:** It cannot be active with RS-485_NMM operation mode.
[8]	**RS485_NMM**	**RS-485 Normal Multi-drop Operation Mode (NMM)** 0 = RS-485 Normal Multi-drop Operation mode (NMM) Disabled. 1 = RS-485 Normal Multi-drop Operation mode (NMM) Enabled. **Note:** It cannot be active with RS-485_AAD operation mode.
[7]	**LIN_TX_EN**	**LIN TX Break Mode Enable Bit** 0 = LIN TX Break mode Disabled. 1 = LIN TX Break mode Enabled.

		Note: When TX break field transfer operation finished, this bit will be cleared automatically.
[6]	**LIN_RX_EN**	**LIN RX Enable Bit** 0 = LIN RX mode Disabled. 1 = LIN RX mode Enabled.
[5:4]	**Reserved**	Reserved.
[3:0]	**LIN_BKFL**	**UART LIN Break Field Length** This field indicates a 4-bit LIN TX break field count. **Note1:**This break field length is UA_LIN_BKFL + 1. **Note2:** According to LIN spec, the reset value is 0xC (break field length = 13).

UART Function Select Register (UA_FUN_SEL)

Register	Offset	R/W	Description	Reset Value
UA_FUN_SEL	UARTx_BA+0x30	R/W	UART Function Select Register	0x0000_0000

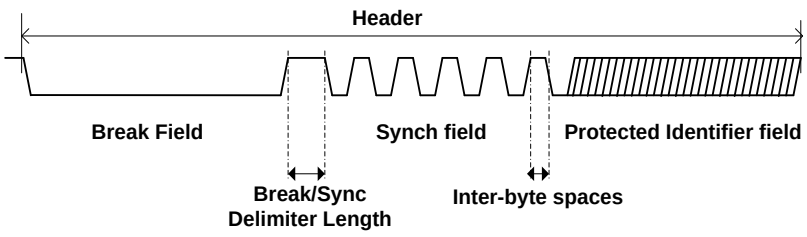
31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved						FUN_SEL	

Bits	Description	
[31:2]	**Reserved**	Reserved.
[1:0]	**FUN_SEL**	**Function Select Enable Bits** 00 = UART function Enabled. 01 = LIN function Enabled. 10 = IrDA function Enabled. 11 = RS-485 function Enabled.

UART LIN Control Register (UA_LIN_CTL)

Register	Offset	R/W	Description	Reset Value
UA_LIN_CTL	UARTx_BA+0x34	R/W	UART LIN Control Register	0x000C_0000

31	30	29	28	27	26	25	24
LIN_PID							
23	22	21	20	19	18	17	16
LIN_HEAD_SEL		LIN_BS_LEN		LIN_BKFL			
15	14	13	12	11	10	9	8
Reserved			BIT_ERR_EN	LIN_RX_DIS	LIN_BKDET_EN	LIN_IDPEN	LIN_SHD
7	6	5	4	3	2	1	0
Reserved			LIN_MUTE_EN	LINS_DUM_EN	LINS_ARS_EN	LINS_HDET_EN	LINS_EN

Bits	Description	
[31:24]	LIN_PID	**LIN PID Register** This field contains the LIN frame ID value when in LIN function mode, the frame ID parity can be generated by software or hardware depends on LIN_IDPEN (UA_LIN_CTL[9]). If the parity generated by hardware (LIN_IDPEN (UA_LIN_CTL[9]) = 1), user fill ID0~ID5, (LIN_PID[24:29]hardware will calculate P0(LIN_PID[30]) and P1(LIN_PID[31]), otherwise user must filled frame ID and parity in this field. **Note1:** User can filled any 8-bit value to this field and the bit 24 indicates ID0 (LSB first). **Note2:** This field can be used for LIN master mode or slave mode.
[23:22]	LIN_HEAD_SEL	**LIN Header Select** 00 = The LIN header includes “break field”. 01 = The LIN header includes “break field” and “sync field”. 10 = The LIN header includes “break field”, “sync field” and “frame ID field”. 11 = Reserved. **Note:** This bit is used to master mode for LIN to sending header field (LIN_SHD (UA_LIN_CTL[8]) = 1) or used to slave to indicates exit from mute mode condition(LIN_MUTE_EN (UA_LIN_CTL[4])).
[21:20]	LIN_BS_LEN	**LIN Break/Sync Delimiter Length** 00 = The LIN break/sync delimiter length is 1 bit time. 10 = The LIN break/sync delimiter length is 2 bit time. 10 = The LIN break/sync delimiter length is 3 bit time. 11 = The LIN break/sync delimiter length is 4 bit time. 

		Note: This bit is used for LIN master to sending header field.
[19:16]	**LIN_BKFL**	**LIN Break Field Length** This field indicates a 4-bit LIN TX break field count. **Note1:**These registers are shadow registers of LIN_BKFL (UA_ALT_CSR[19:16]), User can read/write it by setting LIN_BKFL (UA_ALT_CSR[3:0]) or LIN_BKFL (UA_LIN_CTL[19:16]). **Note2:**This break field length is LIN_BKFL + 1. **Note3:**According to LIN spec, the reset value is 0XC (break field length = 13).
[15:13]	**Reserved**	Reserved.
[12]	**BIT_ERR_EN**	**Bit Error Detect Enable Bit** 0 = Bit error detection function Disabled. 1 = Bit error detection Enabled. **Note:** In LIN function mode, when occur bit error, the BIT_ERR_F (UA_LIN_SR[9]) flag will be asserted. If the LIN_IEN (UA_IER[8]) = 1, an interrupt will be generated.
[11]	**LIN_RX_DIS**	**LIN Receiver Disable Bit** If the receiver is enabled (LIN_RX_DIS(UA_LIN_CTL[11]) = 0), all received byte data will be accepted and stored in the RX-FIFO, and if the receiver is disabled (LIN_RX_DIS (UA_LIN_CTL[11]) = 1), all received byte data will be ignore. 0 = LIN receiver Enabled. 1 = LIN receiver Disabled. **Note:** This bit is only valid when operating in LIN function mode (FUN_SEL (UA_FUN_SEL[1:0]) = 01).
[10]	**LIN_BKDET_EN**	**LIN Break Detection Enable Bit** When detect consecutive dominant greater than 11 bits, and are followed by a delimiter character, the LIN_BKDET_F (UA_LIN_SR[8]) flag is set in UA_LIN_SR register at the end of break field. If the LIN_IEN (UA_IER[8]) =1, an interrupt will be generated. 0 = LIN break detection Disabled. 1 = LIN break detection Enabled.
[9]	**LIN_IDPEN**	**LIN ID Parity Enable Bit** 0 = LIN frame ID parity Disabled. 1 = LIN frame ID parity Enabled. **Note1:** This bit can be used for LIN master to sending header field (LIN_SHD (UA_LIN_CTL[8]) = 1 and LIN_HEAD_SEL (UA_LIN_CTL[23:22]) = 10) or be used for enable LIN slave received frame ID parity checked. **Note2:** This bit is only use when operation header transmitter is in LIN_HEAD_SEL(UA_LIN_CTL[23:22]) = 10.
[8]	**LIN_SHD**	**LIN TX Send Header Enable Bit** The LIN TX header can be “break field” or “break and sync field” or “break, sync and frame ID field”, it is depend on setting LIN_HEAD_SEL (UA_LIN_CTL[23:22]). 0 = Send LIN TX header Disabled. 1 = Send LIN TX header Enabled. **Note1:**These registers are shadow registers of LIN_TX_EN(UA_ALT_CSR[7]); user can read/write it by setting LIN_TX_EN(UA_ALT_CSR[7]) or LIN_SHD(UA_LIN_CTL[8]). **Note2:**When transmitter header field (it may be “break” or “break + sync” or “break + sync + frame ID” selected by LIN_HEAD_SEL (UA_LIN_CTL[23:22]) field) transfer operation finished, this bit will be cleared automatically.
[7:5]	**Reserved**	Reserved.
[4]	**LIN_MUTE_EN**	**LIN Mute Mode Enable Bit** 0 = LIN mute mode Disabled. 1 = LIN mute mode Enabled.

		Note: The exit from mute mode condition and each control and interactions of this field are explained in LIN slave mode.
[3]	**LINS_DUM_EN**	**LIN Slave Divider Update Method Enable Bit** 0 = UA_BAUD updated is writing by software (if no automatic resynchronization update occurs at the same time). 1 = UA_BAUD is updated at the next received character. User must set the bit before checksum reception. **Note1:** This bit only valid when in LIN slave mode (LINS_EN (UA_LIN_CTL[0]) = 1). **Note2:**This bit is used for LIN Slave Automatic Resynchronization mode. (for Non-Automatic Resynchronization mode, this bit should be kept cleared) **Note3:** The control and interactions of this field are explained in 6.13.5.9.4 (Slave mode with automatic resynchronization).
[2]	**LINS_ARS_EN**	**LIN Slave Automatic Resynchronization Mode Enable Bit** 0 = LIN automatic resynchronization Disabled. 1 = LIN automatic resynchronization Enabled. **Note1:** This bit only valid when in LIN slave mode (LINS_EN (UA_LIN_CTL[0]) = 1). **Note2:** When operation in Automatic Resynchronization mode, the baud rate setting must be mode2 (DIV_X_EN(UA_BAUD [29]) and DIV_X_ONE (UA_BAUD [28]) must be 1). **Note3:** The control and interactions of this field are explained in 6.13.5.9.4 (Slave mode with automatic resynchronization).
[1]	**LINS_HDET_EN**	**LIN Slave Header Detection Enable Bit** 0 = LIN slave header detection Disabled. 1 = LIN slave header detection Enabled. **Note1:** This bit only valid when in LIN slave mode (LINS_EN(UA_LIN_CTL[0]) = 1). **Note2:** In LIN function mode, when detect header field (break + sync + frame ID), LINS_HDET_F(UA_LIN_SR [0]) flag will be asserted. If the LIN_IEN (UA_IER[8]) = 1, an interrupt will be generated.
[0]	**LINS_EN**	**LIN Slave Mode Enable Bit** 0 = LIN slave mode Disabled. 1 = LIN slave mode Enabled.

UART LIN Status Register (UA_LIN_SR)

Register	Offset	R/W	Description	Reset Value
UA_LIN_SR	UARTx_BA+0x38	R/W	UART LIN Status Register	0x0000_0000

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

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**BIT_ERR_F**	**Bit Error Detect Status Flag** At TX transfer state, hardware will monitoring the bus state, if the input pin (SIN) state not equals to the output pin (SOUT) state, BIT_ERR_F (UA_LIN_SR[9]) will be set. When occur bit error, if the LIN_IEN (UA_IER[8]) = 1, an interrupt will be generated. **Note1:** This bit is cleared by writing 1 to it. **Note2:** This bit is only valid when enable bit error detection function (BIT_ERR_EN (UA_LIN_CTL[12]) = 1).
[8]	**LIN_BKDET_F**	**LIN Break Detection Flag** This bit is set by hardware when a break is detected and be cleared by writing 1 to it. 0 = LIN break not detected. 1 = LIN break detected. **Note1:** This bit is cleared by writing 1 to it. **Note2:** This bit is only valid when enable LIN break detection function (LIN_BKDET_EN (UA_LIN_CTL[10]) = 1).
[7:4]	**Reserved**	Reserved.
[3]	**LINS_SYNC_F**	**LIN Slave Sync Field** This bit indicates that the LIN sync field is being analyzed in automatic resynchronization mode. When the receiver header have some error been detect, user must to reset the internal circuit to re-search new frame header by writing 1 to this bit. 0 = The current character is not at LIN sync state. 1 = The current character is at LIN sync state. **Note1:** This bit only valid when in LIN slave mode (LINS_EN(UA_LIN_CTL[0]) = 1). **Note2:** This bit is cleared by writing 1 to it. **Note3:** When user writing 1 to it, hardware will reload the initial baud-rate and re-search new frame header.
[2]	**LINS_IDPERR_F**	**LIN Slave ID Parity Error Flag**

		This bit is set by hardware when receipted frame ID parity is not correct. 0 = no active. 1 = Receipted frame ID parity is not correct. **Note1:** This bit is cleared by writing 1 to it. **Note2:** This bit is only valid when in LIN slave mode (LINS_EN(UA_LIN_CTL [0]) = 1) and enable LIN frame ID parity check function (LIN_IDPEN(UA_LIN_CTL [9]) = 1).
[1]	**LINS_HERR_F**	**LIN Slave Header Error Flag** This bit is set by hardware when a LIN header error is detected in LIN slave mode and be cleared by writing 1 to it. The header errors include “break delimiter is too short”, “frame error in sync field or Identifier field”, “sync field data is not 0x55 without automatic resynchronization mode”, “sync field deviation error with automatic resynchronization mode”, “sync field measure time-out with automatic resynchronization mode” and “LIN header reception time-out”. 0 = LIN header error not detected. 1 = LIN header error detected. **Note1:** This bit is cleared by writing 1 to it. **Note2:** This bit is only valid when in LIN slave mode (LINS_EN(UA_LIN_CTL [0]) = 1) and enable LIN slave header detection function (LINS_HDET_EN(UA_LIN_CTL [1]) = 1).
[0]	**LINS_HDET_F**	**LIN Slave Header Detection Flag** This bit is set by hardware when a LIN header is detected in LIN slave mode and be cleared by writing 1 to it. 0 = LIN header not detected. 1 = LIN header detected (break + sync + frame ID). **Note1:** This bit is cleared by writing 1 to it. **Note2:** This bit is only valid when in LIN slave mode (LINS_EN (UA_LIN_CTL [0]) = 1) and enable LIN slave header detection function (LINS_HDET_EN (UA_LIN_CTL [1])). **Note3:** When enable ID parity check (LIN_IDPEN (UA_LIN_CTL [9]) = 1), if hardware detect complete header (“break + sync + frame ID”), the LINS_HDET_F will be set weather the frame ID correct or not.

6.14 I²C Serial Interface Controller (I²C)

6.14.1 Overview

I²C is a two-wire, bi-directional serial bus that provides a simple and efficient method of data exchange between devices. The I²C standard is a true multi-master bus including collision detection and arbitration that prevents data corruption if two or more masters attempt to control the bus simultaneously.I²C controller supports Power-down wake-up function.

6.14.2 Features

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

- Master/Slave mode
- Bidirectional data transfer between masters and slaves
- Multi-master bus (no central master)
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus
- Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer
- A built-in a 14-bit timeout counter requested the I²C interrupt if the I²C bus hangs up and timer-out counter overflows.
- External pull-up resistors are needed for high output
- Programmable clocks allow versatile rate control
- Supports 7-bit addressing mode
- Supports multiple address recognition (four slave address with mask option)

6.14.3 Basic Configuration

The basic configurations of I²C are as follows:

- I²C0 pins are configured on P3_MFP[5:4] orP5_MFP[1:0] or PA_MFP[1:0] registers..
- I²C0 interrupt vector are configured on NVIC_ISER registers.
- Enable I²C0 clock (I2C_EN) on APBCLK [8] register.
- Reset I²C0 controller (I2C_RST) on IPRSTC2 [8] register.

6.14.4 Block Diagram

The block diagram of I²Ccontroller is shown below.

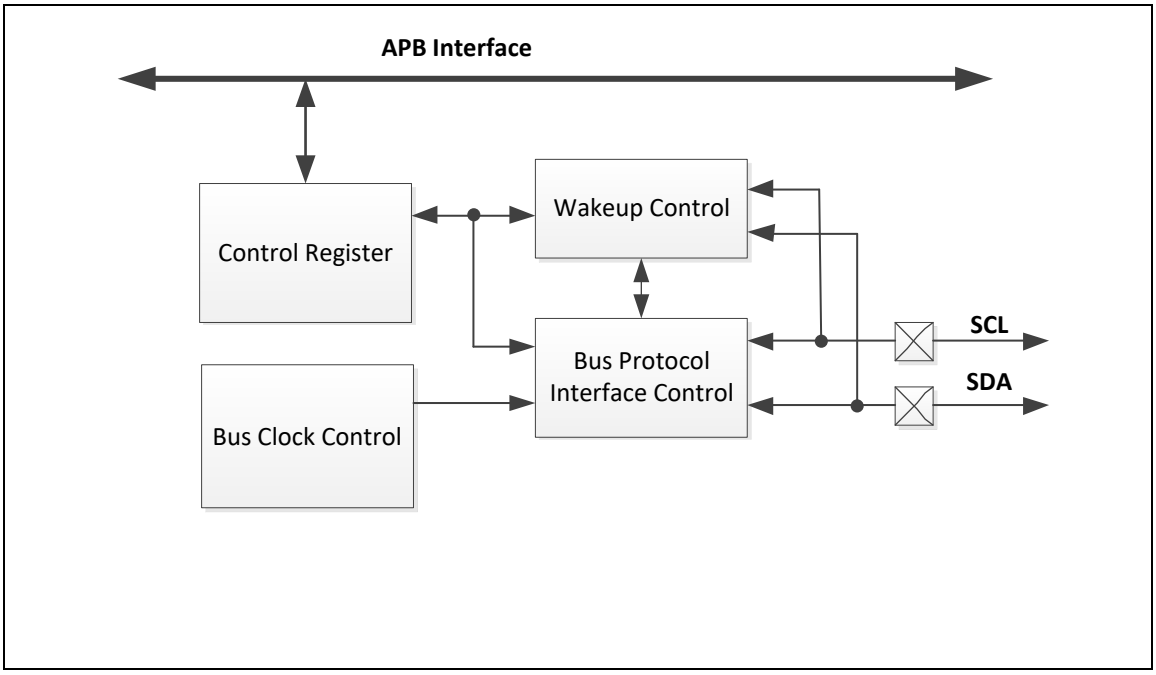


Figure 6-109I2C Controller Block Diagram

6.14.5 Functional Description

On I2C bus, data is transferred between a Master and a Slave. Data bits transfer on the SCL and SDA lines are synchronously on a byte-by-byte basis. Each data byte is 8-bit long. There is one SCL clock pulse for each data bit with the MSB being transmitted first, and an acknowledge bit follows each transferred byte. Each bit is sampled during the high period of SCL; therefore, the SDA line may be changed only during the low period of SCL and must be held stable during the high period of SCL. A transition on the SDA line while SCL is high is interpreted as a command (START or STOP). Please refer to the Figure 6-110 for more detailed I2C BUS Timing.

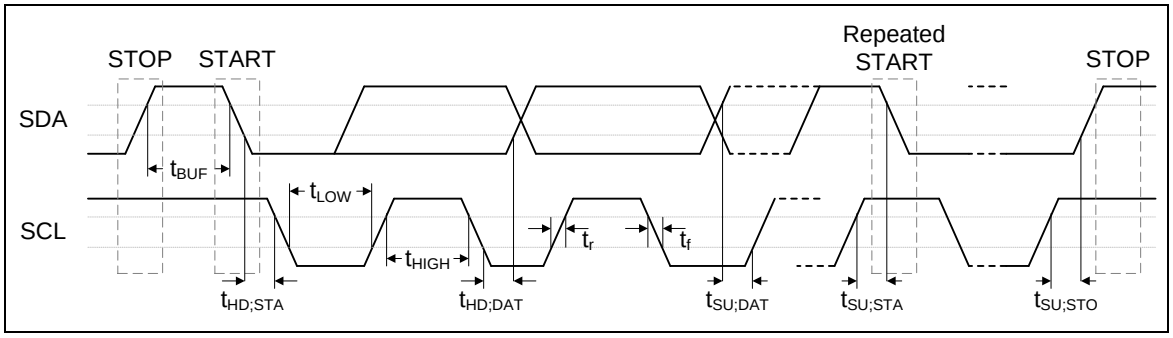


Figure 6-110 I²C Bus Timing

The device's on-chip I²C logic provides the serial interface that meets the I²C bus standard mode specification. The I²C port handles byte transfers autonomously. To enable this port, the bit ENS1 (I2CON[6]) should be set to '1'. The I²C H/W interfaces to the I²C bus via two pins: SDA and SCL. Pull-up resistor is needed for I²C operation as these SDA and SCL are open drain pins. When the I/O pins are used as I²C port, user must set the pins function to I²C in advance.

Note: Pull-up resistor is needed for I²C operation as the SDA and SCL are open-drain pins

6.14.5.1 I²C Protocol

The Figure 6-111shows the typical 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

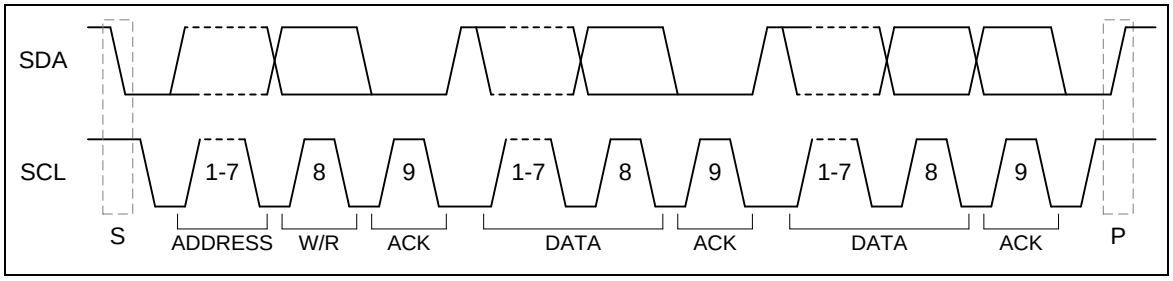


Figure 6-111 I²C Protocol

6.14.5.1.1 START or Repeated START signal

When the bus is free/idle, which means no master device is engaging the bus (both SCL and SDA lines are high), a master can initiate a transfer by sending a START signal. A START signal, usually referred to as the “S” bit, is defined as a HIGH to LOW transition on the SDA line while SCL is HIGH. The START signal denotes the beginning of a new data transmission.

After having sent the address byte (address and read/write bit), the master may send any number of bytes followed by a stop condition. Instead of sending the stop condition it is also allowed to send another start condition again followed by an address (and of course including a read/write bit) and more data. The start condition is called as Repeat START (Sr). This is defined recursively allowing any number of start conditions to be sent. The purpose of this is to allow combined write/read operations to one or more devices without releasing the bus and thus with the guarantee that the operation is not interrupted. The controller uses this method to communicate with another slave or the same slave in a different transfer direction (e.g. from writing to a device to reading from a device) without releasing the bus.

6.14.5.1.2 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.

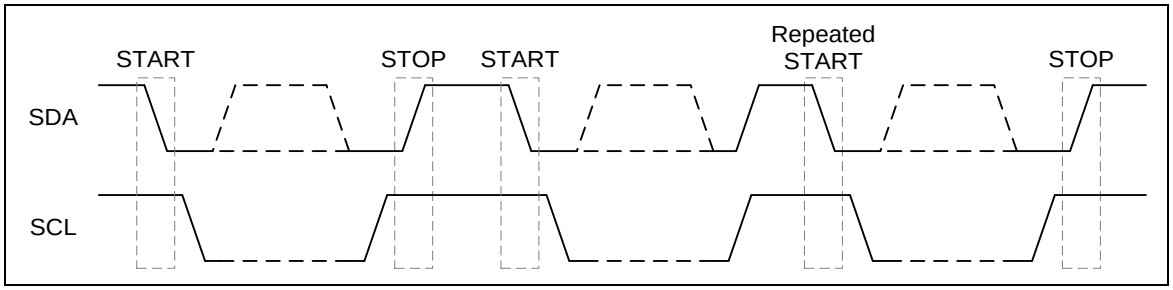


Figure 6-112 START and STOP Condition

6.14.5.1.3 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 Read/Write (R/W) bit. The R/W 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.

6.14.5.1.4 Data Transfer

When slave receives a correct address with a R/W bit, the data will follow R/W bit specified to transfer. Each transferred byte is followed by an acknowledge bit on the 9th SCL clock cycle. If the slave signals a Not Acknowledge (NACK), the master can generate a STOP signal to abort the data transfer or generate a Repeated START signal and start a new transfer cycle.

If the master, as 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.

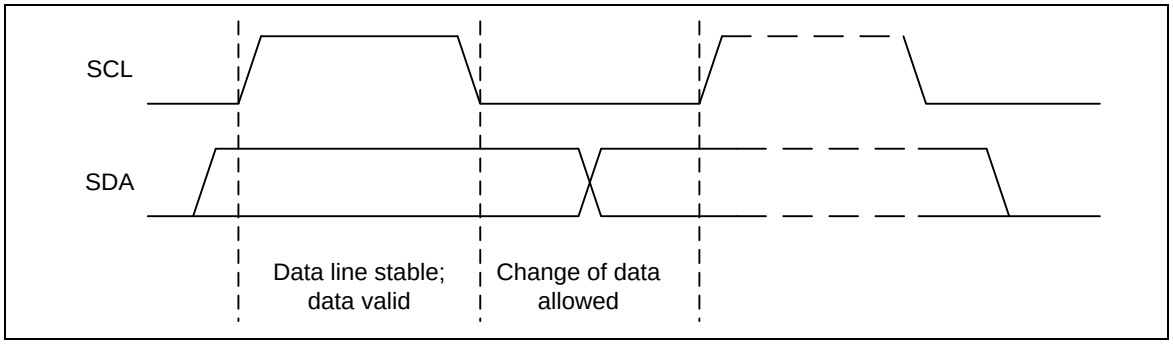


Figure 6-113 Bit Transfer on the I²C Bus

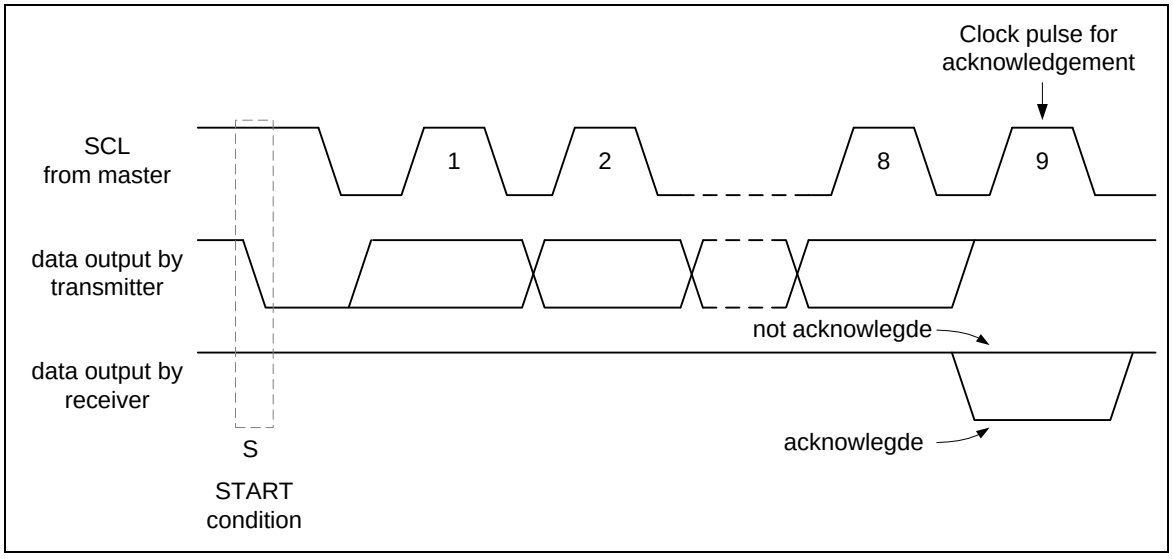


Figure 6-114 Acknowledge on the I²C Bus

6.14.5.1.5 Data transfer on the I²C-bus

Figure 6-115 shows a master transmits data to slave. A master addresses a slave with a 7-bit address and 1-bit write index to denote master wants to transmit data to slave. The master keep transmitting data after slave returns acknowledge to master.

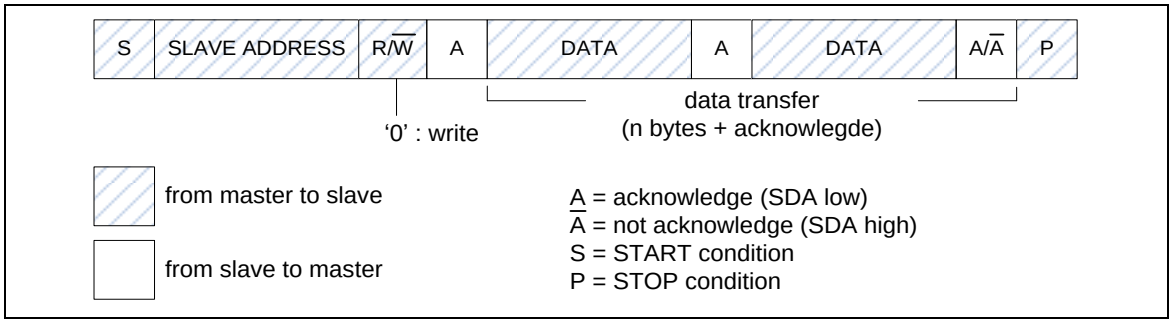


Figure 6-115 Master Transmits Data to Slave

Figure 6-116shows a master read data from slave. A master addresses a slave with a 7-bit address and 1-bit read index to denote master wants to read data from slave. The slavewill start transmitting data after slave returns acknowledge to master.

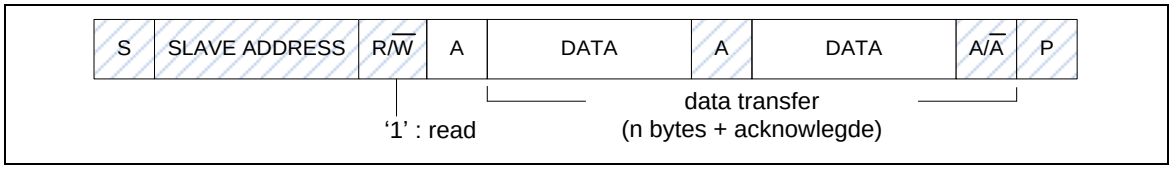


Figure 6-116 Master Reads Data from Slave

6.14.5.2 Operation Mode

The on-chip I²C ports support three operation modes, Master, Slave, and General Call Mode.

In a given application, I²C port may operate as a master or as a slave. In Slave mode, the I²C port hardware looks for its own slave address and the general call address. If one of these addresses is detected, and if the slave is willing to receive or transmit data from/to master(by setting the AA (I2CON[2]) bit), acknowledge pulse will be transmitted out on the 9thclock, hence an interrupt is requested on both master and slave devices if interrupt is enabled. When the microcontroller wishes to become the bus master, hardware waits until the bus is free before entering Master mode so that a possible slave action is not be interrupted. If bus arbitration is lost in Master mode, I²C port switches to Slave mode immediately and can detect its own slave address in the same serial transfer.

To control the I²C bus transfer in each mode, user needs to set I2CON, I2CDAT registers according to current status code of I2CSTATUS register. In other words, for each I²C bus action, user needs to check current status by I2CSTATUS register, and then set I2CON, I2CDAT registers to take bus action. Finally, check the response status by I2CSTATUS.

The bits, STA, STO (I2CON[5:4])and AA (I2CON[2]) are used to control the next state of the I²C hardware after SI (I2CON[3]) flag is cleared. Upon completion of the new action, a new status code will be updated in I2CSTATUS register and the SI flag (I2CON[3]) will be set. If the I²C interrupt control bit EI (I2CON [7]) is set, appropriate action or software branch of the new status code can be performed in the Interrupt service routine.

The Figure 6-117shows the current I²C status code is 0x08, and then set DATA=SLA+W and (STA,STO,SI,AA) = (0,0,1,x) to send the address to I²C bus. If a slave on the bus matches the address and response AA, the I2CSTATUS will be updated by status code 0x18.

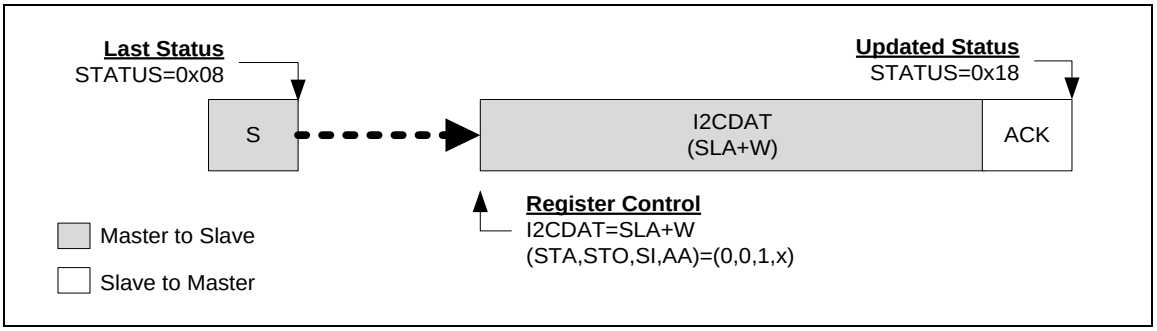


Figure 6-117 Control I2C Bus according to Current I²C Status

6.14.5.3 Master Mode

In theFigure 6-118 and Figure 6-119, all possible protocols for I²C master are shown. User needs to follow proper path of the flow to implement required I²C protocol.

In other words, user can send a START signal to bus and I²C will be in Master Transmitter modeor Master receiver modeafter START signal has been sent successfully and new status code would be 0x08. Followed by START signal, user can send slave address, read/write bit, data and Repeat START, STOP to perform I²C protocol.

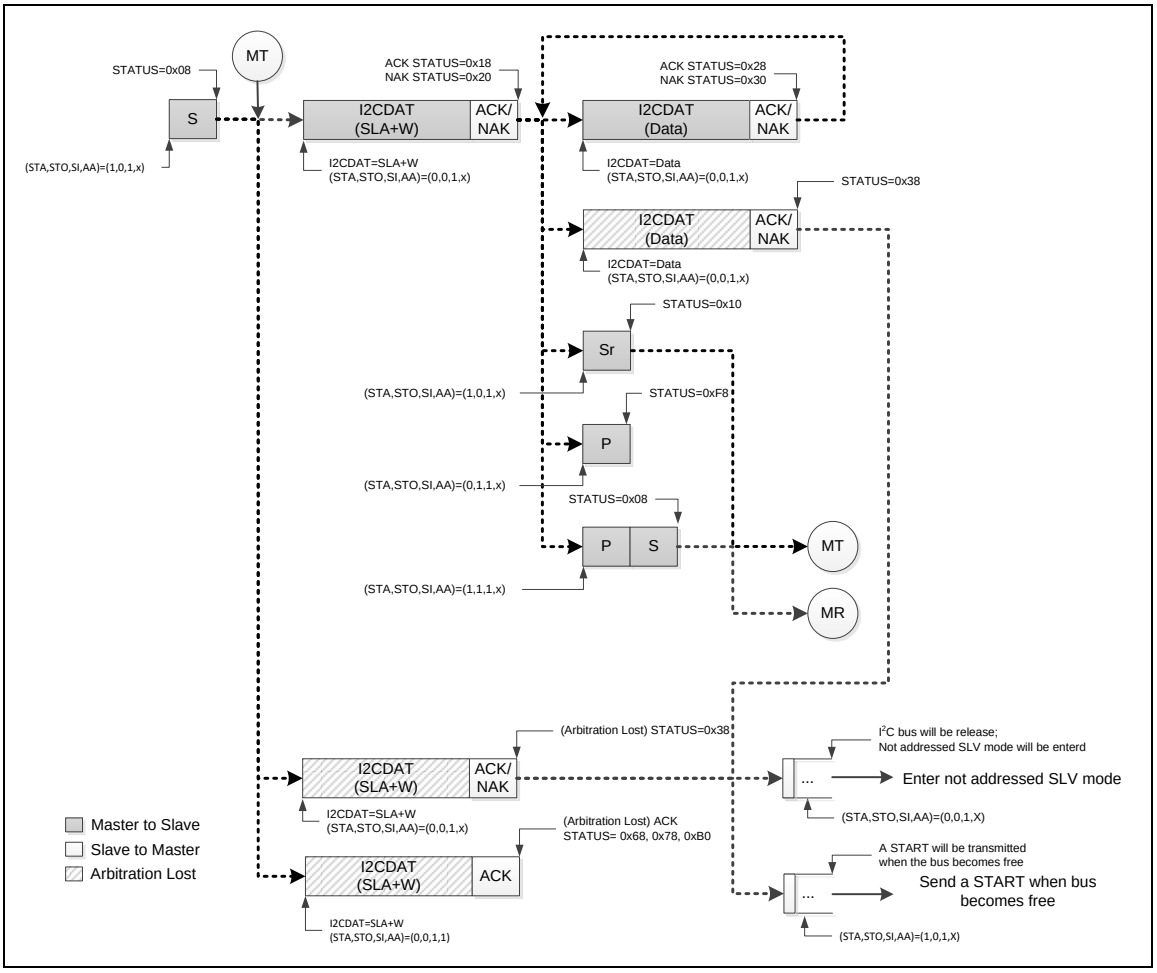


Figure 6-118Master Transmitter Mode Control Flow

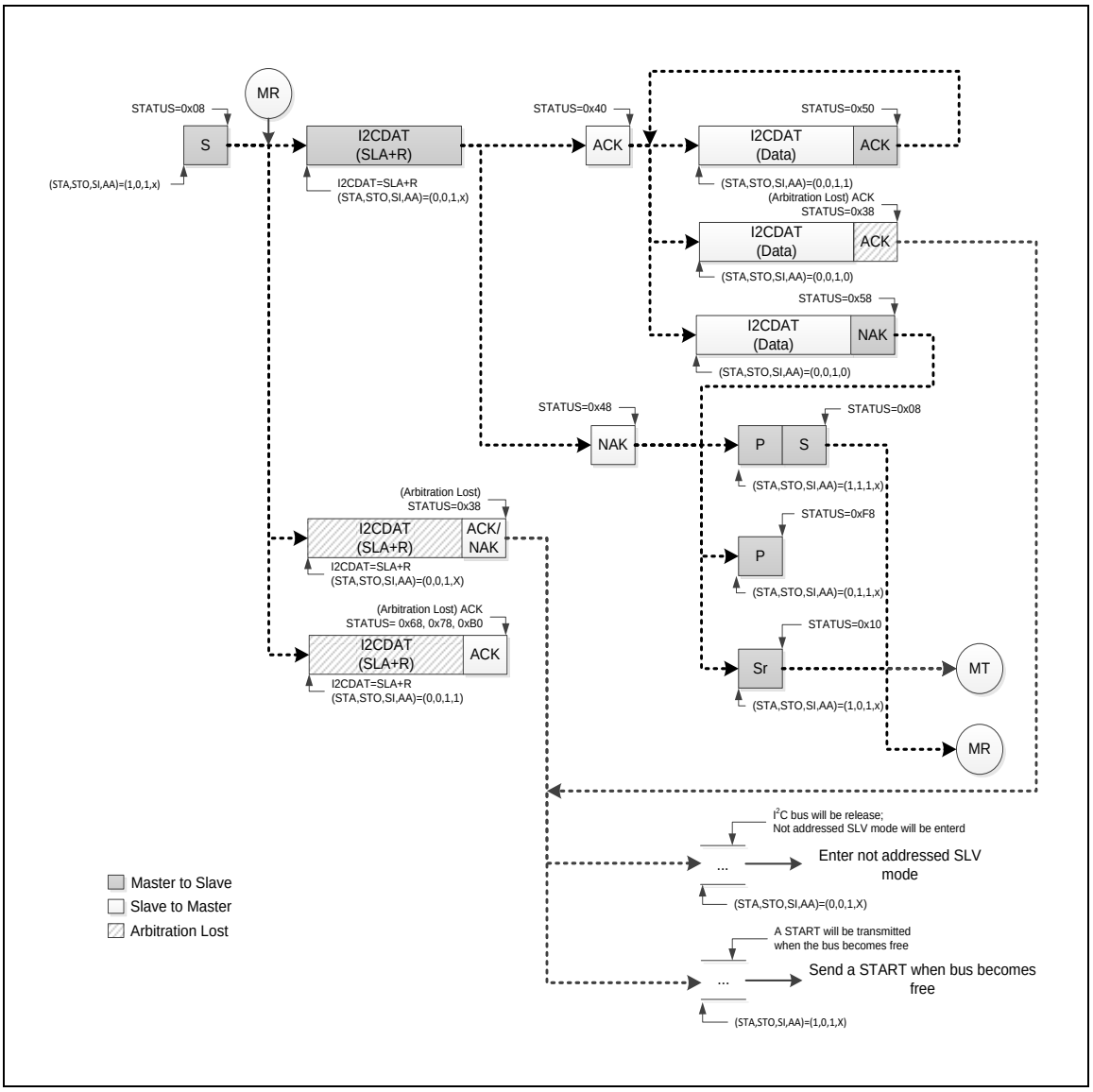


Figure 6-119Master Receiver Mode Control Flow

If the I²C is in Master mode and gets arbitration lost, the status code will be 0x38. In status 0x38, user may set (STA, STO, SI, AA) = (1, 0, 1, X) to send START to re-start Master operation when bus become free. Otherwise, user may set (STA, STO, SI, AA) = (0, 0, 1, X) to release I²C bus and enter not addressed Slave mode.

6.14.5.4 Slave Mode

When reset default, I²C is not addressed and will not recognize the address on I²C bus. User can set slave address by I2CADDRx and set (STA, STO, SI, AA) = (0, 0, 1, 1) to let I²C recognize the address sent by master. The follow figure shows all the possible flow for I²C in Slave mode. Users need to follow a proper flow to implement their own I²C protocol.

If bus arbitration is lost in Master mode, I²C port switches to Slave mode immediately and can detect its own slave address in the same serial transfer. If the detected address is SLA+W (Master want to write data to Slave) after arbitration lost, the status code is 0x68. If the detected address is SLA+R (Master want to read data from Slave) after arbitration lost, the status code is 0xB0.

Note: During I²C communication, the SCL clock will be released when writing '1' to clear SI flag in Slave mode.

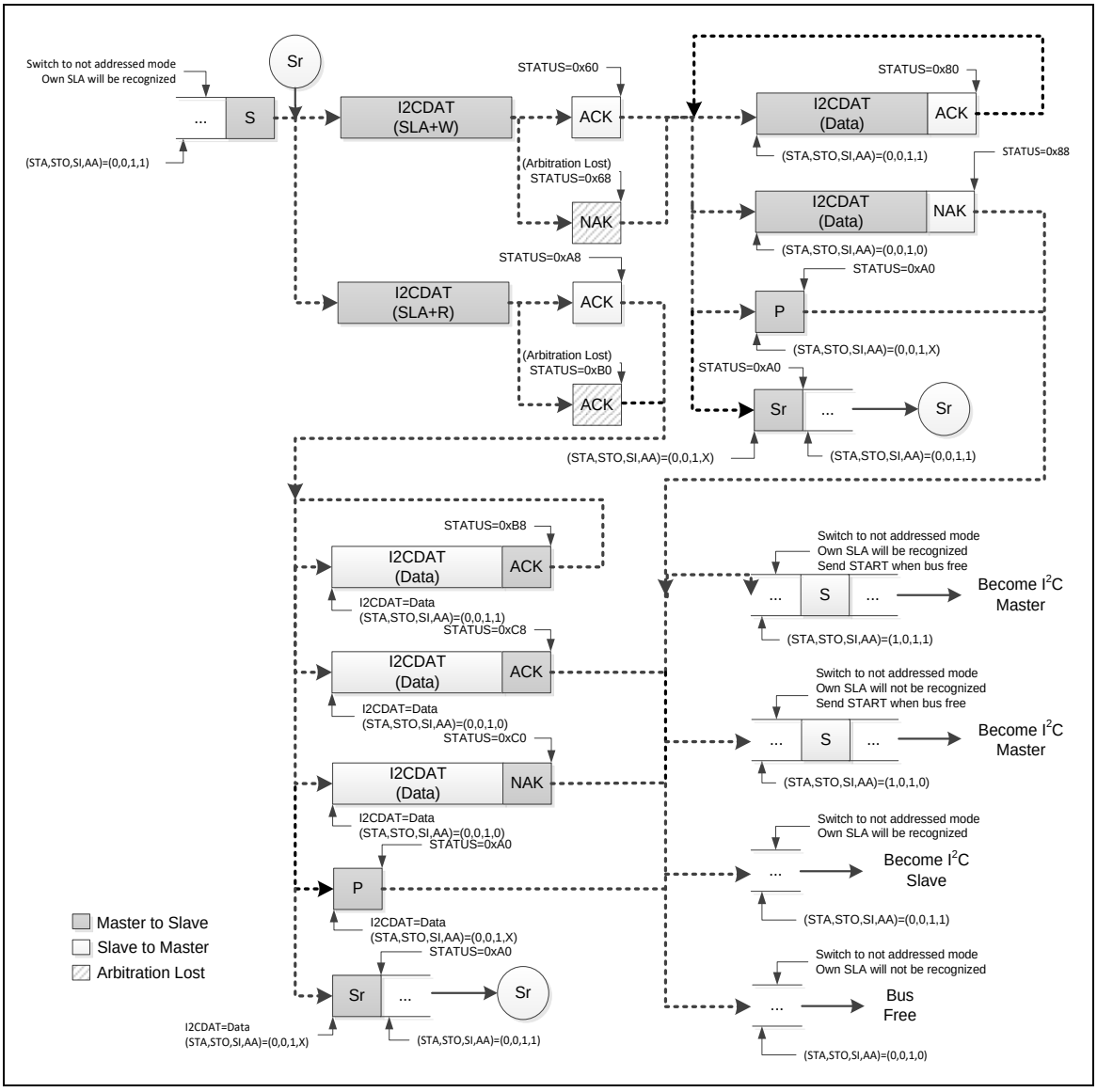


Figure 6-120Slave Mode Control Flow

If I²C is still receiving data in addressed Slave mode but got a STOP or Repeat START, the status code will be 0xA0. User could follow the action for status code 0x88 as shown in the above figure when getting 0xA0 status.

If I²C is still transmitting data in addressed Slave mode but got a STOP or Repeat START, the status code will be 0xA0. User could follow the action for status code 0xC8 as shown in the above figure when getting 0xA0 status.

Note: After slave gets status of 0x88, 0xC8, 0xC0 and 0xA0, slave can switch to not address mode and own SLA will not be recognized. If entering this status, slave will not receive any I²C signal or address from master. At this status, I²C should be reset to leave this status.

6.14.5.5 General Call (GC)Mode

If the GC (I2CADDR[0]) bit is set to 1, the I²C port hardware will respond to General Call address (00H). User can clear GC bit to disable general call function. When the GC bit is set and the I²C is in Slave mode, it can receive the general call address by 0x00 after master send general call

address to I²C bus, then it will follow status of GC mode.

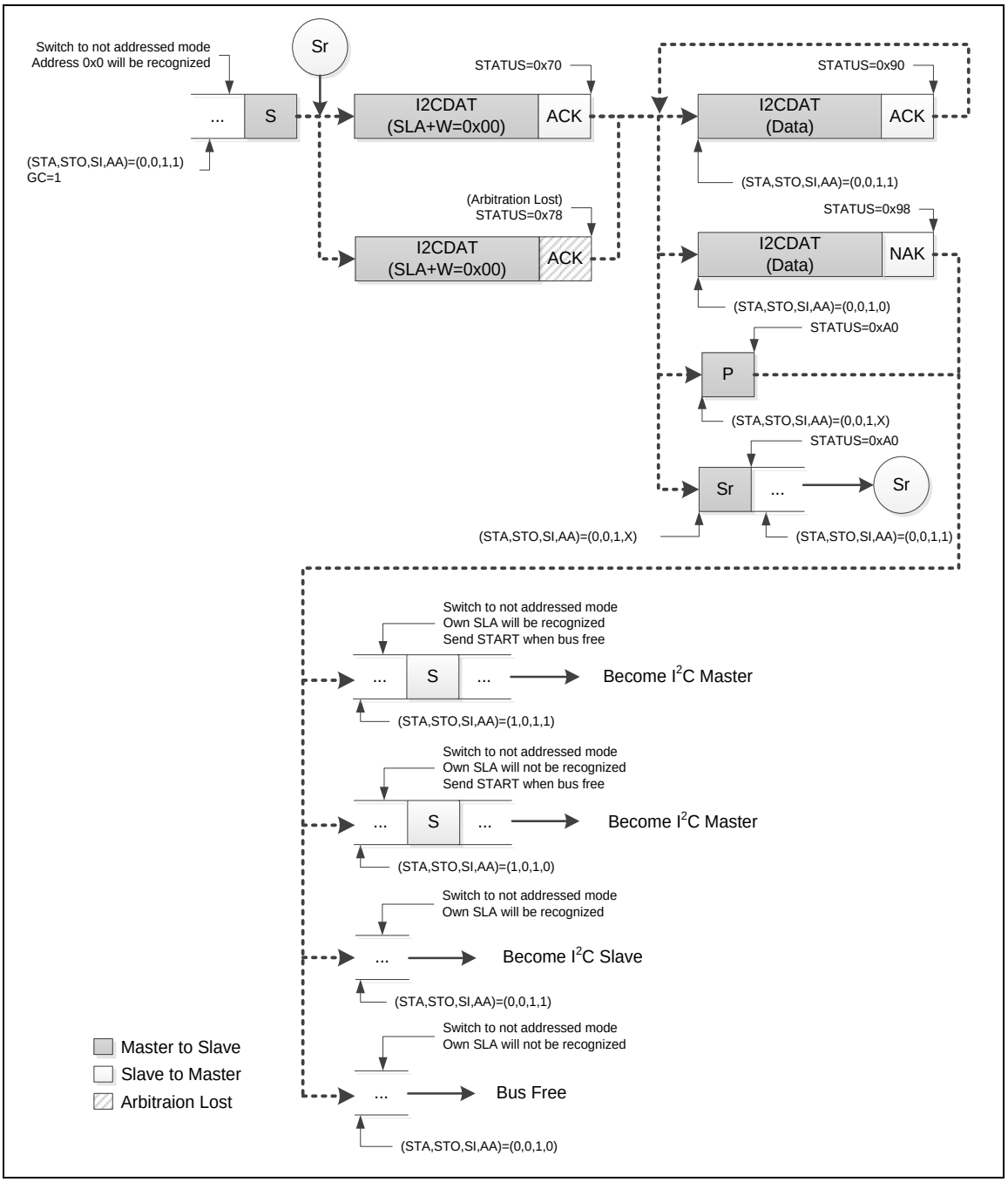


Figure 6-121GC Mode

If I²C is still receiving data in GC mode but got a STOP or Repeat START, the status code will be 0xA0. User could follow the action for status code 0x98 in the above figure when getting 0xA0 status.

Note: After slave gets status of 0x98 and 0xA0, slave can switch to not address mode and own SLA will not be recognized. If entering this status, slave will not receive any I²C signal or address from master. At this time, I²C controller should be reset to leave this status.

.

6.14.5.6 Multi-master

In some applications, there are two or more masters on the same I²C bus to access slaves, and the masters may transmit data simultaneously. The I²C supports multi-master by including collision detection and arbitration to prevent data corruption.

If for some reason two masters initiate command at the same time, the arbitration procedure determines which master wins and can continue with the command. Arbitration is performed on the SDA signal while the SCL signal is high. Each master checks if the SDA signal on the bus corresponds to the generated SDA signal. If the SDA signal on the bus is low but it should be high, then this master has lost arbitration. The device that has lost arbitration can generate SCL pulses until the byte ends and must then release the bus and go into slave mode. The arbitration procedure can continue until all the data is transferred. This means that in multi-master system each master must monitor the bus for collisions and act accordingly.

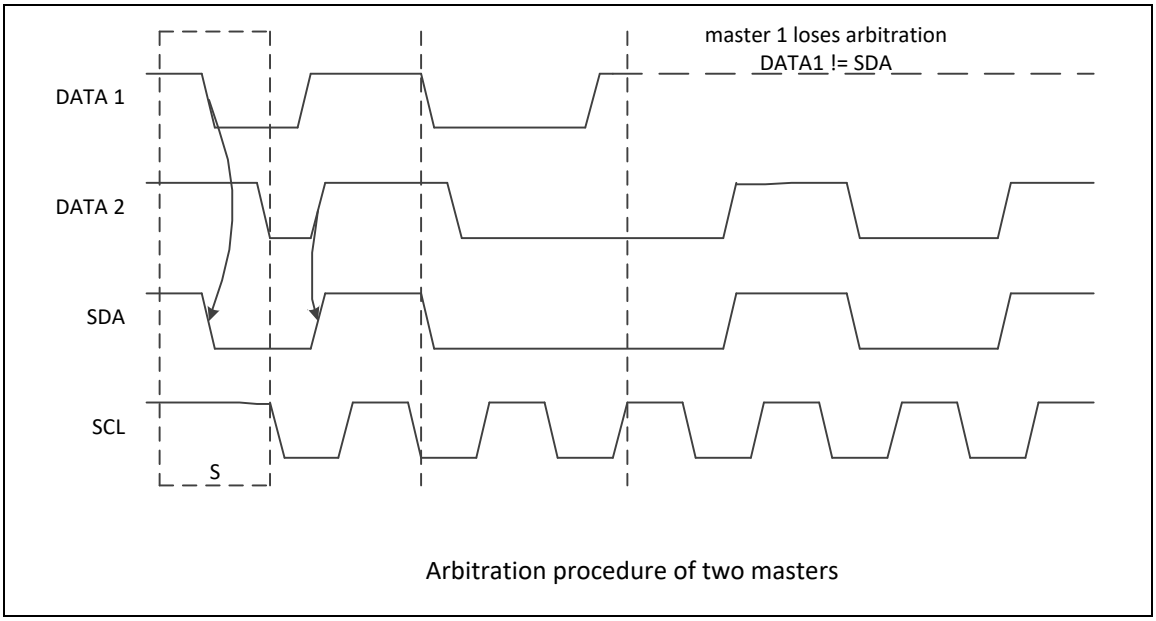


Figure 6-122Arbitration Lost

- When I2CSTATUS = 0x38, an “Arbitration Lost” is received. Arbitration lost event maybe occur during the send START bit, data bits or STOP bit. User could set (STA, STO, SI, AA) = (1, 0, 1, X) to send START again when bus free, or set (STA, STO, SI, AA) = (0, 0, 1, X) to send STOP to back to not addressed Slave mode.
- When I2CSTATUS = 0x00, a “Bus Error” is received. To recover I²C bus from a bus error, STO should be set and SI should be cleared, and then STO is cleared to release bus.
  - Set (STA, STO, SI, AA) = (0, 1, 1, X) to stop current transfer

- Set (STA, STO, SI, AA) = (0, 0, 1, X) to release bus

6.14.6 Protocol Registers

To control I²C port through the following fifteen special function registers: I2CCON (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), I2CTOC (Time-out control register), I2CWKUPCON(wake up control register), I2CWKUPSTS(wake up status register).

6.14.6.1 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 send general call address to I²C bus, 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 bitdoesn'tcare. If the bit is set to 0, that means the received corresponding register bit should be exact the same as address register.

6.14.6.2 Slave Address Mask Register (I2CADM)

The 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 1 it means the received corresponding address bit is "Don't-care". If the bit is set to 0, that means the received corresponding register bit should be exact the same as address register.

6.14.6.3 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. 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.

The acknowledge bit is controlled by the I²C hardware and cannot be accessed by the CPU. Serial data is shifted 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. In order to monitor bus status while sending data, the bus date will be shifted into I2CDATA[7:0] when sending I2CDATA[7:0] to bus. In case of sending data, serial data bits are shifted out from I2CDAT [7:0] on the falling edge of SCL clocks, and is shifted into I2CDAT [7:0] on the rising edge of SCL clocks.

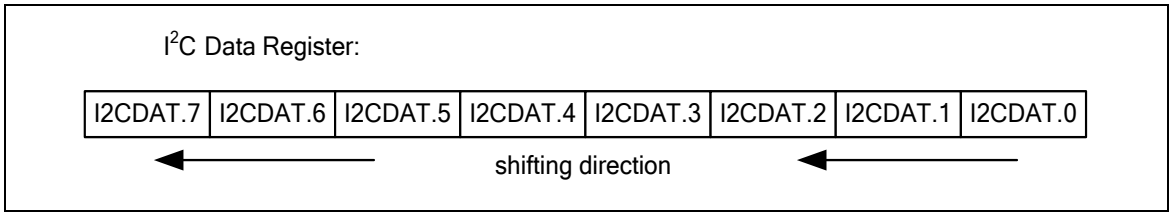


Figure 6-123 I²C Data Shifting Direction

6.14.6.4 Control Register (I2CON)

The CPU can read from and write to I2CON [7:0] directly. Two bits are affected by hardware: the SI bit is set when the I²C 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.

6.14.6.5 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 [7:0] contain the status code andthere are 26 possible status codes. All states are listed in Table 6-17. When I2CSTATUS [7:0] is F8H, no serial interrupt is requested. All other I2CSTATUS [7:0] values correspond to defined I²C states. When each of these states is entered, a status interrupt is requested (SI = 1). A valid status code is present in I2CSTATUS[7:0] one cycle after SI is set by hardware and is still present one cycle after SI has been reset by software.

In addition, the state 00H stands for a Bus Error, which occurs when a START or STOP condition is present at an illegal position in the format frame. A Bus Error may occur during the serial transfer of an address byte, a data byte or an acknowledge bit. To recover I²C from bus error, STO should be set and SI should be clear to enter Not Addressed Slave mode. Then STO is cleared to release bus and to wait new communication. I²C bus cannot recognize stop condition during this action when bus error occurs.

Master Mode		**Slave Mode**	
STATUS	**Description**	**STATUS**	**Description**
0x08	Start	0xA0	Slave Transmit Repeat Start or Stop
0x10	Master Repeat Start	0xA8	Slave Transmit Address ACK
0x18	Master Transmit Address ACK	0xB0	Slave Transmit Arbitration Lost
0x20	Master Transmit Address NACK	0xB8	Slave Transmit Data ACK
0x28	Master Transmit Data ACK	0xC0	Slave Transmit Data NACK
0x30	Master Transmit Data NACK	0xC8	Slave Transmit Last Data ACK
0x38	Master Arbitration Lost	0x60	Slave Receive Address ACK
0x40	Master Receive ACK	0x68	Slave Receive Arbitration Lost
0x48	Master Receive NACK	0x80	Slave Receive Data ACK
0x50	Master Receive ACK	0x88	Slave Receive Data NACK
0x58	Master Receive NACK	0x70	GC mode Address ACK
0x00	Bus error	0x78	GC mode Arbitration Lost
		0x90	GC mode Data ACK

		0x98	GC mode Data NACK
0xF8	Bus Released **Note:** Status “0xF8” exists in both Master/Slave modes, and it won’t raise interrupt.		

Table 6-23 I²C Status Code Description Table

6.14.6.6 I²C Clock Baud Rate Bits (I2CLK)

The data baud rate of I²C is determined by I2CLK [7:0] register when I²C is in Master mode, andit is not necessary when I²C is in Slave mode. In the Slave mode, I²C will automatically synchronize it with any clock frequency from master I²C device.

The data baud rate of I²C setting is Data Baud Rate of I²C = (system clock) / (4x (I2CLK [7:0] +1)). If system clock = 16 MHz, the I2CLK [7:0] = 40 (28H), so data baud rate of I²C = 16 MHz/ (4x (40 +1)) = 97.5 Kbits/sec.

6.14.6.7 The I²C Time-out Counter Register (I2CTOC)

There is a 14-bit time-out counter which can be used to deal with the I2C bus hang-up. If the time out counter is enabled, the counter starts up counting until it overflows (TIF(I2CTOC[0])=1) and generates I2C interrupt to CPU or stops counting by clearing ENTI(I2CTOC[2]) to 0. When time-out counter is enabled, write 1 to SI flag will reset counter and re-start up counting after SI is cleared. If I2C bus hangs up, it causes the I2CSTATUS and flag SI(I2CON[3]) are not updated for a period, the 14-bit time-out counter may overflow and acknowledge CPU the I2C interrupt. Refer to the Figure 6-124 for the 14-bit time out counter. User may write 1 to clear TIF(I2CTOC[0]) to 0.

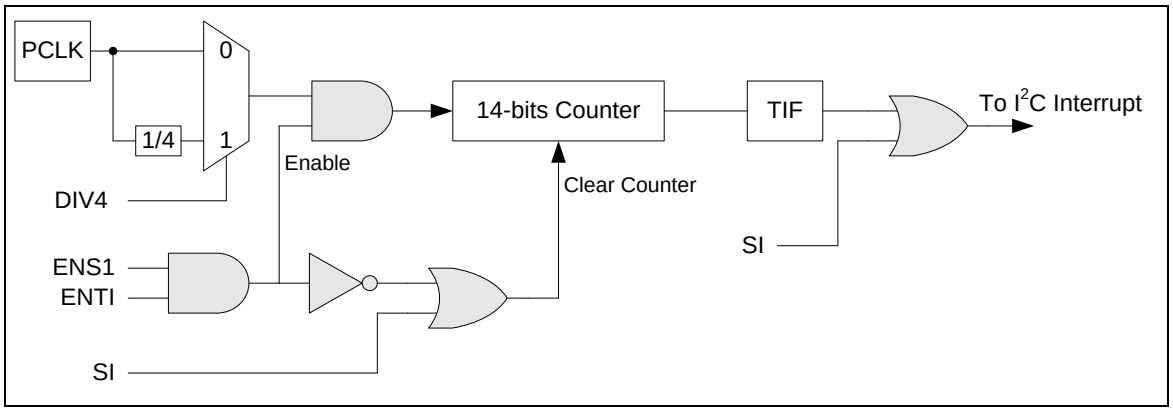


Figure 6-124 I²C Time out Count Block Diagram

6.14.6.8 The I²C wake-up control Register (I2CWKUPCON)

When chip enters Power-down mode, other I²C master can wake up our chip by addressing our I²C device, user must configure the related setting before entering sleep mode.When the chip is woken-up by address match with one of the four address register, the following data will be abandoned at this time.

6.14.6.9 The I²C wake-up status Register (I2CWKUPSTS)

When system is woken up by other I²C master device, WKUPIF(I2CWKUPSTS[0]) is set to indicate this event. User needs write “1” to clear this bit.

6.14.6.10 Example for Random Read on EEPROM

The following steps are used to configure the I²C0 related registers when using I²C to read data from EEPROM.

1. Set I²C0 the multi-function pin in the P3_MFP[5:4] registers as SCL and SDA pins.
2. Enable I²C0 APB clock, I2C0_EN=1 in the “APBCLK0” register.

3. Set I2C0_RST=1 to reset I²C0 controller then set I²C0 controller to normal operation, I2C0_RST=0 in the “IPRSTC2” register.
4. Set EI=1 to enable I²C0 controller in the “I2CON” register.
5. Give I²C0 clock a divided register value for I²C clock rate in the “I2CLK”.
6. Set SETENA=0x00000040 in the “NVIC_ISER” register to set I²C0 IRQ.
7. Set EI=1 to enable I²C0 Interrupt in the “I2CON” register.
8. Set I²C0 address registers “I2CADDR0~I2CADDR3”.

Random read operation is one of the methods of access EEPROM. The method allows the master to access any address of EEPROM space. The Figure 6-125shows the EEPROM random read operation.

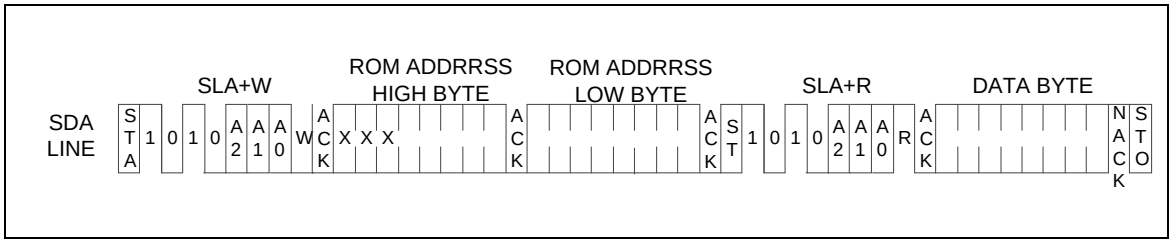


Figure 6-125EEPROMRandom Read

The Figure 6-126shows how to use I²C controller to implement the protocol of EEPROM random read.

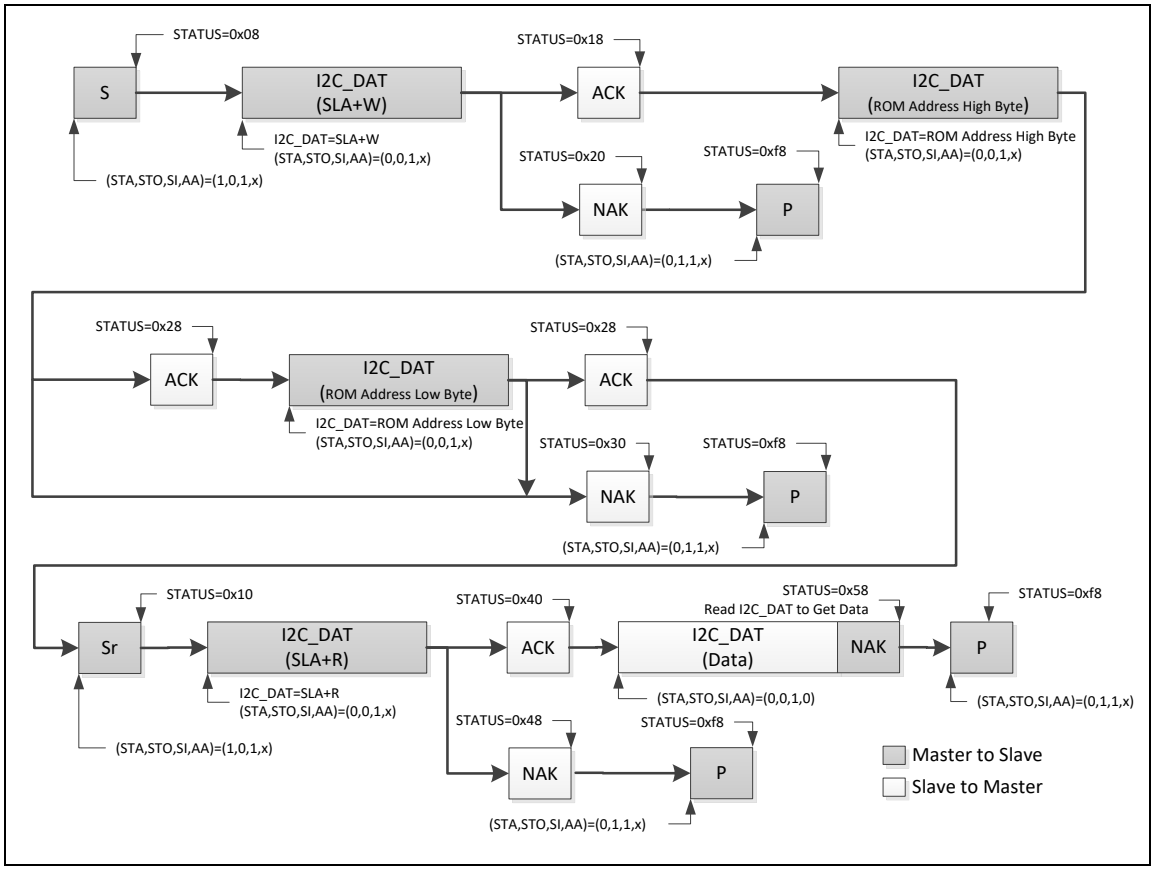


Figure 6-126Protocol of EEPROM Random Read

The I²C controller, which is a master, sends START to bus. Then, it sends a SLA+W (Slave address +

Write bit) to EERPOM followed by two bytes data address to set the EEPROM address to read. Finally, a Repeat START followed by SLA+R is sent to read the data from EEPROM.

6.14.7 Register Map

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

Register	Offset	R/W	Description	Reset Value
I2C Base Address: **I2C0_BA = 0x4002_0000**				
I2CON	I2Cx_BA+0x00	R/W	I²C Control Register	0x0000_0000
I2CADDR0	I2Cx_BA+0x04	R/W	I²C Slave Address Register0	0x0000_0000
I2CDAT	I2Cx_BA+0x08	R/W	I²C Data Register	0x0000_0000
I2CSTATUS	I2Cx_BA+0x0C	R	I²C Status Register	0x0000_00F8
I2CLK	I2Cx_BA+0x10	R/W	I²C Clock Divided Register	0x0000_0000
I2CTOC	I2Cx_BA+0x14	R/W	I²C Time Out Counter Register	0x0000_0000
I2CADDR1	I2Cx_BA+0x18	R/W	I²C Slave Address Register1	0x0000_0000
I2CADDR2	I2Cx_BA+0x1C	R/W	I²C Slave Address Register2	0x0000_0000
I2CADDR3	I2Cx_BA+0x20	R/W	I²C Slave Address Register3	0x0000_0000
I2CADM0	I2Cx_BA+0x24	R/W	I²C Slave Address Mask Register0	0x0000_0000
I2CADM1	I2Cx_BA+0x28	R/W	I²C Slave Address Mask Register1	0x0000_0000
I2CADM2	I2Cx_BA+0x2C	R/W	I²C Slave Address Mask Register2	0x0000_0000
I2CADM3	I2Cx_BA+0x30	R/W	I²C Slave Address Mask Register3	0x0000_0000
I2CWKUPCON	I2Cx_BA+0x3C	R/W	I²C Wake Up Control Register	0x0000_0000
I2CWKUPSTS	I2Cx_BA+0x40	R/W	I²C Wake Up Status Register	0x0000_0000

6.14.8 Register Description

I²C Control Register (I2CON)

Register	Offset	R/W	Description	Reset Value
I2CON	I2Cx_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** 0 = I²C interruptDisabled. 1 = I²C interruptEnabled.
[6]	**ENS1**	**I²C Controller Enable Bit** Set to enable I²C serial function controller. When ENS1 =1 the I²C serial function enables. The multi-function pin function must set to I²C function first. 0 = I²C controller Disabled. 1 = I²C controller Enabled.
[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 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	I2Cx_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							

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.

I²C Status Register (I2CSTATUS)

Register	Offset	R/W	Description	Reset Value
I2CSTATUS	I2Cx_BA+0x0C	R	I²C Status Register	0x0000_00F8

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
I2CSTATUS							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**I2CSTATUS**	**I²C Status Register** 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 I²C 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.

I²C Clock Divided Register (I2CLK)

Register	Offset	R/W	Description	Reset Value
I2CLK	I2Cx_BA+0x10	R/W	I²C Clock Divided Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
I2CLK							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**I2CLK**	**I²C Clock Divided Register** The I²C clock rate bits: Data Baud Rate of I²C = (system clock) / (4x (I2CLK+1)). **Note:**The minimum value of I2CLK is 4.

I²C Time-out Counter Register (I2CTOC)

Register	Offset	R/W	Description	Reset Value
I2CTOC	I2Cx_BA+0x14	R/W	I²C 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 Bit** When Enabled, the 14-bit time out counter will start counting when SI is clear. Write 1 to SI flag will reset counter and re-start up counting after SI is cleared. 0 = Time out counter Disabled. 1 = Time out counter Enabled.
[1]	**DIV4**	**Time-out Counter Input Clock Is Divided by 4** When Enabled, the time out period is extend 4 times. 0 = The time out counter input clock divided by 4 Disabled. 1 = The time out counter input clock divided by 4 Enabled.
[0]	**TIF**	**Time-out Flag** This bit is set by H/W when I²C time out happened and it can interrupt CPU if I²C interrupt enable bit (EI) is set to 1. Software can write 1 to clear this bit.

I²C Slave Address Register (I2CADDRx)

Register	Offset	R/W	Description	Reset Value
I2CADDR0	I2Cx_BA+0x04	R/W	I²C Slave Address Register0	0x0000_0000
I2CADDR1	I2Cx_BA+0x18	R/W	I²C Slave Address Register1	0x0000_0000
I2CADDR2	I2Cx_BA+0x1C	R/W	I²C Slave Address Register2	0x0000_0000
I2CADDR3	I2Cx_BA+0x20	R/W	I²C Slave Address Register3	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
I2CADDR							**GC**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:1]	**I2CADDR**	**I²C Address Register** The content of this register is irrelevant when I²C is in Master mode. In Slave mode, the seven most significant bits must be loaded with the chip's own address. The I²C hardware will react if one of the addresses is matched.
[0]	**GC**	**General Call Function** 0 = General Call functionDisabled. 1 = General Call functionEnabled.

I²C Slave Address Mask Register (I2CADMx)

Register	Offset	R/W	Description	Reset Value
I2CADM0	I2Cx_BA+0x24	R/W	I²C Slave Address Mask Register0	0x0000_0000
I2CADM1	I2Cx_BA+0x28	R/W	I²C Slave Address Mask Register1	0x0000_0000
I2CADM2	I2Cx_BA+0x2C	R/W	I²C Slave Address Mask Register2	0x0000_0000
I2CADM3	I2Cx_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							**Reserved**

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:1]	**I2CADM**	**I²C Address Mask Register** I²C bus controller supports multiple address recognition with four address mask registers. 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 0, that means the received corresponding register bit should be exact the same as address register. 0 = Mask Disabled (the received corresponding register bit should be exact the same as address register.). 1 = Mask Enabled (the received corresponding address bit is don't care.).
[0]	**Reserved**	Reserved.

I²C Wake Up Control Register (I2CWKUPCON)

Register	Offset	R/W	Description	Reset Value
I2CWKUPCON	I2Cx_BA+0x3C	R/W	I²C Wake Up Control Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**WKUPEN**	**I²C Wake-up Function Enable Bit** 0 = I²C wake up function Disabled. 1 = I²C wake up function Enabled.

I²C Wake-up Status Register (I2CWKUPSTS)

Register	Offset	R/W	Description	Reset Value
I2CWKUPSTS	I2Cx_BA+0x40	R/W	I²C Wake-up Status Register	0x0000_0000

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

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**WKUPIF**	**I²C Wake-up Interrupt Flag** When chip is woken-up from Power-down mode by I²C, this bit is set to 1. Software can write 1 to clear this bit

6.15 Serial Peripheral Interface (SPI)

6.15.1 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. The NuMicro®NM1530seriescontainsup to threesets of SPI controllers performing a serial-to-parallel conversion on data received from a peripheral device, and a parallel-to-serial conversion on data transmittedto a peripheral device. Each set of SPI controller can be configured as a master ora slave device.

6.15.2 Features

- Up to three sets of SPI controllers
- Supports Master or Slave mode operation
- Configurable bit length of a transaction word from 8 to 32-bit
- Provides separate 8-layer depth transmit and receive FIFO buffers
- Supports MSB first or LSB first transfer sequence
- Supports the Byte Reorder function
- Supports 3-wire, no slave select signal, bi-direction interface

6.15.3 Block Diagram

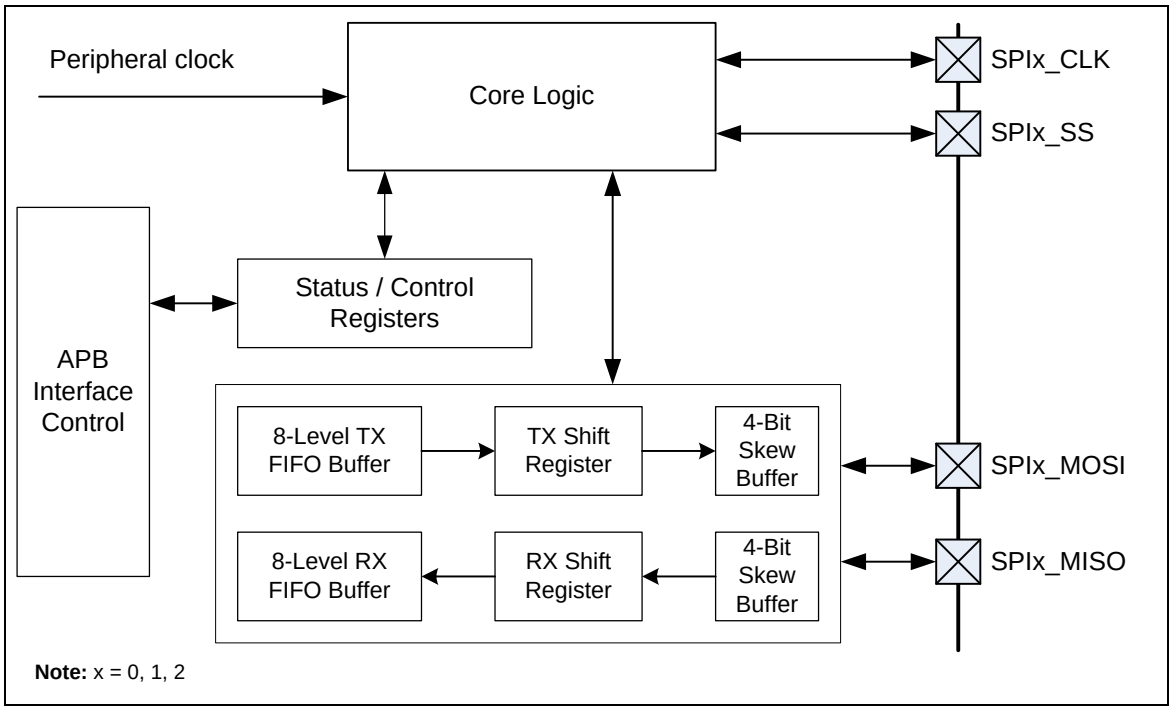


Figure 6-127 SPI Block Diagram

TX FIFO Buffer:

The transmit FIFO buffer is an 8-level depth, 32-bit wide, first-in, first-out register buffer. The data can be written to the transmit FIFO buffer in advance through software by writing the SPI_TX register.

RX FIFO Buffer:

The received FIFO buffer is also an 8-level depth, 32-bit wide, first-in, first-out register buffer. The receive control logic will store the received data to this buffer. The FIFO buffer data can be read from SPI_RX register by software.

TX Shift Register:

The transmit shift register is a 32-bit wide register buffer. The transmit data is loaded from the TX FIFO buffer and shifted out bit-by-bit to the skew buffer.

RX Shift Register:

The receive shift register is also a 32-bit wide register buffer. The receive data is shifted in bit-by-bit from the skew buffer and is loaded into RX FIFO buffer when a transaction done.

Skew Buffer:

The skew buffer is a 4-bit buffer.

For transmitting, it is written from shift register by peripheral clock and read out by SPI bus clock. The first three bitsof TX shift register will be loaded to skew buffer in the beginning of transmission.The remainingbitswill be shifted into skew buffer asany bit in skew buffer is shifted out to SPI bus.

For receiving, the serial data captured fromSPI bus is written into the skew buffer basing on the SPI bus clock.These captured data will be read out and written into the RX shift register basingon the SPI peripheral clock.

6.15.4 Basic Configuration

The basic configurations of SPI0 are as follows:

- SPI0 pin functions are configured in P2_MFPand P5_MFP registers.
- Select the source of SPI0 peripheral clock on SPI0_S (CLKSEL1[4]).
- Enable SPI0peripheral clock on SPI0_EN (APBCLK[12]).
- Reset SPI0 controller on SPI0_RST (IPRSTC2[12]).

The basic configurations of SPI1 are as follows:

- SPI1 pin functions are configured in P9_MFPregister.
- Select the source of SPI1 peripheral clock on SPI1_S (CLKSEL1[5]).
- Enable SPI1peripheral clock on SPI1_EN (APBCLK[13]).
- Reset SPI1 controller on SPI1_RST (IPRSTC2[13]).

The basic configurations of SPI2 are as follows:

- SPI2 pin functions are configured in P2_MFPand P5_MFP registers.
- Select the source of SPI2 peripheral clock on SPI2_S (CLKSEL1[6]).
- Enable SPI2peripheral clock on SPI2_EN (APBCLK[14]).
- Reset SPI2 controller on SPI2_RST (IPRSTC2[14]).

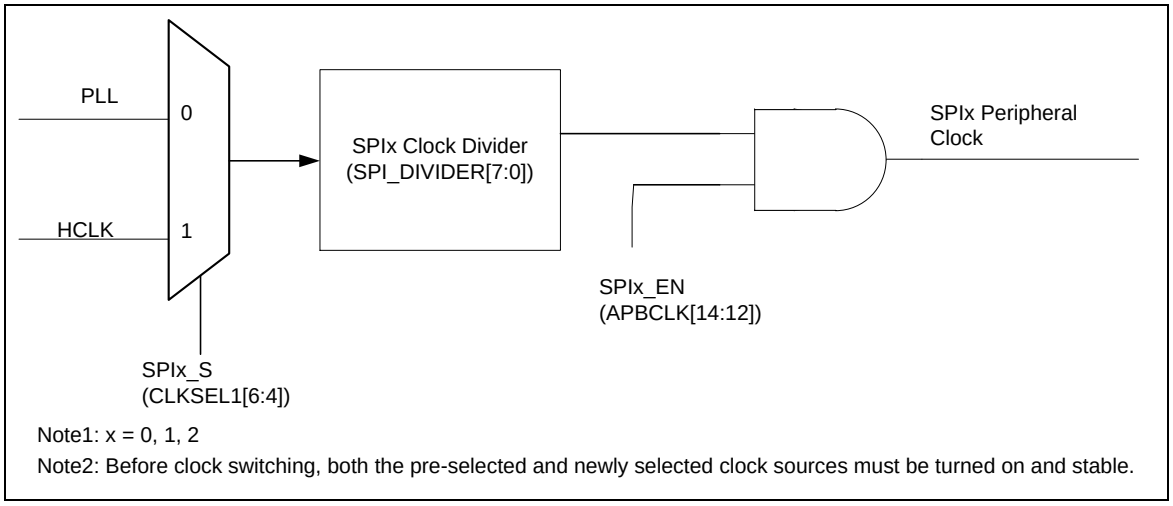
6.15.5 Functional Description

6.15.5.1 Terminology

SPI Peripheral Clock and SPI Serial Clock

The SPI controller needs the SPI peripheral clock to drive the SPI logic unit to perform the data transfer. The SPI peripheral clock rate is determined by the settings of clock source and clock divisor. The SPIx_SS bit of CLKSEL1 register determines the clock source of the SPI peripheral clock. The clock source can be HCLK or PLL output clock. The DIVIDER setting of SPI_DIVIDER register determines the divisor of the clock rate calculation.

In Master mode, the output frequency of the SPI bus clock output pin is equal to the SPI peripheral clock rate. In general, the SPI bus clock denotes as SPI clock. In Slave mode, the SPI bus clock is provided by an off-chip master device. The SPI peripheral clock rate of slave device must be faster than the SPI bus clock rate of the master device connected together. The frequency of SPI peripheral clock cannot be faster than the APB clock rate regardless of Master or Slave mode.



Note1: x = 0, 1, 2

Note2: Before clock switching, both the pre-selected and newly selected clock sources must be turned on and stable.

Figure 6-128SPI Clock Control Diagram

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.

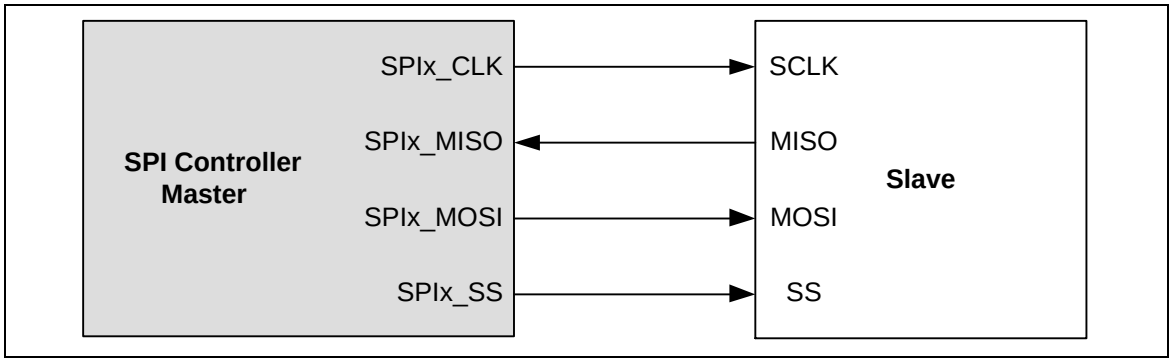


Figure 6-129SPI Master Mode Application Block Diagram

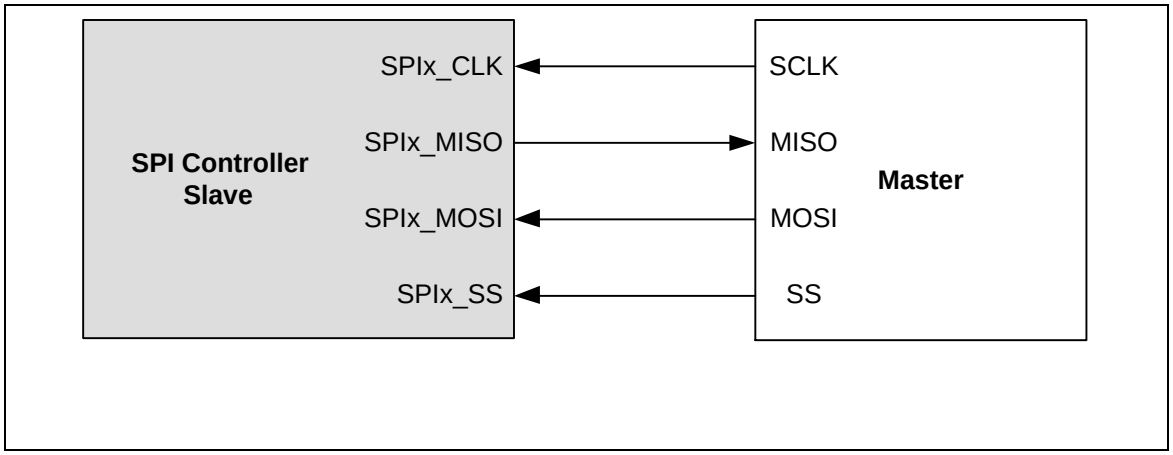


Figure 6-130SPI Slave Mode Application Block Diagram

Slave Selection

In Master mode, this SPI controller can drive off-chip Slave device through the slave select output pin SPIx_SS. In Slave mode, the off-chip Master device drives the slave select signal from the SPIx_SS input port to this SPI controller. The duration between the slave select active edge and the first SPI clock input shall over 3 SPI peripheral clock cycles of Slave.

In Master/Slave mode, the active state of slave select signal can be programmed to low active or high active in SS_LVL (SPI_SSR[2]). The selection of slave select condition depends on what type of device is connected.

In Slave mode, the SS_LTRIG (SPI_SSR[4]) defines the slave select signal SPIx_SS is level trigger or edge trigger.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(SPI_CNTRL[7:3]). To recognize the inactive state of the slave selection signal, the inactive period of the slave selectionsignal must be larger than or equal to 3peripheral clock cycles between two successive transactions.

Level-trigger/Edge-trigger

In Slave mode, the slave select signal can be configured as level-trigger or edge-trigger. In edge-trigger, the data transfer starts from an active edge and ends on an inactive edge of the slave signal. The unit-transfer interrupt flag IF (SPI_CNTRL[16]) will be set to 1 as an inactive edge is detected 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. In level-trigger, the unit-transfer interrupt flag IF of slave will be set when one of the following two conditions occurs. The first condition is that if the number of transferred bits matches the settings of TX_BIT_LEN(SPI_CNTRL[7:3]), the unit transfer interrupt flag IF of slave will be set. As to the second condition, if the master set the slave select pin to inactive level during the transfer is in progress, it will force slave device to terminate the current transfer no matter how many bits have been transferred and the unit transfer interrupt flag will be set. User can read the status of LTRIG_FLAG bit to check if the data has been completely transferred

Timing Condition

The CLKP (SPI_CNTRL[11]) defines the SPI clock idle state. If CLKP = 1, the output of SPIbus clock is high at idle state; if CLKP = 0, it is lowat idle state.

TX_NEG (SPI_CNTRL[2]) defines the data transmitted out either on negative edge or on positive

edge of SPIbus clock.

RX_NEG (SPI_CNTRL[1]) defines the data received either on negative edge or on positive edge of SPI bus clock.

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.

Transmit/Receive Bit Length

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

When SPI controller finishes a transaction, i.e. receives or transmits a special count of bits defined in TX_BIT_LEN (SPI_CNTRL[7:3]), the unit transfer interrupt flag will be set to 1.

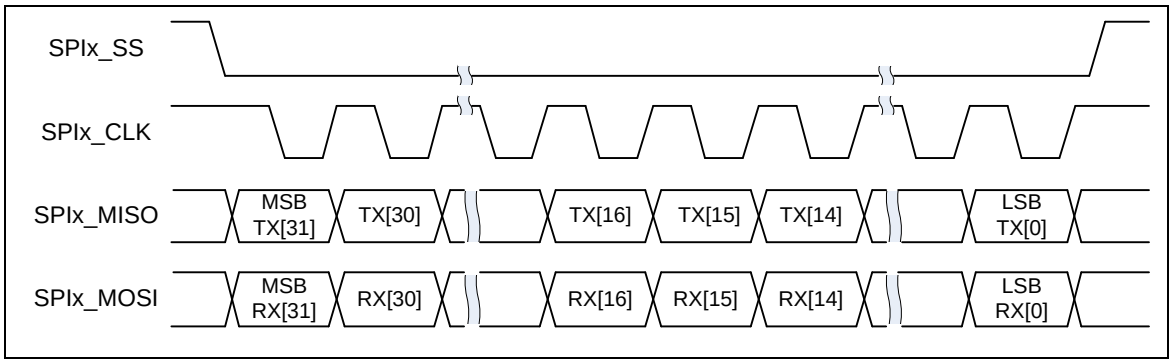


Figure 6-13132-Bit in One Transaction

LSB/MSB First

The LSB (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.

6.15.5.2 Automatic Slave Selection

In Master mode, if AUTOSS (SPI_SSR[3]) is set, the slave select signal will be generated automatically and output to the SPIx_SS pin according to whether SSR (SPI_SSR[0]) is enabled or not. The slave selection signal will be set to active state automatically whenthe SPI data transfer is started by writingto TX FIFOin FIFO mode or bysetting the GO_BUSY (SPI_CNTRL[0]) when the FIFO mode is disabled. It will be set to inactive state when SPI bus is idle. If SPI bus is not idle, i.e. TX FIFO, TX shift register or TX skew buffer is not empty, the slave selection signal will be set to inactive state between transactions if the value of SP_CYCLE(SPI_CNTRL[15:12]) is greater than or equal to 3.

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.

If the AUTOSS bit is cleared, the slave select output signal will be asserted/de-asserted by manual setting/clearing SSR (SPI_SSR[0]). The active state of the slave select output signal is specified in SS_LVL bit (SPI_SSR[2]).

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 peripheral clock periods between two successive transactions.

The duration between the slave selection signal active edgeand the first SPI bus clock edge is 1

SPI bus clock cycle and the duration between the last SPI bus clock and the slave selection signal inactive edge is 1.5 SPI bus clock cycle.

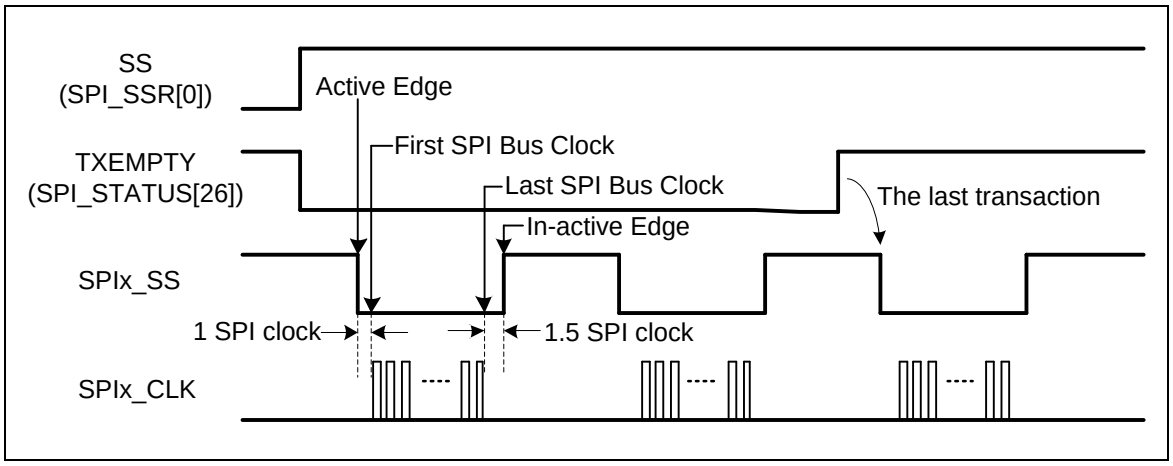


Figure 6-132AutomaticSlave Selection (SS_LVL = 0, SP_CYCLE > 0x2)

6.15.5.1 Word Suspend

These four bits field 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 the software disables the FIFO mode.

6.15.5.2 Byte Reorder

When the transfer is set as MSB first (LSB (SPI_CNTRL[10])= 0) and the REORDER (SPI_CNTRL[19]) is set to 1, the data stored in the TX buffer and RX buffer will be rearranged in the order as [Byte0, Byte1, Byte2, Byte3] in 32-bit Transfer 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 (SPI_CNTRL[7:3]) is configured as 16, 24, or 32 bits.

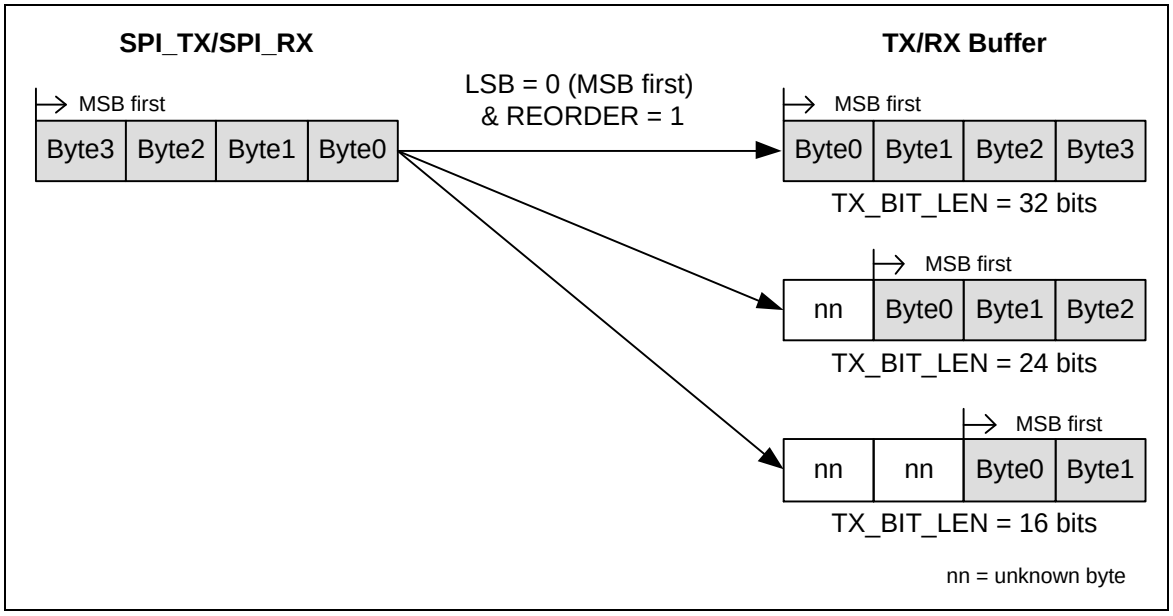


Figure 6-133 Byte Reorder Function

6.15.5.3 Byte Suspend

In Master mode, if REORDER (SPI_CNTRL[19]) is set to 1, a suspend interval of 0.5 ~ 15.5 SPI clock periods will be inserted by hardware between two successive bytes in a transaction word. Both settings of byte suspend interval and word suspend interval are configured in SP_CYCLE (SPI_CNTRL[15:12]).

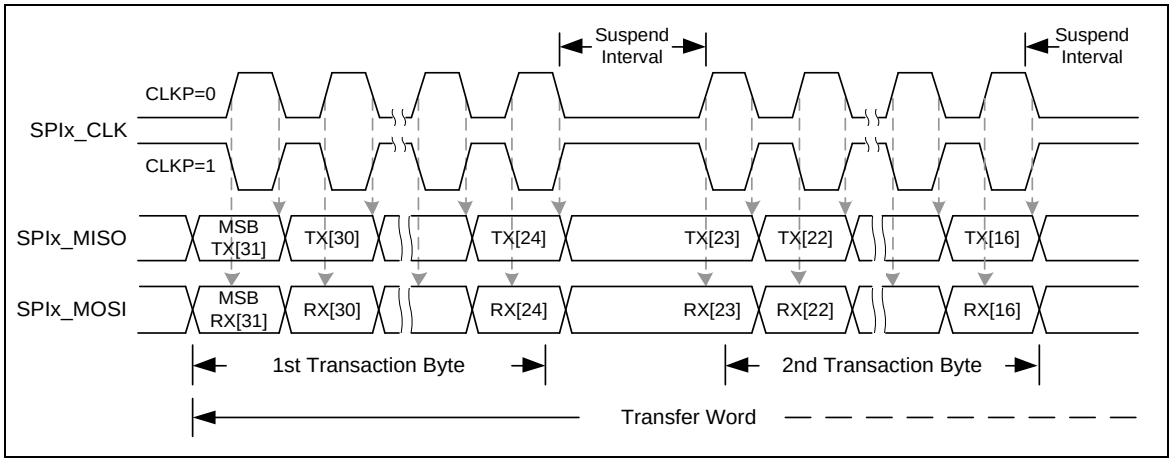


Figure 6-134 Timing Waveform for Byte Suspend

6.15.5.4 Slave 3-Wire Mode

When the NOSLVSEL (SPI_CNTRL2[8]) 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,SPIx_CLK, SPIx_MISO, and SPIx_MOSI, are required to communicate with a SPI master. The SPIx_SS pin can be configured as a GPIO. When the NOSLVSEL bit is set to 1, the SPI slavewill be ready to transmit/receive data after the GO_BUSY (SPI_CNTRL[0])is set to 1.As the number of received bits meets the requirement which defined in TX_BIT_LEN (SPI_CNTRL[7:3]), the unit-transfer interrupt flag IF (SPI_CNTRL[16]) will be set to 1.

Note: In Slave 3-wire mode, the SS_LTRIG, SPI_SSR[4], should be set as 1.

6.15.5.5 FIFO Mode

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

The transmit FIFO buffer is an 8-layer depth, 32-bit wide, first-in and first-out register buffer. Data can be written to the transmit FIFO buffer through software by writing the SPI_TX 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 (SPI_STATUS[27]) 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 (SPI_STATUS[26]) will be set to 1. Notice that the TX_EMPTY flag (SPI_STATUS[26]) is set to 1 while the last transaction is still in progress. In Master mode, both the GO_BUSY (SPI_CNTRL[0]) and TX_EMPTY (SPI_STATUS[26]) 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 and first-out register buffer. The receive control logic will store the received data to this buffer. The software can read the FIFO buffer data from SPI_RX register. There are FIFO related status bits, like RX_EMPTY (SPI_STATUS[24]) and RX_FULL (SPI_STATUS[25]), to indicate the current status of FIFO buffer.

The transmitting and receiving threshold can be configured by setting the TX_THRESHOLD (SPI_FIFO_CTL[30:28]) and RX_THRESHOLD (SPI_FIFO_CTL[26:24]). When the count of valid data stored in transmit FIFO buffer is less than or equal to TX_THRESHOLD setting, the TX_INTSTS (SPI_STATUS[4]) 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 (SPI_STATUS[0]) will be set to 1.

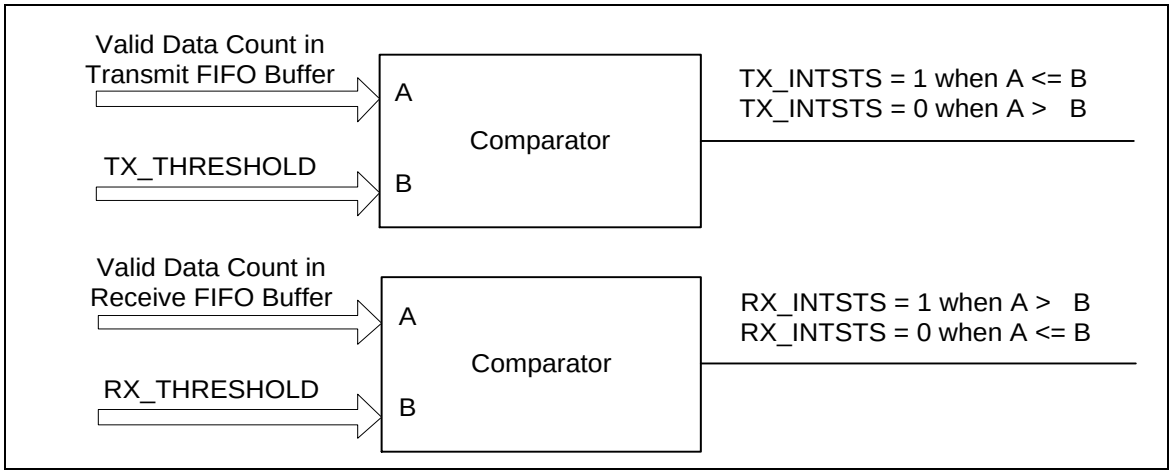


Figure 6-135FIFO Threshold Comparator

In FIFO mode, the software can write 8 data to the SPI transmit FIFO buffer in advance. When the SPI controller operates with FIFO mode, GO_BUSY (SPI_CNTRL[0]) will be controlled by hardware, software should not modify the content of SPI_CNTRL register unless clearing the FIFO bit (SPI_CNTRL[21]) to disable the FIFO mode.

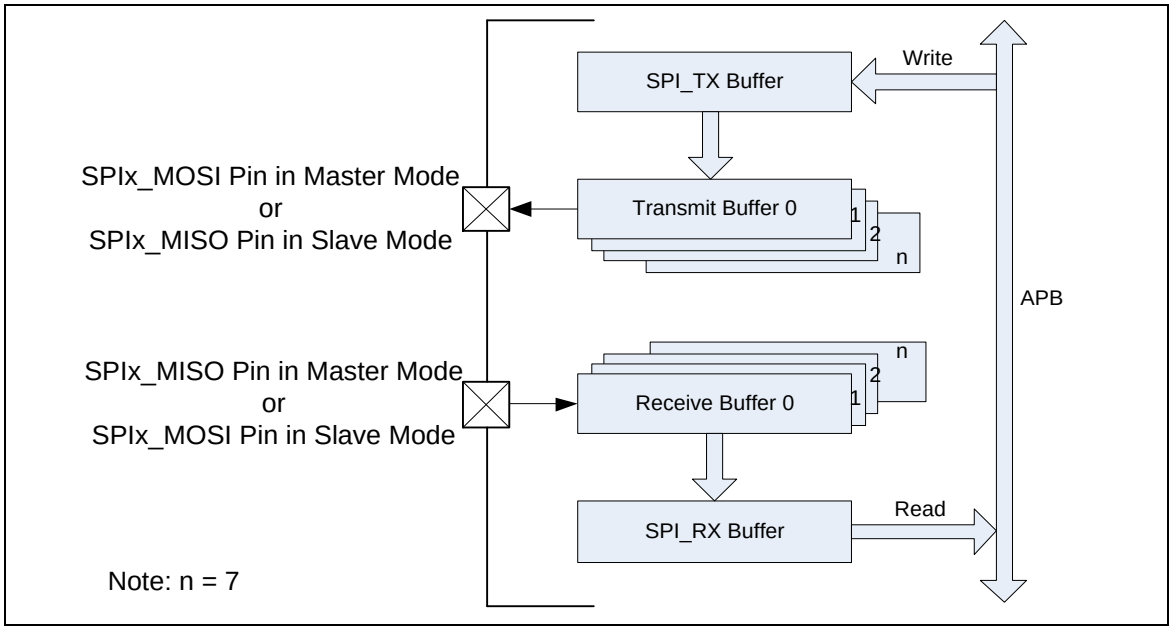


Figure 6-136FIFO Mode Block Diagram

In Master mode, the first datum is written to the SPI_Txregister,the TX_EMPTY flag (SPI_STATUS[16]) will be cleared to 0.The transmission will start after 1 APB clock cycle and 6 peripheral clock cycles. User can write the next data into SPI_Txregister immediately. The SPI controller will insert a suspend interval between two successive transactions.The period of suspend interval is decided by the setting of SP_CYCLE (SPI_CNRTL[15:12]). If the SP_CYCLE (SPI_CNTRL[15:12]) equals 0, SPI controller can perform continuous transfer. User can write data into SPI_Txregisteras long as the TX_FULL(SPI_STATUS[27]) is 0.

In the Example 1 of theFigure6-137, it indicates the updated condition of TX_EMPTY (SPI_STATUS[26]) and the relationship among the FIFO buffer, shift register and the skew buffer. The TX_EMPTY (SPI_STATUS[16]) is set to 0 when the Data0 is written into the FIFO buffer. The Data0 will be loaded into the shift register by core logical and the TX_EMPTY (SPI_STATUS[26]) will be to 1. The Data0 in shift register will be shifted into skew buffer by bit for transmission until the transfer is done.

In the Example 2, it indicates the updated condition of TX_FULL (SPI_STATUS[27]) when there are 8 data in the FIFO buffer and the next data of Data9 is not written into the FIFO buffer when the TX_FULL = 1.

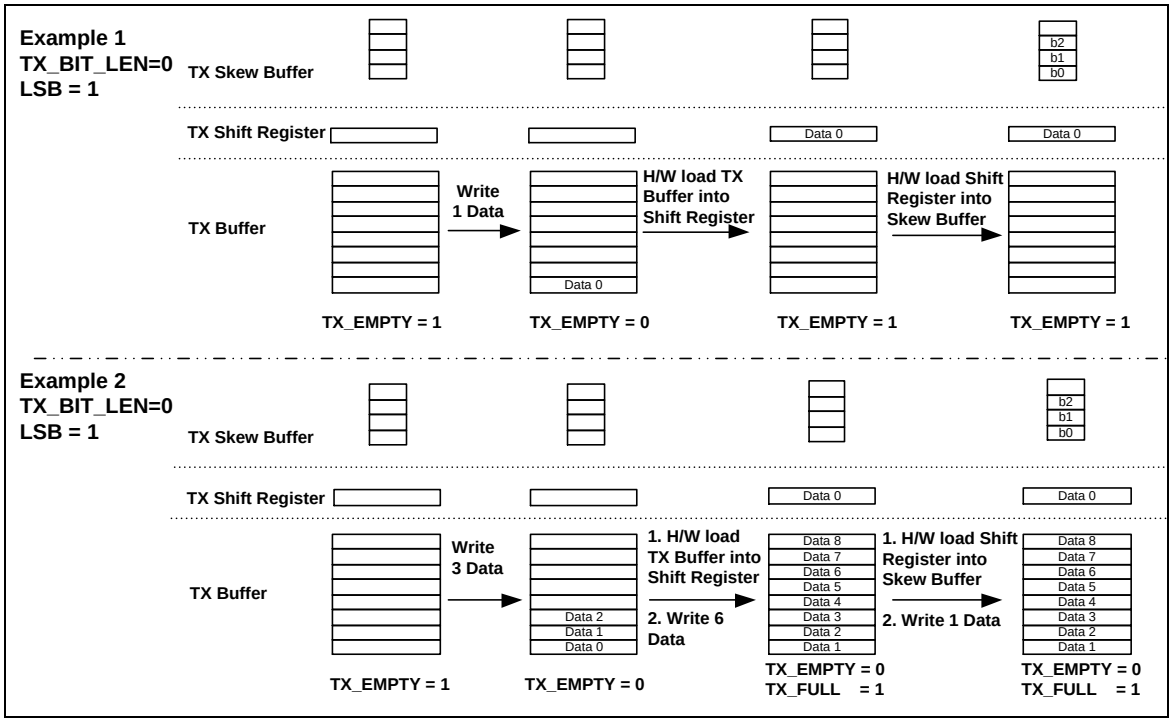


Figure6-137Transmit FIFO Buffer Example

The subsequent transactionswill betriggered automatically if the transmitted data are updated in time. If the SPI_TX registerdoes not be updated afterall data transferare done, the transfer will stop.

In Master mode, during reception operation, the serial data are received from SPIx_MISO pin and stored to receive FIFO buffer.

The receive data (Data0's b0, b1, …, b31) is stored into skew buffer first according the serial clock (SPIx_CLK) and then is shifted into the shift register bit by bit. The core logic will load the data in shift register into FIFO buffer when the receive data bit reach the value of TX_BIT_LEN (SPI_CNTRL[7:3]). The RX_EMPTY(SPI_STATUS[24]) will be cleared to 0 while the receive FIFO buffer contains unread data (see the Example 1 of Receive FIFO Buffer Example). The received data can be read by software from SPI_Rxregisteras long as the RX_EMPTY(SPI_STATUS[24]) is 0.If the receive FIFO buffer contains 8 unread data, the RX_FULL(SPI_STATUS[25]) will be set to 1 (see the Example 2 of Receive FIFO Buffer Example).In Slave mode, when the FIFO bit is set as 1, the GO_BUSY bit will be set as 1 by hardware automatically. If user wants to stop the Slave mode SPI data transfer, both the FIFO bit and GO_BUSY bit must be cleared to 0 by software.

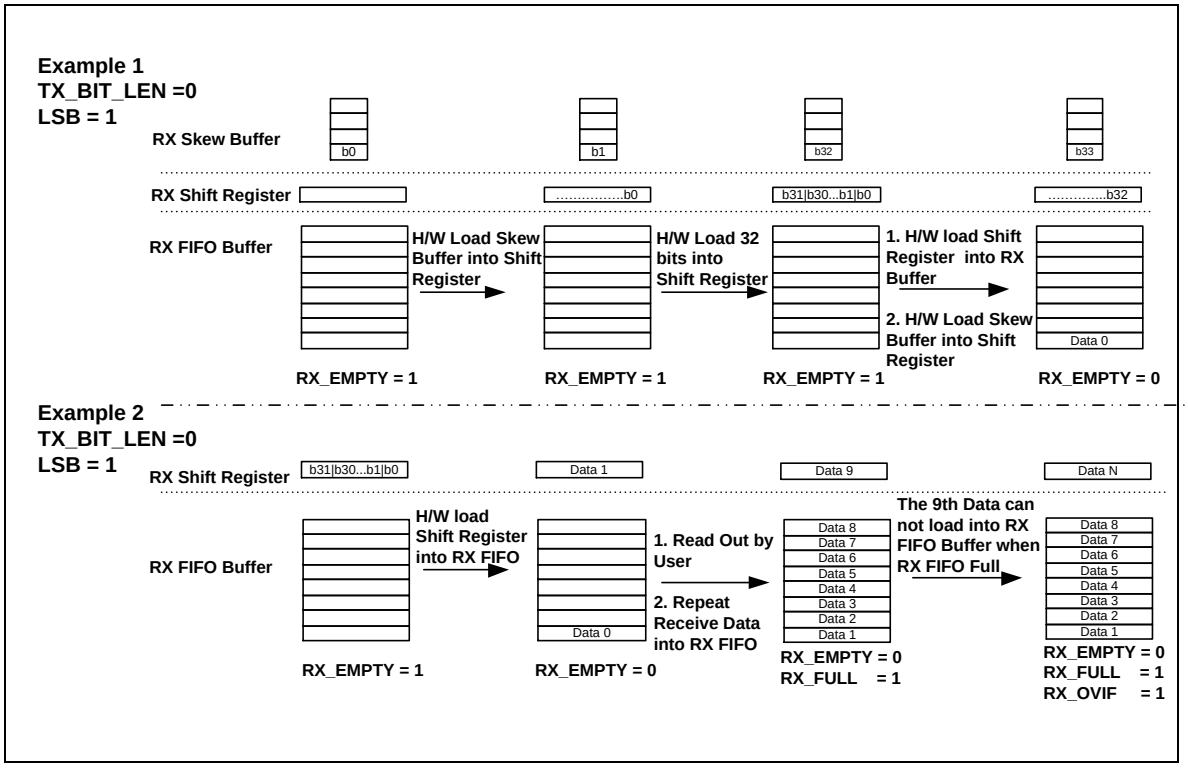


Figure6-138Receive FIFO Buffer Example

In Slave mode, during transmission operation, when the software writes data to SPI_TX register, 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. The software can write data to SPI_TX register as long as TX_FULL flag is 0. After all data have been drawn out by the SPI transmission logic unit and the software does not update the SPI_TX register, the TX_EMPTY flag will be set to 1.

In Slave mode reception operation, the serial data is received from SPIx_MOSI pin and stored to SPI_RX register. The reception mechanism is similar to Master mode reception operation.

6.15.5.6 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(SPI_STATUS[11]), will be set to 1 when the slave senses the SPI clock signal. The SPI controller will issue an interrupt if the SSTA_INTEN (SPI_CNTRL2[10]) is set to 1. If the count of the received bits is less than the setting of TX_BIT_LEN(SPI_CNTRL[7:3]) and there is no more SPI clock input over the expected time period which is defined by the user, the user can set the SLV_ABORT (SPI_CNTRL2[9]) to abort the current transfer.The unit transfer interrupt flag, IF, (SPI_STATUS[16]) will be set to 1 if the software set the SLV_ABORT bit(SPI_CNTRL2[9]).

- 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 peripheral clock periods in Master mode or over 576 SPI peripheral clock periods in Slave mode, it will send a time out interrupt to the system if the time out interrupt enable bit TIMEOUT_INTEN (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 TX_INTEN (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 RX_INTEN (SPI_FIFO_CTL[2]) is set to 1.

6.15.6 Timing Diagram

The active state of slave select signal can be defined by setting theSS_LVL bit (SPI_SSR[2]) and SS_LTRIG bit (SPI_SSR[4]). The SPI clock 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 SPI 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.

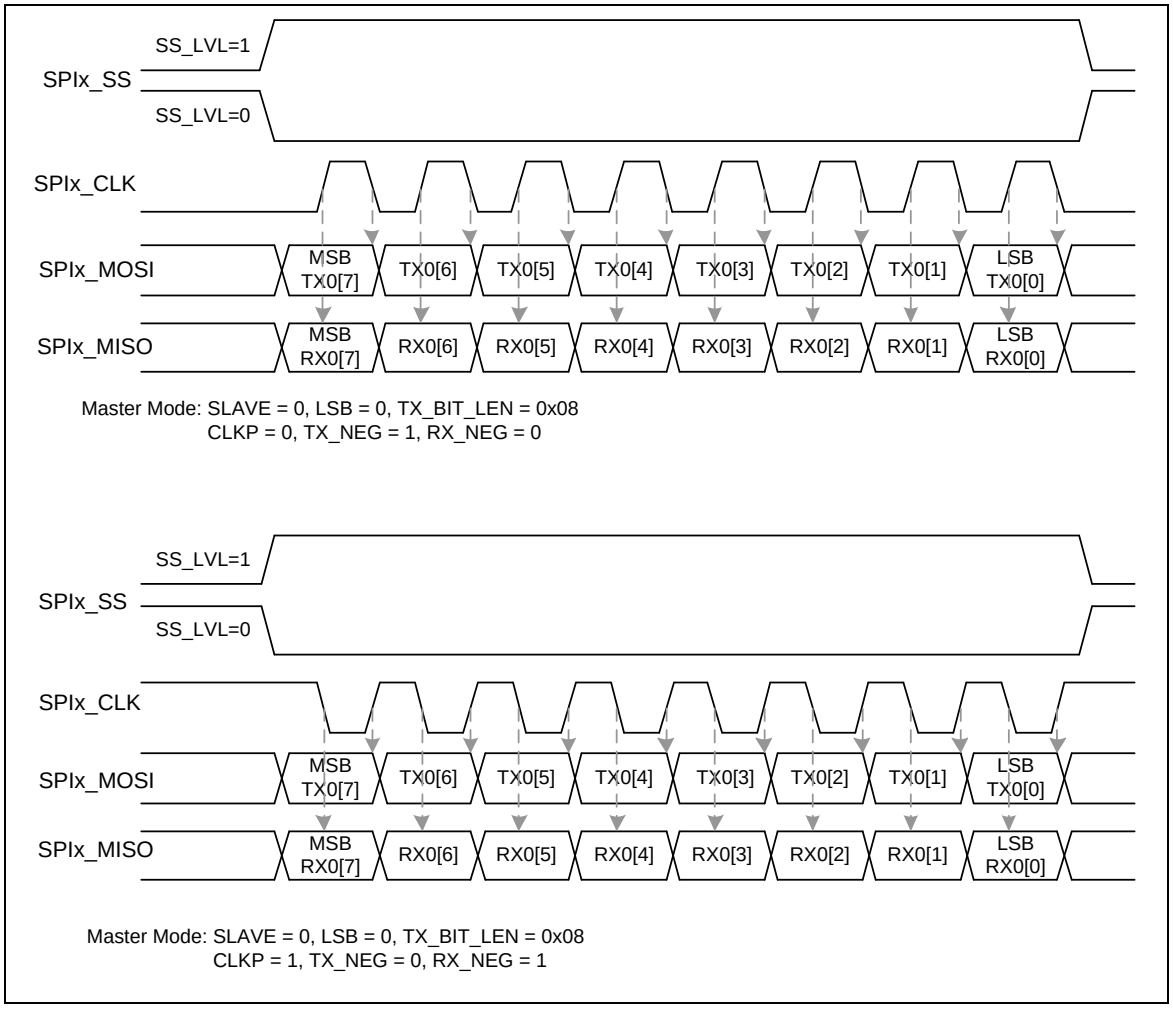


Figure 6-139 SPI Timing in Master Mode

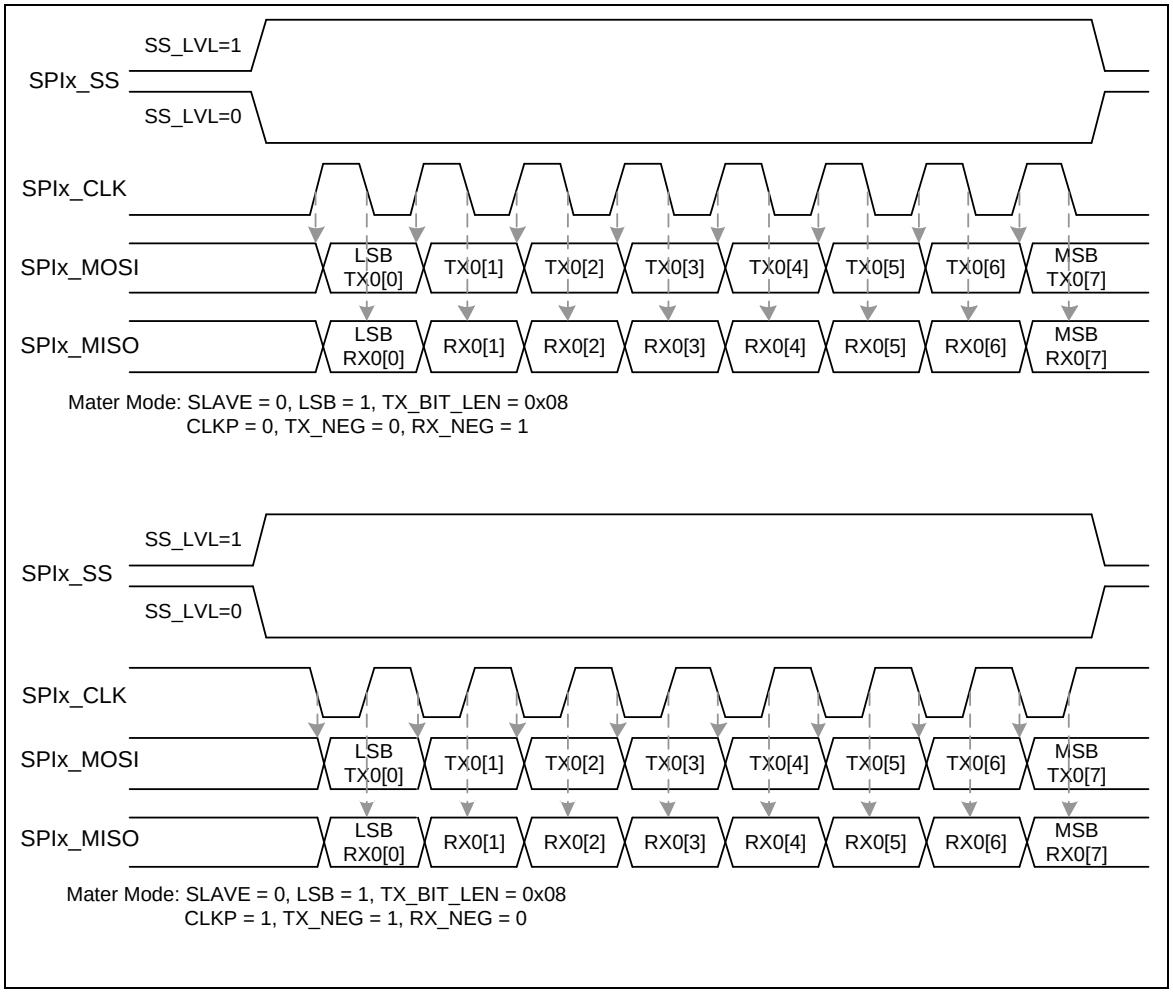


Figure 6-140 SPI Timing in Master Mode (Alternate Phase of SPI_CLK)

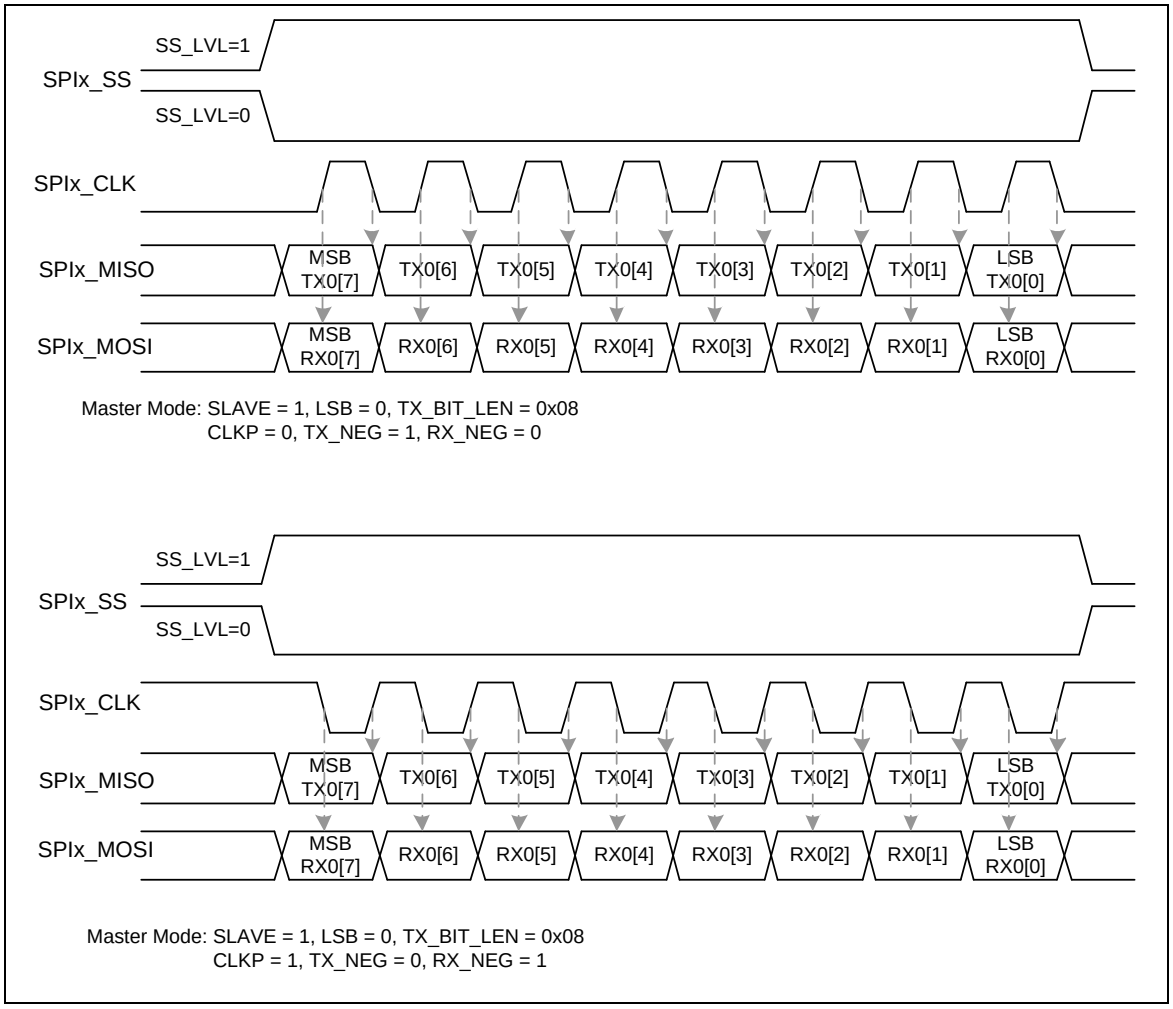


Figure 6-141 SPI Timing in Slave Mode

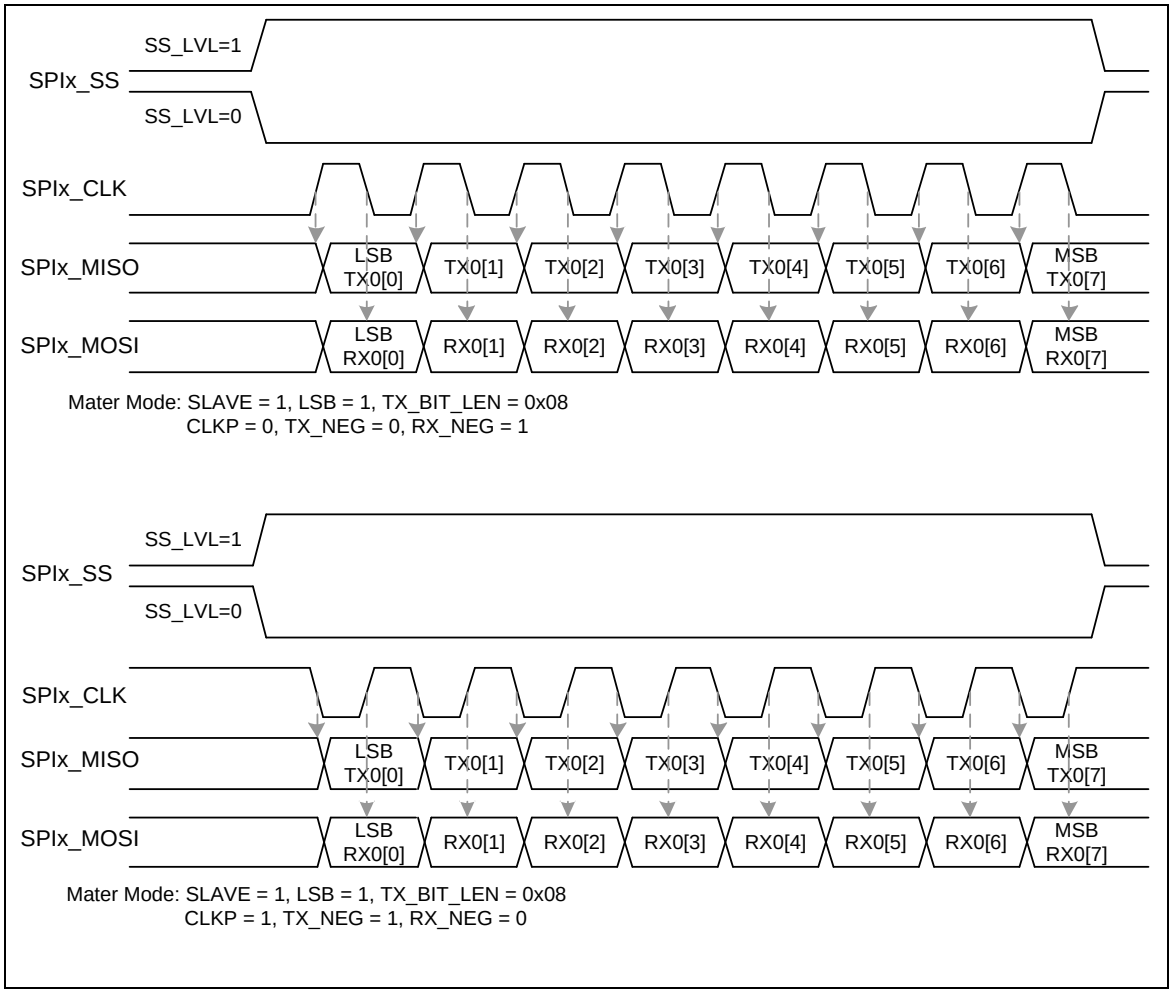


Figure 6-142 SPI Timing in Slave Mode (Alternate Phase of SPI bus clock)

6.15.7 Programming Examples

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

- Data bit is latched on positive-edge of SPIbus clock.
- Data bit is driven onnegative-edge of SPIbus clock.
- Data is transferred from MSB first.
- SPI bus clock is low at idle state.
- Only one byte of data will be transmitted/received in a transaction.
- 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 SPI bus clock.

2) Write the related settings into the SPI_CNTRL register to control this SPI Master actions
   1. Set this SPI controller as Master device. SLAVE (SPI_CNTRL[18]) = 0.
   2. Set the SPI bus clock idle state as low-level. CLKP (SPI_CNTRL[11]) = 0.
   3. Transmit dataonfalling edge of SPI bus clock. TX_NEG (SPI_CNTRL[2]) = 1.
   4. Capture dataonrising edge of SPI bus clock. RX_NEG (SPI_CNTRL[1]) = 0.
   5. Set the bit length of a transaction as 8bits. TX_BIT_LEN (SPI_CNTRL[7:3]) = 0x08.
   6. Set transfer sequence as MSB first.LSB (SPI_CNTRL[10]) = 0.
3) Write the SPI_SSR register a proper value for the related settings of Master mode:
   1. Disable the automatic slave selection function. AUTOSS(SPI_SSR[3]) to 0.
   2. Select low level trigger output of slave select signal.Set SS_LVL (SPI_SSR[2]) = 0 and SS_LTRIG(SPI_SSR[4]) = 1.
   3. SetSSR (SPI_SSR[0]) to 1 to active the off-chip Slave device.
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_TX 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, software does not need to update the SPI_TX register.
6) Set the GO_BUSY (SPI_CNTRL[0]) to 1 to start the data transfer on the SPI interface.
7) Waiting for SPI interrupt if the Unit Transfer Interrupt function is enabled or just polling the GO_BUSY bit (SPI_CNTRL[0]) till it is cleared to 0 by hardware automatically.
8) Read out the received one byte data from SPI_RX[7:0].
9) Go to 4) to continue another data transfer or set SSR 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 SPIbus clock.
- Data bit is driven onnegative-edge of SPIbus clock.
- Data is transferred from LSB first.
- SPI bus clock is high at idle state.
- Only one byte of data will 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 ofSlave mode.

   Select high level and level trigger for the input of slave select signal by setting SS_LVL (SPI_SSR[2])to 1 and setting SS_LTRIG (SPI_SSR[4])to 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. SLAVE (SPI_CNTRL[18]) = 1.

2. Select the SPI clock high at idle state. CLKP (SPI_CNTRL[11]) = 1.
3. Transmit data at negative-edge of SPI bus clock. TX_NEG (SPI_CNTRL[2])= 1.
4. Capture data at positive-edge of SPI bus clock. RX_NEG (SPI_CNTRL[1]) = 0.
5. Set the bit length of a transaction as 8-bit. TX_BIT_LEN (SPI_CNTRL[7:3])= 0x08.
6. Set transfer sequence as LSB first.LSB (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_TX 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, software does not need to updatethe SPI_TX register.
5) Set GO_BUSY (SPI_CNTRL[0])to 1 to wait for the slave select trigger input and SPI bus clock input from the off-chip Master device to start the data transfer with the SPI interface.
6) Waiting for SPI interrupt if the Unit Transfer Interrupt function is enabled, or just polling the GO_BUSY bit (SPI_CNTRL[0]) till it is cleared to 0 by hardware automatically.
7) Read out the received one byte data from SPI_RX[7:0].
8) Go to 3) to continue another data transfer orclear GO_BUSY bit to stop data transfer.

6.15.8 Register Map

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

Register	Offset	R/W	Description	Reset Value
SPI Base Address: **SPI0_BA = 0x4003_0000** **SPI1_BA = 0x4003_4000** **SPI2_BA = 0x4013_0000**				
SPI_CNTRL **x=0,1,2**	SPIx_BA+0x00	R/W	Control and Status Register	0x0500_3004
SPI_DIVIDER **x=0,1,2**	SPIx_BA+0x04	R/W	Clock Divider Register	0x0000_0000
SPI_SSR **x=0,1,2**	SPIx_BA+0x08	R/W	Slave Select Register	0x0000_0000
SPI_RX **x=0,1,2**	SPIx_BA+0x10	R	Data Receive Register	0x0000_0000
SPI_TX **x=0,1,2**	SPIx_BA+0x20	W	Data Transmit Register	0x0000_0000
SPI_CNTRL2 **x=0,1,2**	SPIx_BA+0x3C	R/W	Control and Status Register 2	0x0000_1000
SPI_FIFO_CTL **x=0,1,2**	SPIx_BA+0x40	R/W	SPI FIFO Control Register	0x4400_0000
SPI_STATUS **x=0,1,2**	SPIx_BA+0x44	R/W	SPI Status Register	0x0500_0000

6.15.9 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
Reserved		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 FullIndicator (Read Only)** A mutual mirror bit of SPI_STATUS[27]. 0 = TransmitFIFO buffer is not full. 1 = TransmitFIFO buffer is full.
[26]	**TX_EMPTY**	**Transmit FIFO Buffer Empty Indicator (Read Only)** A mutual mirror bit of SPI_STAUTS[26]. 0 = TransmitFIFO buffer is not empty. 1 = TransmitFIFO buffer is empty.
[25]	**RX_FULL**	**Receive FIFO Buffer Full Indicator (Read Only)** A mutual mirror bit of SPI_STATUS[25]. 0 = Receive FIOF buffer is not full. 1 = Receive FIFO buffer is full.
[24]	**RX_EMPTY**	**Receive FIFO Buffer Empty Indicator (Read Only)** A mutual mirror bit of SPI_STATUS[24]. 0 = Receive FIFO buffer is not empty. 1 = Receive FIFO buffer is empty.
[23:22]	**Reserved**	Reserved.
[21]	**FIFO**	**FIFO Mode Enable Bit** 0 = FIFO mode Disabled. 1 = FIFO mode Enabled. **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 writing data to the transmit FIFO buffer; 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.
[20]	**Reserved**	Reserved.
[19]	**REORDER**	**Byte Reorder Function Enable Bit** 0 = Byte reorder function Disabled. 1 = Byte reorder function Enabled.A byte suspend interval will be inserted among each byte. The period of the byte suspend interval depends on the setting of SP_CYCLE. **Note:** 2. Byte reorder function is only available if TX_BIT_LEN is defined as 16, 24, and 32 bits. 3. In Slave mode with level-trigger configuration, the slave select pin must be kept at active state during the byte suspend interval.
[18]	**SLAVE**	**Slave Mode Enable Bit** 0 = Master mode. 1 = Slave mode.
[17]	**IE**	**Unit Transfer Interrupt Enable Bit** 0 = SPI unit transferinterrupt Disabled. 1 = SPI unit transferinterrupt Enabled.
[16]	**IF**	**Unit Transfer Interrupt Flag** 0 = No transaction has been finished since this bit was cleared to 0. 1 = SPI controller has finished one unit transfer. **Note:** This bit will be cleared by writing 1 to 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 SPI clock cycle Example: SP_CYCLE = 0x0 … 0.5 SPI clock cycle. SP_CYCLE = 0x1 … 1.5 SPI clock cycle. …… SP_CYCLE = 0xE … 14.5 SPI clock cycle. SP_CYCLE = 0xF … 15.5 SPI clock cycle.
[11]	**CLKP**	**Clock Polarity** 0 = SPIbus clockis idle low. 1 = SPIbus clockis idle high.
[10]	**LSB**	**Send LSB First** 0 = The MSB, which bit of transmit/receive register depends on the setting of TX_BIT_LEN, is transmitted/received first. 1 = The LSB, bit 0 of the SPI TX register, is sent first to the SPI data output pin, and the first bit received from the SPI data input pin will be put in the LSB position of the RX register (bit 0 of SPI_RX).
[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 is8 bits and can up to 32bits.

		TX_BIT_LEN = 0x01~0x07 … can't use. TX_BIT_LEN = 0x08 … 8 bits. TX_BIT_LEN = 0x09 … 9 bits. …… TX_BIT_LEN = 0x1F … 31bits. TX_BIT_LEN = 0x00 … 32bits.
[2]	**TX_NEG**	**Transmit onNegative Edge** 0 = Transmitted data output signal is changed on the rising-edge of SPIbus clock. 1 = Transmitted data output signal is changed on the falling-edge of SPIbus clock.
[1]	**RX_NEG**	**Receive onNegative Edge** 0 = Received data input signal is latched on the rising-edge of SPIbus clock. 1 = Received data input signal is latched on the falling-edge of SPIbus clock.
[0]	**GO_BUSY**	**SPI Transfer Control Bit and Busy Status** 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. 0 = Data transfer stopped. 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. **Note:** When FIFO mode is disabled, all configurations should be set before writing 1 to this GO_BUSY bit.

SPI Clock 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
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
DIVIDER							

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7:0]	**DIVIDER**	**Clock Divider 1 Register** The value in this field is the frequency divider for generating the SPI peripheral clock, f_{spi_eclk}, and the SPI bus clock of SPI master. The frequency is obtained according to the following equation. $f_{spi\_eclk} = \frac{f_{spi\_clock\_src}}{(DIVIDER+1)}$ where $f_{spi\_clock\_src}$ is the SPI peripheral 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	Reserved	SSR

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**LTRIG_FLAG**	**Level Trigger Accomplish Flag** In Slave mode, this bit indicateswhether the received bit number meets the requirement or not after the current transaction done. 0 = Transferred bit length of one transaction doesnot meet the specified requirement. 1 = Transferred bit length meets the specified requirement which defined in TX_BIT_LEN.**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 peripheral 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)** 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. 1 = Slave select signal is level-trigger.The SS_LVL bit decides the signal is active low or active high.
[3]	**AUTOSS**	**Automatic Slave Select Function Enable (Master Only)** 0 = If this bit is cleared, slave select signals will be asserted/de-asserted by setting /clearing the corresponding bits ofSPI_SSR[0]. 1 = If this bit is set, SPI_SS signals will be generated automatically by hardware. It means that device/slave select signal, which is set in SPI_SSR[0], will be asserted by the SPI controller when transmit/receive is started, and will be de-asserted after each transmit/receive is finished.
[2]	**SS_LVL**	**Slave Select Active Level** This bit defines the active status of slave select signal (SPI_SS). 0 = The slave select signal SPI_SS is active on low-level/falling-edge. 1 = The slave select signal SPI_SS is active on high-level/rising-edge.
[1]	**Reserved**	Reserved.
[0]	**SSR**	**Slave Select Control Bits (Master Only)** If AUTOSS bit is cleared 0 = Set the SPIx_SS pin to inactive state.

		1 = Set the SPIx_SS pin to active state. If the AUTOSS bit is set 0 = Keep the SPIx_SS pin to inactive state. 1 = Select the SPIx_SS pin to be automatically driven to active state for the duration of data transfer, and will be driven to inactive state for the rest of the time. The active state of SPIx_SS is specified in SS_LVL.

SPI Data Receive Register (SPI_RX)

Register	Offset	R/W	Description	Reset Value
SPI_RX	SPIx_BA+0x10	R	Data Receive Register	0x0000_0000

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

Bits	Description	
[31:0]	RX	**Data Receive Register** The data receive register holds the datum received from SPI data input pin. If the FIFO mode is disabled, the software can access the last received data 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, thenthe receive FIFO buffers 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_TX	SPIx_BA+0x20	W	Data Transmit Register	0x0000_0000

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

Bits	Description	
[31:0]	TX	**Data Transmit Register** The Data Transmit Register holds the data to be transmitted in the next transfer. The number of valid bits depends on the setting of transmit bit length field TX_BIT_LEN in the SPI_CNTRL register. For example, if TX_BIT_LEN is set to 0x08, the bit field 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 Control and Status Register 2 (SPI_CNTRL2)

Register	Offset	R/W	Description	Reset Value
SPI_CNTRL2	SPIx_BA+0x3C	R/W	Control and Status Register 2	0x0000_1000

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

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**SS_INT_OPT**	**Slave Select Inactive Interrupt Option** This setting is only available if the SPI controller isconfigured as level trigger slave device. 0 = As the slave select signal goes to inactive level, the IF bit will NOT be set to 1. 1 =As the slave select signal goes to inactive level, the IF bit will be set to 1.
[15:12]	**Reserved**	Reserved.
[11]	**SLV_START_INTSTS**	**Slave 3-wire Mode Start Interrupt Status** This bit indicatesifa transaction has started in Slave 3-wire mode. It is a mutual mirror bit of SPI_STATUS[11]. 0 = Slave has not detected any SPI clock transition since the SSTA_INTEN bit was set to 1. 1 =A transaction has started in Slave 3-wire mode. It will be cleared automaticallywhena transactionis done or by writing 1to this bit.
[10]	**SSTA_INTEN**	**Slave 3-wire Mode Start Interrupt Enable Bit** 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. 0 =Transaction start interrupt Disabled. 1 =Transaction start interrupt Enabled. It will be cleared to 0as the current transfer is done or the SLV_START_INTSTS bit is cleared.
[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 SPI clock input over one transaction time in Slave 3-wire mode, user can set this bit to force the current transfer done and then user can get a transfer done interrupt event. 0 = No effect.

		1 = Force the current transaction done. **Note:**This bitwill be cleared to 0 automatically by hardware afterit is set to 1 by software..
[8]	**NOSLVSEL**	**Slave 3-wire Mode Enable Bit** In Slave 3-wire mode, the SPI controller can work with 3-wire interface including SPI_CLK, SPI_MISO, and SPI_MOSI. 0 = 4-wire bi-direction interface in Slave mode. 1 = 3-wire bi-direction interface in Slave mode. The controller will be ready to transmit/receive data after the GO_BUSY bit is set to 1. **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 Bit** 0 = Time-out interrupt Disabled. 1 = Time-out interrupt Enabled.
[20:7]	**Reserved**	Reserved.
[6]	**RXOV_INTEN**	**Receive FIFO Overrun Interrupt Enable Bit** 0 = Receive FIFO overrun interrupt Disabled. 1 = Receive FIFO overrun interrupt Enabled.
[5:4]	**Reserved**	Reserved.
[3]	**TX_INTEN**	**Transmit Threshold Interrupt Enable Bit** 0 = TX threshold interrupt Disabled. 1 = TX threshold interrupt Enabled.
[2]	**RX_INTEN**	**Receive Threshold Interrupt Enable Bit** 0 = RX threshold interrupt Disabled.

		1 = RX threshold interrupt Enabled.
[1]	**TX_CLR**	**Clear Transmit FIFO Buffer** 0 = No effect. 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]	**RX_CLR**	**Clear Receive FIFO Buffer** 0 = No effect. 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.

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_OVERRUN	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)** A mutual mirror bit of SPI_CNTRL[27]. 0 = Transmit FIFO buffer is not full. 1 = Transmit FIFO buffer is full.
[26]	**TX_EMPTY**	**Transmit FIFO Buffer Empty Indicator (Read Only)** A mutual mirror bit of SPI_CNTRL[26]. 0 = Transmit FIFO buffer is not empty. 1 = Transmit FIFO buffer is empty.
[25]	**RX_FULL**	**Receive FIFO Buffer Empty Indicator (Read Only)** A mutual mirror bit of SPI_CNTRL[25]. 0 = Receive FIFO buffer is not empty. 1 = Receive FIFO buffer is empty.
[24]	**RX_EMPTY**	**Receive FIFO Buffer Empty Indicator (Read Only)** A mutual mirror bit of SPI_CNTRL[24]. 0 = Receive FIFO buffer is not empty. 1 = Receive FIFO buffer is empty.
[23:21]	**Reserved**	Reserved.
[20]	**TIMEOUT**	**Time-out Interrupt Flag** 0 = No receive FIFO time-out event. 1 = Receive FIFO buffer is not empty and no read operation on receive FIFO buffer over 64 SPI clock period in Master mode or over 576 SPI peripheral clock period in Slave mode. When the received FIFO buffer is read by software, the time-out status will be cleared automatically. **Note:** This bit will be cleared by writing 1 to itself.

[19:17]	**Reserved**	Reserved.
[16]	**IF**	**SPI Unit Transfer Interrupt Flag** A mutual mirror bit of SPI_CNTRL[16]. 0 = No transaction has been finishedsince this bit was cleared to 0. 1 = SPI controller has finished one unit transfer. **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 ifa transaction has started in Slave 3-wire mode. It is a mutual mirror bit of SPI_CNTRL2[11]. 0 = Slave has not detected any SPI clock transition since the SSTA_INTEN bit was set to 1. The transfer is not started. 1 =A transactionhas started in Slave 3-wire mode. It will be clearedasa transaction is done or by writing 1to this bit.
[10:5]	**Reserved**	Reserved.
[4]	**TX_INTSTS**	**Transmit FIFO Threshold Interrupt Status (Read Only)** 0 = The valid data count within the transmit FIFO buffer is larger than the setting value of TX_THRESHOLD. 1 = The valid data count within the transmit FIFO buffer is less than or equal to the setting value of TX_THRESHOLD.
[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. 0 = Receive FIFO does not overrun. 1 = Receive FIFO overruns. **Note:** This bit will be cleared by writing 1 to itself.
[1]	**Reserved**	Reserved.
[0]	**RX_INTSTS**	**Receive FIFO Threshold Interrupt Status (Read Only)** 0 =The valid data count within the receive FIFO buffer is less than or equal to the setting value of RX_THRESHOLD. 1 = The valid data count within the receive FIFO buffer is larger than the setting value of RX_THRESHOLD.

6.16 Controller Area Network (CAN)

6.16.1 Overview

The C_CAN consists of the CAN Core, Message RAM, Message Handler, Control Registers and Module Interface (ReferFigure 6-143). The CAN Core performs communication according to the CAN protocol version 2.0 part A and B. The bit rate can be programmed to values up to 1Mbit/s. For the connection to the physical layer, additional transceiver hardware is required.

For communication on a CAN network, individual Message Objects are configured. The Message Objects and Identifier Masks for acceptance filtering of received messages are stored in the Message RAM. All functions concerning the handling of messages are implemented in the Message Handler. These functions include acceptance filtering, the transfer of messages between the CAN Core and the Message RAM, and the handling of transmission requests as well as the generation of the module interrupt.

The register set of the C_CAN can be accessed directly by software through the module interface. These registers are used to control/configure the CAN Core and the Message Handler and to access the Message RAM.

6.16.2 Features

- Supports CAN protocol version 2.0 part A and B.
- Bit rates up to 1 Mbit/s.
- 32 Message Objects.
- Each Message Object has its own identifier mask.
- Programmable FIFO mode (concatenation of Message Objects).
- Maskable interrupt.
- Disabled Automatic Re-transmission mode for Time Triggered CAN applications.
- Programmable loop-back mode for self-test operation.
- 16-bit module interfaces to the AMBA APB bus.
- Supports wake-up function.

6.16.3 Basic Configuration

The basic configurations of CAN are as follows.

- CAN pins are configured on P2_MFP[5:4] or P3_MFP[7:6]registers.
- Enable CAN clock (CAN_EN (APBCLK[24])).
- Reset CAN controller (CAN_RST (IPRSTC2[24])).

6.16.4 Block Diagram

The C_CAN interfaces with the AMBA APB bus. The Figure 6-143 shows the block diagram of the C_CAN.

- **CAN Core**

CAN Protocol Controller and Rx/Tx Shift Register for serial/parallel conversion of messages.

- **Message RAM**

Stores Message Objects and Identifier Masks

- **Registers**

All registers used to control and to configure the C_CAN.

- **Message Handler**

State Machine that controls the data transfer between the Rx/Tx Shift Register of the CAN Core and the Message RAM as well as the generation of interrupts as programmed in the Control and Configuration Registers.

- **Module Interface**

C_CAN interfaces to the AMBA APB 16-bit bus from ARM.

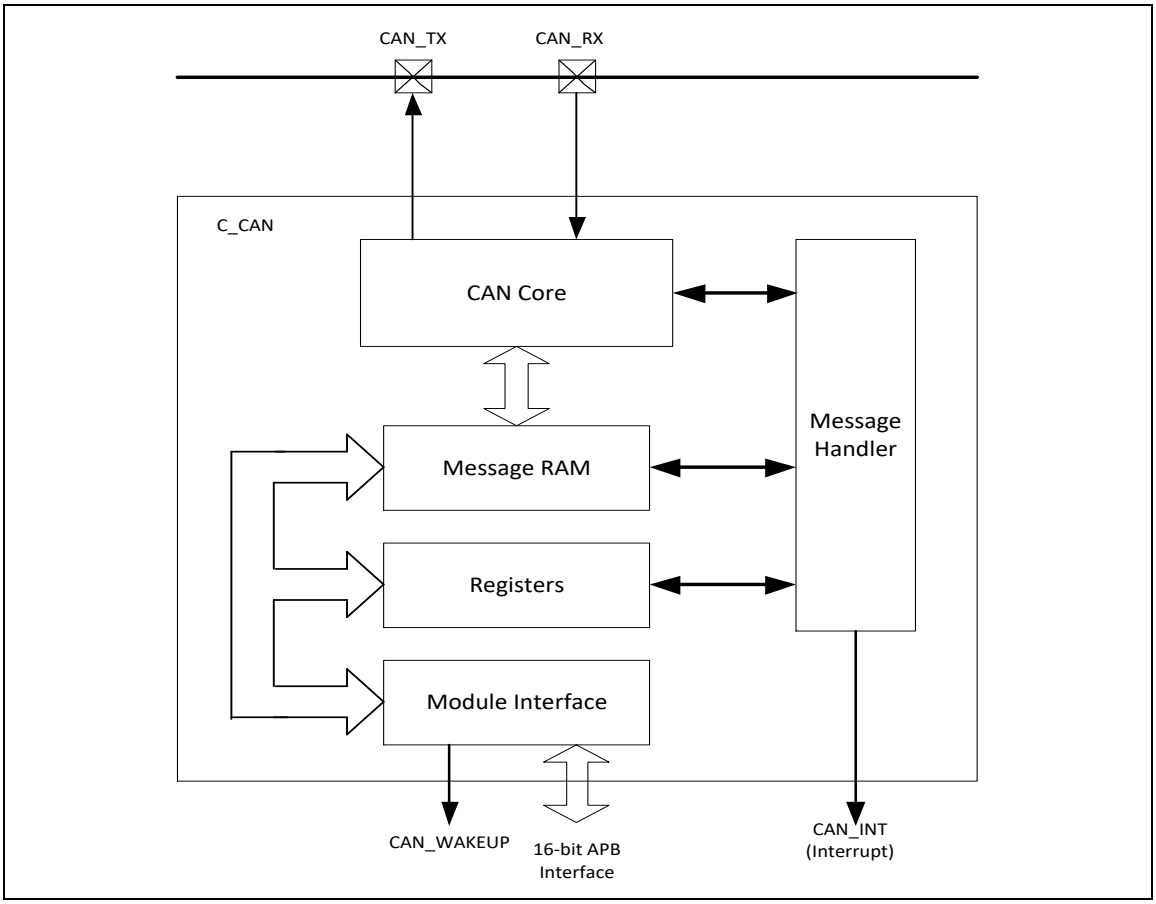


Figure 6-143 CAN Peripheral Block Diagram

6.16.5 Functional Description

6.16.5.1 Software Initialization

The software initialization is started by setting the Init bit (CAN_CON[0]), either by a software or a hardware reset, or by going to bus-off state.

While the Init bit is set, all messages transfer to and from the CAN bus are stopped, and the status of the CAN_TX output pin is recessive (HIGH). The Error Management Logic (EML) counters are unchanged. Setting the Init bit does not change any configuration register.

To initialize the CAN Controller, software has to set up the Bit Timing Register and each Message Object. If a Message Object is not required, the corresponding MsgVal bit (CAN_Ifn_ARB2[15]) should be cleared. Otherwise, the entire Message Object has to be initialized.

Access to the Bit Timing Register and to the Baud Rate Prescaler Extension Register for configuring bit timing is enabled when both the Init and CCE (CAN_CON[6]) bits are set.

Resetting the Init bit (by software only) finishes the software initialization. Later, the Bit Stream Processor (BSP) (see Section 6.5.7.15: Configuring the Bit Timing) synchronizes itself to the data transfer on the CAN bus by waiting for the occurrence of a sequence of 11 consecutive recessive bits (= Bus Idle) before it can take part in bus activities and start the message transfer.

The initialization of the Message Objects is independent of Init and can be done on the fly, but the Message Objects should all be configured to particular identifiers or set to not valid before the BSP starts the message transfer.

To change the configuration of a Message Object during normal operation, the software has to start by resetting the corresponding MsgVal bit. When the configuration is completed, MsgVal bit is set again.

6.16.5.2 CAN Message Transfer

Once the C_CAN is initialized and Init bit (CAN_CON[0]) is reset to zero, the C_CAN Core synchronizes itself to the CAN bus and starts the message transfer.

Received messages are stored in their appropriate Message Objects if they pass the Message Handler's acceptance filtering. The whole message including all arbitration bits, DLC (CAN_Ifn_MCON[3:0]) and eight data bytes (CAN_Ifn_DAT_A1/2; CAN_Ifn_DAT_B1/2) are stored in the Message Object. If the Identifier Mask is used, the arbitration bits which are masked to "don't care" may be overwritten in the Message Object.

Software can read or write each message any time through the Interface Registers and the Message Handler guarantees data consistency in case of concurrent accesses.

Messages to be transmitted are updated by the application software. If a permanent Message Object (arbitration and control bits are set during configuration) exists for the message, only the data bytes are updated and the TxRqst bit (CAN_Ifn_MCON[8]) with NewDat bit (CAN_Ifn_MCON[15]) are set to start the transmission. If several transmit messages are assigned to the same Message Object (when the number of Message Objects is not sufficient), the whole Message Object has to be configured before the transmission of this message is requested.

The transmission of any number of Message Objects may be requested at the same time. Message objects are transmitted subsequently according to their internal priority. Messages may be updated or set to not valid any time, even when their requested transmission is still pending. The old data will be discarded when a message is updated before its pending transmission has started.

Depending on the configuration of the Message Object, the transmission of a message may be requested autonomously by the reception of a remote frame with a matching identifier.

6.16.5.3 Disabled Automatic Retransmission

In accordance with the CAN Specification (see ISO11898, 6.3.3 Recovery Management), the C_CAN provides means for automatic retransmission of frames that have lost arbitration or have been disturbed by errors during transmission. The frame transmission service will not be confirmed to the user before the transmission is successfully completed. This means that, by default, automatic retransmission is enabled. It can be disabled to enable the C_CAN to work within a Time Triggered CAN (TTCAN, see ISO11898-1) environment.

The Disabled Automatic Retransmission mode is enabled by setting the Disable Automatic Retransmission (DAR bit (CAN_CON[5])) to one. In this operation mode, the programmer has to consider the different behavior of bits TxRqst (CAN_Ifn_MCON[8]) and NewDat (CAN_Ifn_MCON[15]) of the Message Buffers:

- When a transmission starts, bit TxRqst of the respective Message Buffer is cleared, while bit NewDat remains set.
- When the transmission completed successfully, bit NewDat is cleared.
- When a transmission fails (lost arbitration or error), bit NewDat remains set.
- To restart the transmission, the software should set the bit TxRqst again.

6.16.6 Test Mode

Test Mode is entered by setting the Test bit (CAN_CON[7]). In Test Mode, bits Tx1 (CAN_TEST[6]), Tx0 (CAN_TEST[5]), Lback (CAN_TEST[4]), Silent (CAN_TEST[3]) and Basic (CAN_TEST[2]) are writeable. Bit Rx (CAN_TEST[7]) monitors the state of the CAN_RX pin and therefore is only readable. All Test Register functions are disabled when the Test bit is cleared.

6.16.6.1 Silent Mode

The CAN Core can be set in Silent Mode by programming the Silent bit (CAN_TEST[3]) to one. In Silent Mode, the C_CAN is able to receive valid data frames and valid remote frames, but it sends only recessive bits on the CAN bus and it cannot start a transmission. If the CAN Core is required to send a dominant bit (ACK bit, Error Frames), the bit is rerouted internally so that the CAN Core monitors this dominant bit, although the CAN bus may remain in recessive state. The Silent Mode can be used to analysis the traffic on a CAN bus without affecting it by the transmission of dominant bits. The Figure 6-144shows the connection of signals CAN_TX and CAN_RX to the CAN Core in Silent Mode.

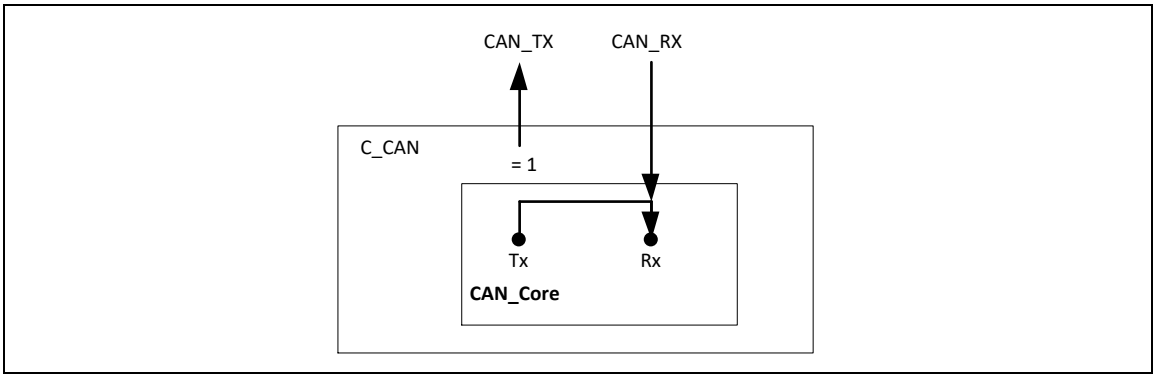


Figure 6-144 CAN Core in Silent Mode

6.16.6.2 Loop Back Mode

The CAN Core can be set in Loop Back Mode by programming the Test Register bit Lback (CAN_TEST[4]) to one. In Loop Back Mode, the CAN Core treats its own transmitted messages as received messages and stores them in a Receive Buffer (if they pass acceptance filtering). The Figure

6-145shows the connection of signals, CAN_TX and CAN_RX, to the CAN Core in Loop Back Mode.

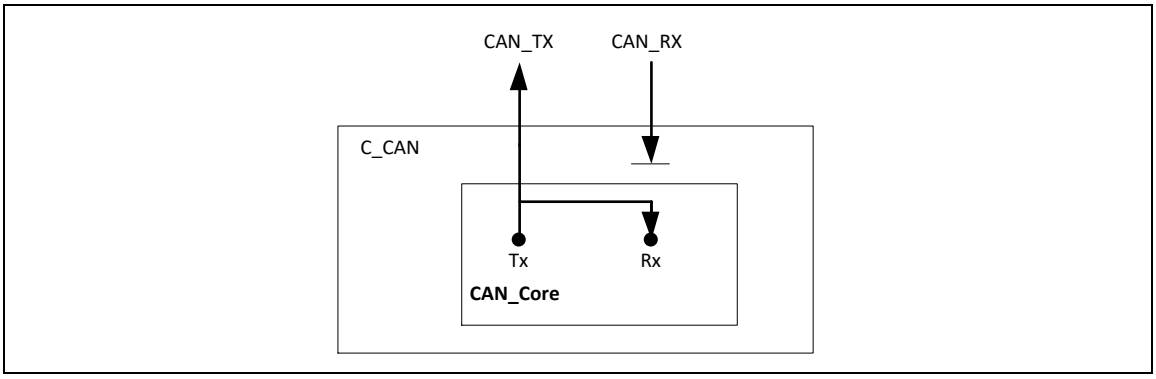


Figure 6-145 CAN Core in Loop Back Mod

This mode is provided for self-test functions. To be independent from external stimulation, the CAN Core ignores acknowledge errors (recessive bit sampled in the acknowledge slot of a data/ remote frame) in Loop Back Mode. In this mode, the CAN Core performs an internal feedback from its Tx output to its Rx input. The actual value of the CAN_RX input pin is disregarded by the CAN Core. The transmitted messages can be monitored on the CAN_TX pin.

6.16.6.3 Loop Back Combined with Silent Mode

It is also possible to combine Loop Back Mode and Silent Mode by programming bits Lback (CAN_TEST[4]) and Silent (CAN_TEST[3]) to one at the same time. This mode can be used for a “Hot Selftest”, which means that C_CAN can be tested without affecting a running CAN system connected to the CAN_TX and CAN_RX pins. In this mode, the CAN_RX pin is disconnected from the CAN Core and the CAN_TX pin is held recessive. The Figure 6-146shows the connection of signals CAN_TX and CAN_RX to the CAN Core in case of the combination of Loop Back Mode with Silent Mode.

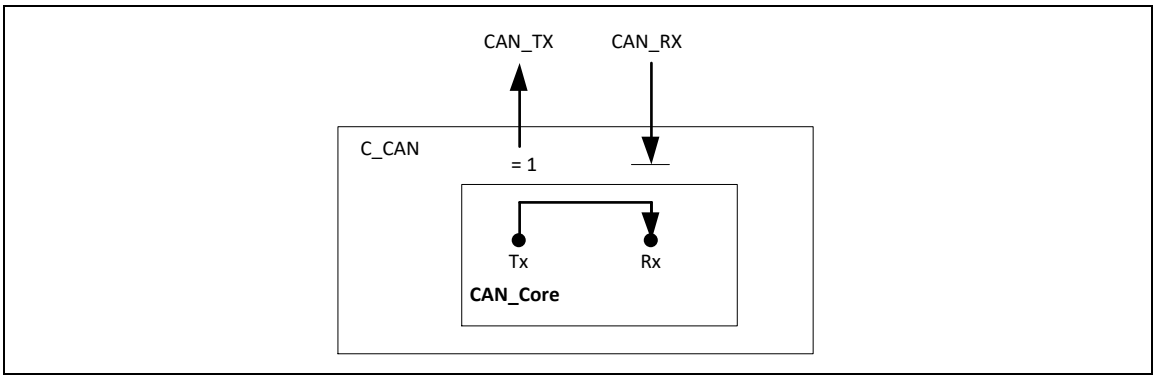


Figure 6-146 CAN Core in Loop Back Mode Combined with Silent Mode

6.16.6.4 Basic Mode

The CAN Core can be set in Basic Mode by programming the Basic bit (CAN_TEST[2]) to one. In this mode, the C_CAN runs without the Message RAM.

The IF1 Registers are used as Transmit Buffer. The transmission of the contents of the IF1 Registers is requested by writing the Busy bit (CAN_Ifn_CREQ[15]) of the IF1 Command Request Register to one. The IF1 Registers are locked while the Busy bit is set. The Busy bit indicates that the transmission is pending.

As soon the CAN bus is idle, the IF1 Registers are loaded into the shift register of the CAN Core and the transmission is started. When the transmission has been completed, the Busy bit is reset and the

locked IF1 Registers are released.

A pending transmission can be aborted at any time by resetting the Busy bit in the IF1 Command Request Register while the IF1 Registers are locked. If the software has reset the Busy bit, a possible retransmission in case of lost arbitration or in case of an error is disabled.

The IF2 Registers are used as a Receive Buffer. After the reception of a message the contents of the shift register is stored into the IF2 Registers, without any acceptance filtering.

Additionally, the actual contents of the shift register can be monitored during the message transfer. Each time a read Message Object is initiated by writing the Busy bit of the IF2 Command Request Register to one, the contents of the shift register are stored into the IF2 Registers.

In Basic Mode, the evaluation of all Message Object related control and status bits and the control bits of the Ifn Command Mask Registers are turned off. The message number of the Command request registers is not evaluated. The NewDat (CAN_Ifn_MCON[15]) and MsgLst (CAN_Ifn_MCON[14]) bits retain their function, DLC3-0 indicates the received DLC (CAN_Ifn_MCON[3:0]), and the other control bits are read as '0'.

6.16.6.5 Software Control of CAN_TX Pin

Four output functions are available for the CAN transmit pin, CAN_TX. In addition to its default function (serial data output), the CAN transmit pin can drive the CAN Sample Point signal to monitor CAN_Core's bit timing and it can drive constant dominant or recessive values. The latter two functions, combined with the readable CAN receive pin CAN_RX, can be used to check the physical layer of the CAN bus.

The output mode for the CAN_TX pin is selected by programming the Tx1 (CAN_TEST[6]) and Tx0 (CAN_TEST[5]) bits.

The three test functions of the CAN_TX pin interfere with all CAN protocol functions. CAN_TX must be left in its default function when CAN message transfer or any of the test modes (Loop Back Mode, Silent Mode or Basic Mode) are selected.

6.16.7 CAN Communications

6.16.7.1 Managing Message Objects

The configuration of the Message Objects in the Message RAM (with the exception of the bits MsgVal, NewDat, IntPnd and TxRqst) will not be affected by resetting the chip. All the Message Objects must be initialized by the application software or they must be "not valid" (MsgVal bit = '0') and the bit timing must be configured before the application software clears the Init bit (CAN_CON[0]).

The configuration of a Message Object is done by programming Mask, Arbitration, Control and Data fields of one of the two interface registers to the desired values. By writing to the corresponding Ifn Command Request Register, the Ifn Message Buffer Registers are loaded into the addressed Message Object in the Message RAM.

When the Init bit is cleared, the CAN Protocol Controller state machine of the CAN_Core and the state machine of the Message Handler control the internal data flow of the C_CAN. Received messages that pass the acceptance filtering are stored into the Message RAM, messages with pending transmission request are loaded into the CAN_Core's Shift Register and are transmitted through the CAN bus.

The application software reads received messages and updates messages to be transmitted through the Ifn Interface Registers. Depending on the configuration, the application software is interrupted on certain CAN message and CAN error events.

6.16.7.2 Message Handler State Machine

The Message Handler controls the data transfer between the Rx/Tx Shift Register of the CAN Core, the Message RAM and the Ifn Registers.

The Message Handler FSM controls the following functions:

- Data Transfer from Ifn Registers to the Message RAM
- Data Transfer from Message RAM to the Ifn Registers
- Data Transfer from Shift Register to the Message RAM
- Data Transfer from Message RAM to Shift Register
- Data Transfer from Shift Register to the Acceptance Filtering unit
- Scanning of Message RAM for a matching Message Object
- Handling of TxRqst flags
- Handling of interrupts.

6.16.7.3 Data Transfer from/to Message RAM

When the application software initiates a data transfer between the Ifn Registers and Message RAM, the Message Handler sets the Busy bit (CAN_Ifn_CREQ[15]) to ‘1’. After the transfer has completed, the Busy bit is again cleared (see theFigure 6-147).

The respective Command Mask Register specifies whether a complete Message Object or only parts of it will be transferred. Due to the structure of the Message RAM, it is not possible to write single bits/bytes of one Message Object. It is always necessary to write a complete Message Object into the Message RAM. Therefore, the data transfer from the Ifn Registers to the Message RAM requires a read-modify-write cycle. First, those parts of the Message Object that are not to be changed are read from the Message RAM and then the complete contents of the Message Buffer Registers are written into the Message Object.

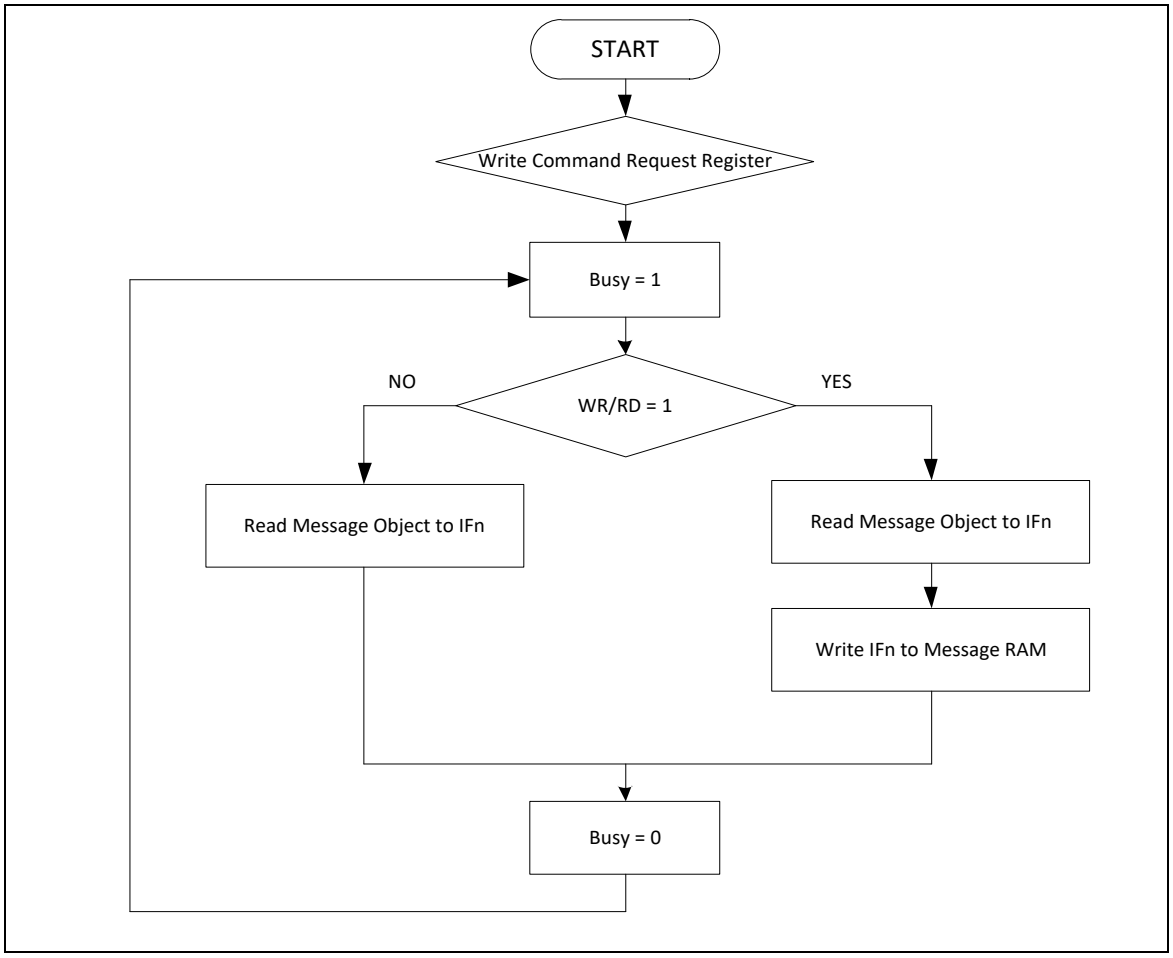


Figure 6-147 Data transfer between Ifn Registers and Message

After a partial write of a Message Object, the Message Buffer Registers that are not selected in the Command Mask Register will set the actual contents of the selected Message Object.

After a partial read of a Message Object, the Message Buffer Registers that are not selected in the Command Mask Register will be left unchanged.

6.16.7.4 Message Transmission

If the shift register of the CAN Core cell is ready for loading and if there is no data transfer between the Ifn Registers and Message RAM, the MsgVal bit (CAN_Ifn_ARB2[15]) and TxRqst bits (CAN_TXREQ1/2) are evaluated. The valid Message Object with the highest priority pending transmission request is loaded into the shift register by the Message Handler and the transmission is started. The NewDat (CAN_Ifn_MCON[15]) bit of the Message Object is reset.

After a successful transmission and also if no new data was written to the Message Object (NewDat = '0') since the start of the transmission, the TxRqst bit of the Message Control register (CAN_Ifn_MCON[8]) will be reset. If TxIE bit (CAN_Ifn_MCON[11]) is set, IntPnd bit (CAN_Ifn_MCON[13]) of the Interrupt Identifier register will be set after a successful transmission. If the C_CAN has lost the arbitration or if an error occurred during the transmission, the message will be retransmitted as soon as the CAN bus is free again. Meanwhile, if the transmission of a message with higher priority has been requested, the messages will be transmitted in the order of their priority.

6.16.7.5 Acceptance Filtering of Received Messages

When the arbitration and control field (Identifier + IDE + RTR + DLC) of an incoming message is completely shifted into the Rx/Tx Shift Register of the CAN Core, the Message Handler FSM starts the scanning of the Message RAM for a matching valid Message Object.

To scan the Message RAM for a matching Message Object, the Acceptance Filtering unit is loaded with the arbitration bits from the CAN Core shift register. The arbitration and mask fields (including MsgVal (CAN_Ifn_ARB2[15]), Umask (CAN_Ifn_MCON[12]), NewDat (CAN_Ifn_MCON[15]) and EoB (CAN_Ifn_MCON[7])) of Message Object 1 are then loaded into the Acceptance Filtering unit and compared with the arbitration field from the shift register. This is repeated with each following Message Object until a matching Message Object is found or until the end of the Message RAM is reached.

If a match occurs, the scan is stopped and the Message Handler FSM proceeds depending on the type of frame (Data Frame or Remote Frame) received.

Reception of Data Frame

The Message Handler FSM stores the message from the CAN Core shift register into the respective Message Object in the Message RAM. Not only the data bytes, but all arbitration bits and the Data Length Code are stored into the corresponding Message Object. This is done to keep the data bytes connected with the identifier even if arbitration mask registers are used.

The NewDat bit (CAN_Ifn_MCON[15]) is set to indicate that new data (not yet seen by the software) has been received. The application software should reset NewDat bit when the Message Object has been read. If at the time of reception, the NewDat bit was already set, MsgLst (CAN_Ifn_MCON[14]) is set to indicate that the previous data (supposedly not seen by the software) is lost. If the RxIE bit (CAN_Ifn_MCON[10]) is set, the IntPnd bit (CAN_Ifn_MCON[13]) is set, causing the Interrupt Register to point to this Message Object.

The TxRqst bit (CAN_Ifn_MCON[8]) of this Message Object is reset to prevent the transmission of a Remote Frame, while the requested Data Frame has just been received.

Reception of Remote Frame

When a Remote Frame is received, three different configurations of the matching Message Object have to be considered:

4. Dir (CAN_Ifn_ARB2[13]) = ‘1’ (direction = transmit), RmtEn (CAN_Ifn_MCON[9]) = ‘1’ and Umask (CAN_Ifn_MCON[12]) = ‘1’ or’0’

At the reception of a matching Remote Frame, the TxRqst bit of this Message Object is set. The rest of the Message Object remains unchanged.

2) Dir = ‘1’ (direction = transmit), RmtEn = ‘0’ and Umask =’0’

At the reception of a matching Remote Frame, the TxRqst bit of this Message Object remains unchanged; the Remote Frame is ignored.

3) Dir = ‘1’ (direction = transmit), RmtEn = ‘0’ and Umask =’1’

At the reception of a matching Remote Frame, the TxRqst bit of this Message Object is reset. The arbitration and control field (Identifier + IDE + RTR + DLC) from the shift register is stored in the Message Object of the Message RAM and the NewDat bit (CAN_Ifn_MCON[15]) of this Message Object is set. The data field of the Message Object remains unchanged; the Remote Frame is treated similar to a received Data Frame.

6.16.7.6 Receive/Transmit Priority

The receive/transmit priority for the Message Objects is attached to the message number. Message Object 1 has the highest priority, while Message Object 32 has the lowest priority. If more than one transmission request is pending, they are serviced due to the priority of the corresponding Message Object

6.16.7.7 Configuring a Transmit Object

The Table 6-24shows how a Transmit Object should be initialized.

Ms	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	Appl.	Appl.	1	1	0	0	0	appl.	0	appl.	0

Table 6-24InitializationofaTransmitObject

Note: appl. = application software.

The Arbitration Register values (ID28-0 (CAN_Ifn_ARB1/2) and Xtd bit (CAN_Ifn_ARB2[14])) are provided by the application. They define the identifier and type of the outgoing message. If an 11-bit Identifier (“Standard Frame”) is used, it is programmed to ID28 – ID18. The ID17 – ID0 can then be disregarded.

If the TxIE bit (CAN_Ifn_MCON[11]) is set, the IntPnd bit (CAN_Ifn_MCON[13]) will be set after a successful transmission of the Message Object.

If the RmtEn bit (CAN_Ifn_MCON[9]) is set, a matching received Remote Frame will cause the TxRqst bit (CAN_Ifn_MCON[8]) to be set; the Remote Frame will autonomously be answered by a Data Frame.

The Data Register values (DLC3-0 (CAN_Ifn_MCON[3:0]), Data(0)-(7)) are provided by the application, TxRqst and RmtEn may not be set before the data is valid.

The Mask Registers (Msk28-0, Umask, MXtd and Mdir bits) may be used (Umask (CAN_Ifn_MCON[12]) = ’1’) to allow groups of Remote Frames with similar identifiers to set the TxRqst bit. The Dir bit (CAN_Ifn_ARB2[13]) should not be masked.

6.16.7.8 Updating a Transmit Object

The software may update the data bytes of a Transmit Object any time through the Ifn Interface registers, neither MsgVal bit (CAN_Ifn_ARB2[15]) nor TxRqst (CAN_Ifn_MCON[8]) have to be reset before the update.

Even if only a part of the data bytes are to be updated, all four bytes of the corresponding Ifn Data A Register or Ifn Data B Register have to be valid before the contents of that register are transferred to the Message Object. Either the application software has to write all four bytes into the Ifn Data Register or the Message Object is transferred to the Ifn Data Register before the software writes the new data bytes.

When only the (eight) data bytes are updated, first 0x0087 is written to the Command Mask Register and then the number of the Message Object is written to the Command Request Register, concurrently updating the data bytes and setting TxRqst.

To prevent the reset of TxRqst at the end of a transmission that may already be in progress while the data is updated, NewDat (CAN_Ifn_MCON[15]) has to be set together with TxRqst.

When NewDat is set together with TxRqst, NewDat will be reset as soon as the new transmission has

started.

6.16.7.9 Configuring a Receive Object

The Table 6-25shows how a Receive Object should be initialized.

MsgVal	Arb	Data	Mask	EoB	Dir	NewDat	MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
1	appl.	Appl.	Appl.	1	0	0	0	appl.	0	0	0	0

Table 6-25InitializationofaReceiveObject

The Arbitration Registers values (ID28-0 (CAN_Ifn_ARB1/2) and Xtd bit (CAN_Ifn_ARB2[14])) are provided by the application. They define the identifier and type of accepted received messages. If an 11-bit Identifier (“Standard Frame”) is used, it is programmed to ID28 – ID18. Then ID17 – ID0 can be disregarded. When a Data Frame with an 11-bit Identifier is received, ID17 – ID0 will be set to ‘0’.

If the RxIE bit (CAN_Ifn_MCON[10]) is set, the IntPnd bit (CAN_Ifn_MCON[13]) will be set when a received Data Frame is accepted and stored in the Message Object.

The Data Length Code (DLC3-0 (CAN_Ifn_MCON[3:0])) is provided by the application. When the Message Handler stores a Data Frame in the Message Object, it will store the received Data Length Code and eight data bytes. If the Data Length Code is less than 8, the remaining bytes of the Message Object will be overwritten by unspecified values.

The Mask Registers (Msk28-0, Umask, MXtd and Mdir bits) may be used (Umask (CAN_Ifn_MCON[12]) = ’1’) to allow groups of Data Frames with similar identifiers to be accepted. The Dir bit (CAN_Ifn_ARB2[13]) should not be masked in typical applications.

6.16.7.10 Handling Received Messages

The application software may read a received message any time through the Ifn Interface registers. The data consistency is guaranteed by the Message Handler state machine.

Typically, the software will write first 0x007F to the Command Mask Register and then the number of the Message Object to the Command Request Register. This combination will transfer the whole received message from the Message RAM into the Message Buffer Register. Additionally, the bits NewDat (CAN_Ifn_MCON[15])and IntPnd (CAN_Ifn_MCON[13]) are cleared in the Message RAM (not in the Message Buffer).

If the Message Object uses masks for acceptance filtering, the arbitration bits show which of the matching messages have been received.

The actual value of NewDat shows whether a new message has been received since the last time this Message Object was read. The actual value of MsgLst (CAN_Ifn_MCON[14]) shows whether more than one message has been received since the last time this Message Object was read. MsgLst will not be automatically reset.

By means of a Remote Frame, the software may request another CAN node to provide new data for a receive object. Setting the TxRqst bit (CAN_Ifn_MCON[8]) of a receive object will cause the transmission of a Remote Frame with the receive object’s identifier. This Remote Frame triggers the other CAN node to start the transmission of the matching Data Frame. If the matching Data Frame is received before the Remote Frame could be transmitted, the TxRqst bit is automatically reset.

6.16.7.11 Configuring a FIFO Buffer

With the exception of the EoB bit (CAN_Ifn_MCON[7]), the configuration of Receive Objects belonging to a FIFO Buffer is the same as the configuration of a (single) Receive Object, see Section 6.16.7.9: Configuring a Receive Object.

To concatenate two or more Message Objects into a FIFO Buffer, the identifiers and masks (if used) of these Message Objects have to be programmed to matching values. Due to the implicit priority of the Message Objects, the Message Object with the lowest number will be the first Message Object of the FIFO Buffer. The EoB bit of all Message Objects of a FIFO Buffer except the last have to be programmed to zero. The EoB bit of the last Message Object of a FIFO Buffer is set to one, configuring it as the End of the Block.

6.16.7.12 Receiving Messages with FIFO Buffers

Received messages with identifiers matching to a FIFO Buffer are stored into a Message Object of this FIFO Buffer starting with the Message Object with the lowest message number.

When a message is stored into a Message Object of a FIFO Buffer, the NewDat bit (CAN_Ifn_MCON[15]) of this Message Object is set. By setting NewDat while EoB (CAN_Ifn_MCON[7]) is zero, the Message Object is locked for further write access by the Message Handler until the application software has written the NewDat bit back to zero.

Messages are stored into a FIFO Buffer until the last Message Object of this FIFO Buffer is reached. If none of the preceding Message Objects is released by writing NewDat to zero, all further messages for this FIFO Buffer will be written into the last Message Object of the FIFO Buffer and therefore overwrite the previous messages.

6.16.7.13 Reading from a FIFO Buffer

When the application software transfers the contents of a Message Object to the Ifn Message Buffer register by writing its number to the Ifn Command Request Register, the corresponding Command Mask Register should be programmed in such a way that bits NewDat (CAN_Ifn_MCON[15]) and IntPnd (CAN_Ifn_MCON[13]) are reset to zero (TxRqst/NewDat (CAN_Ifn_CMASK[2]) = ‘1’ and ClrIntPnd (CAN_Ifn_CMASK[3]) = ‘1’). The values of these bits in the Message Control Register always reflect the status before resetting the bits.

To assure the correct function of a FIFO Buffer, the application software should read the Message Objects starting at the FIFO Object with the lowest message number.

The Figure 6-148shows how a set of Message Objects which are concatenated to a FIFO Buffer can be handled by the application software.

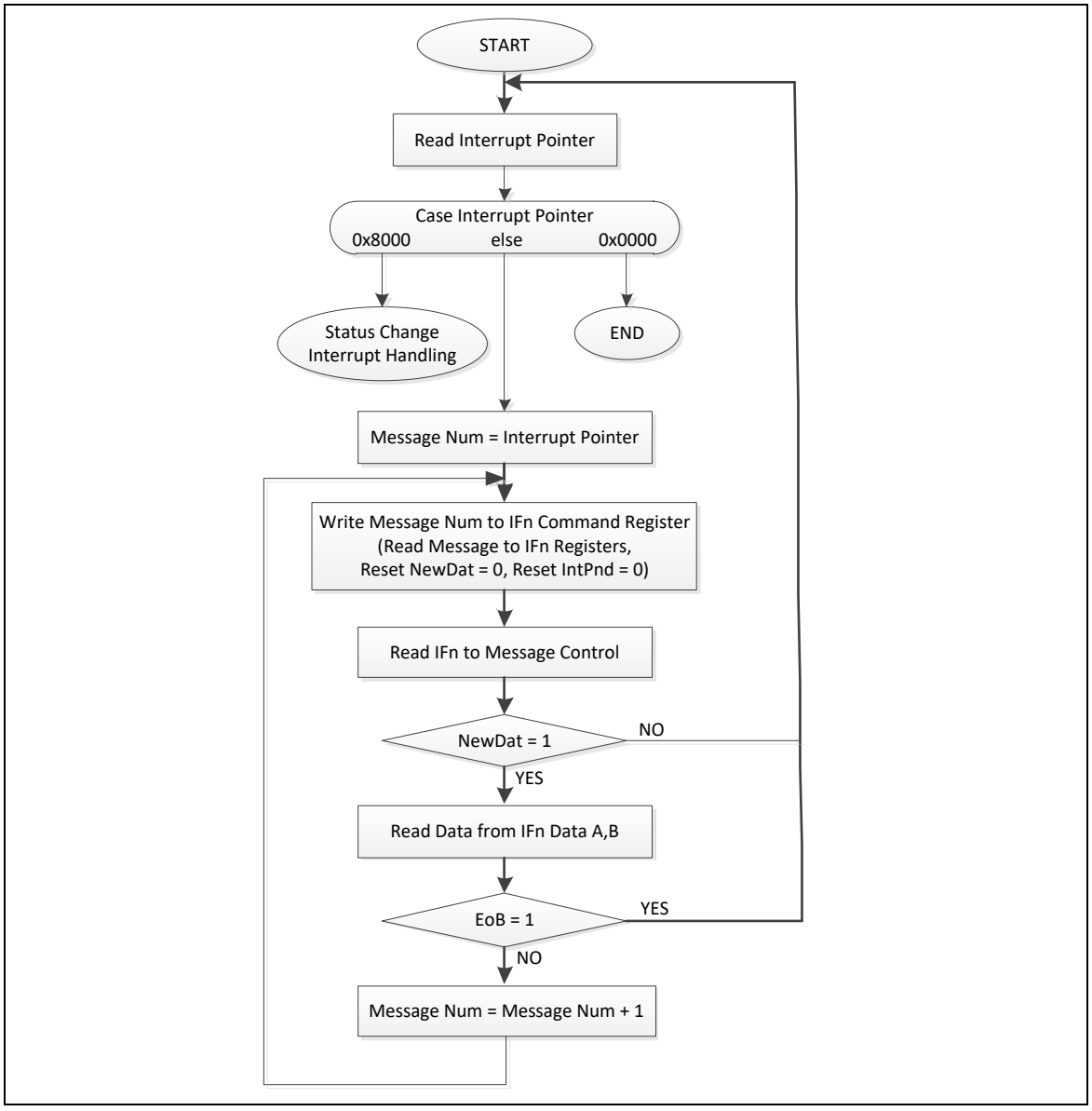


Figure 6-148 Application Software Handling of a FIFO Buffer

6.16.7.14 Handling Interrupts

If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the application software has cleared it.

The Status Interrupt has the highest priority. Among the message interrupts, interrupt priority of the Message Object decreases with increasing message number.

A message interrupt is cleared by clearing the IntPnd bit (CAN_Ifn_MCON[13]) of the Message Object. The Status Interrupt is cleared by reading the Status Register.

The interrupt identifier, IntId, in the Interrupt Register, indicates the cause of the interrupt. When no interrupt is pending, the register will hold the value zero. If the value of the Interrupt Register is different from zero, then there is an interrupt pending and, if IE (CAN_Ifn_CON[1]) is set, the CAN_INT interrupt signal is active. The interrupt remains active until the Interrupt Register is back to value zero (the cause of the interrupt is reset) or until IE is reset.

The value 0x8000 indicates that an interrupt is pending because the CAN Core has updated (not necessarily changed) the Status Register (Error Interrupt or Status Interrupt). This interrupt has the highest priority. The application software can update (reset) the status bits RxOk (CAN_STATUS[4]), TxOk (CAN_STATUS[3]) and LEC (CAN_STATUS[2:0]), but a write access of the software to the Status Register can never generate or reset an interrupt.

All other values indicate that the source of the interrupt is one of the Message Objects. IntId points to the pending message interrupt with the highest interrupt priority.

The application software controls whether a change of the Status Register may cause an interrupt (bits EIE (CAN_Ifn_MCON[3]) and SIE (CAN_Ifn_MCON[2])) and whether the interrupt line becomes active when the Interrupt Register is different from zero (bit IE in the CAN Control Register). The Interrupt Register will be updated even when IE is reset.

The application software has two possibilities to follow the source of a message interrupt. First, it can follow the IntId in the Interrupt Register and second it can poll the Interrupt Pending Register.

An interrupt service routine that is reading the message that is the source of the interrupt may read the message and reset the Message Object's IntPnd at the same time (bit ClrIntPnd (CAN_Ifn_CMASK[3])). When IntPnd is cleared, the Interrupt Register will point to the next Message Object with a pending interrupt.

6.16.7.15 Configuring the Bit Timing

Even if minor errors in the configuration of the CAN bit timing do not result in immediate failure, the performance of a CAN network can be reduced significantly.

In many cases, the CAN bit synchronization will amend a faulty configuration of the CAN bit timing to such a degree that only occasionally an error frame is generated. However, in the case of arbitration, when two or more CAN nodes simultaneously try to transmit a frame, a misplaced sample point may cause one of the transmitters to become error passive.

The analysis of such sporadic errors requires a detailed knowledge of the CAN bit synchronization inside a CAN node and interaction of the CAN nodes on the CAN bus.

6.16.7.16 Bit Time and Bit Rate

CAN supports bit rates in the range of lower than 1 Kbit/s up to 1000 Kbit/s. Each member of the CAN network has its own clock generator, usually a quartz oscillator. The timing parameter of the bit time (i.e. the reciprocal of the bit rate) can be configured individually for each CAN node, creating a common bit rate even though the oscillator periods of the CAN nodes (f_{osc}) may be different.

The frequencies of these oscillators are not absolutely stable, small variations are caused by changes in temperature or voltage and by deteriorating components. As long as the variations remain inside a specific oscillator tolerance range (d_f), the CAN nodes are able to compensate for the different bit rates by re-synchronizing to the bit stream.

According to the CAN specification, the bit time is divided into four segments (see theFigure 6-149). The Synchronization Segment, the Propagation Time Segment, the Phase Buffer Segment 1 and the Phase Buffer Segment 2. Each segment consists of a specific, programmable number of time quanta (see Table 6-26 CAN Bit Time Parameters). The length of the time quantum (t_q), which is the basic time unit of the bit time, is defined by the CAN controller's APB clock f_{APB} and the BRP bit (CAN_BTIME[5:0]) : t_q = BRP / f_{APB}.

The Synchronization Segment, Sync_Seg, is that part of the bit time where edges of the CAN bus level are expected to occur. The distance between an edge that occurs outside of Sync_Seg, and the Sync_Seg is called the phase error of that edge. The Propagation Time Segment, Prop_Seg, is intended to compensate for the physical delay time within the CAN network. The Phase Buffer Segments Phase_Seg1 and Phase_Seg2 surround the Sample Point. The (Re-)Synchronization Jump Width (SJW) defines how far a re-synchronization may move the Sample Point inside the limits

defined by the Phase Buffer Segments to compensate for edge phase errors.

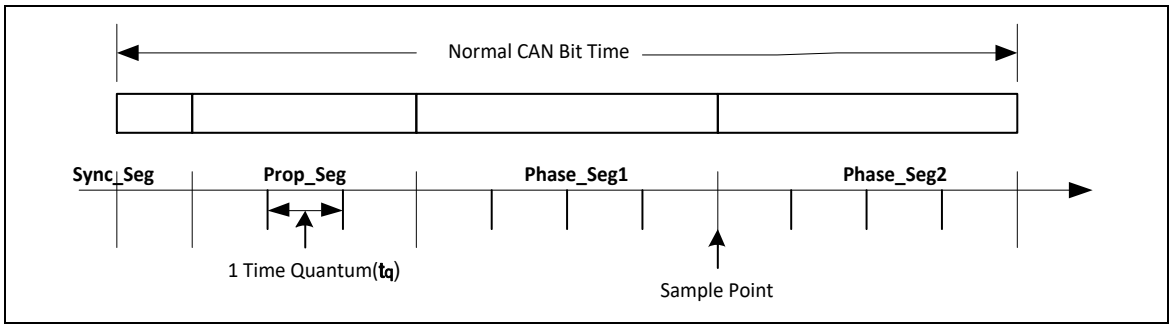


Figure 6-149Bit Timing

Parameter	Range	Remark
BRP	[1..32]	Defines the length of the time quantum t_q
Sync_Seg	1 t_q	Fixed length, synchronization of bus input to APB clock
Prop_Seg	[1..8] t_q	Compensates for the physical delay time
Phase_Seg1	[1..8] t_q	Which may be lengthened temporarily by synchronization
Phase_Seg2	[1..8] t_q	Which may be shortened temporarily by synchronization
SJW	[1..4] t_q	Which may not be longer than either Phase Buffer Segment
This table describes the minimum programmable ranges required by the CAN protocol		

Table 6-26 CAN Bit Time Parameters

A given bit rate may be met by different bit time configurations, but for the proper function of the CAN network the physical delay time and the oscillator's tolerance range have to be considered.

6.16.7.17 Propagation Time Segment

This part of the bit time is used to compensate physical delay time within the network. These delay time consist of the signal propagation time on the bus and the internal delay time of the CAN nodes.

Any CAN node synchronized to the bit stream on the CAN bus will be out of phase with the transmitter of that bit stream, caused by the signal propagation time between the two nodes. The CAN protocol's non-destructive bitwise arbitration and the dominant acknowledge bit provided by receivers of CAN messages requires that a CAN node transmitting a bit stream must also be able to receive dominant bits transmitted by other CAN nodes that are synchronized to that bit stream. The example in the Figure 6-150shows the phase shift and propagation time between two CAN nodes.

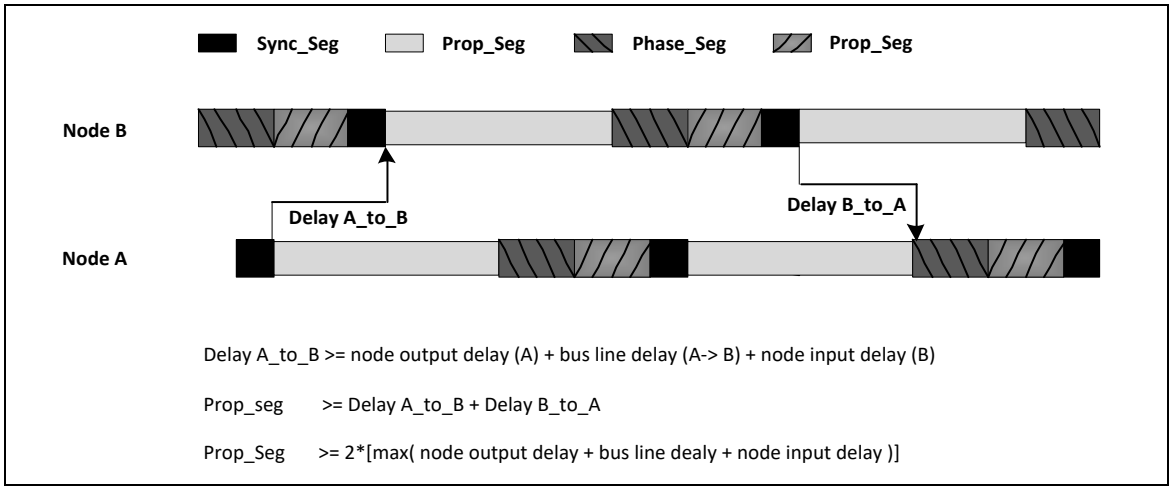


Figure 6-150 Propagation Time Segment

In this example, both nodes A and B are transmitters, performing an arbitration for the CAN bus. Node A has sent its Start of Frame bit less than one bit time earlier than node B, therefore node B has synchronized itself to the received edge from recessive to dominant. Since node B has received this edge delay (A_to_B) after it has been transmitted, B's bit timing segments are shifted with respect to A. Node B sends an identifier with higher priority and so it will win the arbitration at a specific identifier bit when it transmits a dominant bit while node A transmits a recessive bit. The dominant bit transmitted by node B will arrive at node A after the delay (B_to_A).

Due to oscillator tolerances, the actual position of node A's Sample Point can be anywhere inside the nominal range of node A's Phase Buffer Segments, so the bit transmitted by node B must arrive at node A before the start of Phase_Seg1. This condition defines the length of Prop_Seg.

If the edge from recessive to dominant transmitted by node B arrives at node A after the start of Phase_Seg1, it can happen that node A samples a recessive bit instead of a dominant bit, resulting in a bit error and the destruction of the current frame by an error flag.

The error occurs only when two nodes arbitrate for the CAN bus that have oscillators of opposite ends of the tolerance range and that are separated by a long bus line. This is an example of a minor error in the bit timing configuration (Prop_Seg is too short) that causes sporadic bus errors.

Some CAN implementations provide an optional 3 Sample Mode but the C_CAN does not. In this mode, the CAN bus input signal passes a digital low-pass filter, using three samples and a majority logic to determine the valid bit value. This results in an additional input delay of 1 t_q, requiring a longer Prop_Seg.

6.16.7.18 Phase Buffer Segments and Synchronization

The Phase Buffer Segments (Phase_Seg1 and Phase_Seg2) and the Synchronization Jump Width (SJW) are used to compensate for the oscillator tolerance. The Phase Buffer Segments may be lengthened or shortened by synchronization.

Synchronizations occur on edges from recessive to dominant, their purpose is to control the distance between edges and Sample Points.

Edges are detected by sampling the actual bus level in each time quantum and comparing it with the bus level at the previous Sample Point. A synchronization may be done only if a recessive bit was sampled at the previous Sample Point and if the bus level at the actual time quantum is dominant.

An edge is synchronous if it occurs inside of Sync_Seg, otherwise the distance between edge and the end of Sync_Seg is the edge phase error, measured in time quanta. If the edge occurs before Sync_Seg, the phase error is negative, else it is positive.

Two types of synchronization exist, Hard Synchronization and Re-synchronization.

A Hard Synchronization is done once at the start of a frame and inside a frame only when Re-synchronizations occur.

- Hard Synchronization

After a hard synchronization, the bit time is restarted with the end of Sync_Seg, regardless of the edge phase error. Thus hard synchronization forces the edge, which has caused the hard synchronization to lie within the synchronization segment of the restarted bit time.

- Bit Re-synchronization

Re-synchronization leads to a shortening or lengthening of the bit time such that the position of the sample point is shifted with regard to the edge.

When the phase error of the edge which causes Re-synchronization is positive, Phase_Seg1 is lengthened. If the magnitude of the phase error is less than SJW, Phase_Seg1 is lengthened by the magnitude of the phase error, else it is lengthened by SJW.

When the phase error of the edge, which causes Re-synchronization is negative, Phase_Seg2 is shortened. If the magnitude of the phase error is less than SJW, Phase_Seg2 is shortened by the magnitude of the phase error, else it is shortened by SJW.

When the magnitude of the phase error of the edge is less than or equal to the programmed value of SJW, the results of Hard Synchronization and Re-synchronization are the same. If the magnitude of the phase error is larger than SJW, the Re-synchronization cannot compensate the phase error completely, an error (phase error – SJW) remains.

Only one synchronization may be done between two Sample Points. The Synchronizations maintain a minimum distance between edges and Sample Points, giving the bus level time to stabilize and filtering out spikes that are shorter than (Prop_Seg + Phase_Seg1).

Apart from noise spikes, most synchronizations are caused by arbitration. All nodes synchronize “hard” on the edge transmitted by the “leading” transceiver that started transmitting first, but due to propagation delay time, they cannot become ideally synchronized. The “leading” transmitter does not necessarily win the arbitration, therefore the receivers have to synchronize themselves to different transmitters that subsequently “take the lead” and that are differently synchronized to the previously “leading” transmitter. The same happens at the acknowledge field, where the transmitter and some of the receivers will have to synchronize to that receiver that “takes the lead” in the transmission of the dominant acknowledge bit.

Synchronizations after the end of the arbitration will be caused by oscillator tolerance, when the differences in the oscillator’s clock periods of transmitter and receivers sum up during the time between synchronizations (at most ten bits). These summarized differences may not be longer than the SJW, limiting the oscillator’s tolerance range.

The examples in the Figure 6-151shows how the Phase Buffer Segments are used to compensate for phase errors. There are three drawings of each two consecutive bit timings. The upper drawing shows the synchronization on a “late” edge, the lower drawing shows the synchronization on an “early” edge, and the middle drawing is the reference without synchronization.

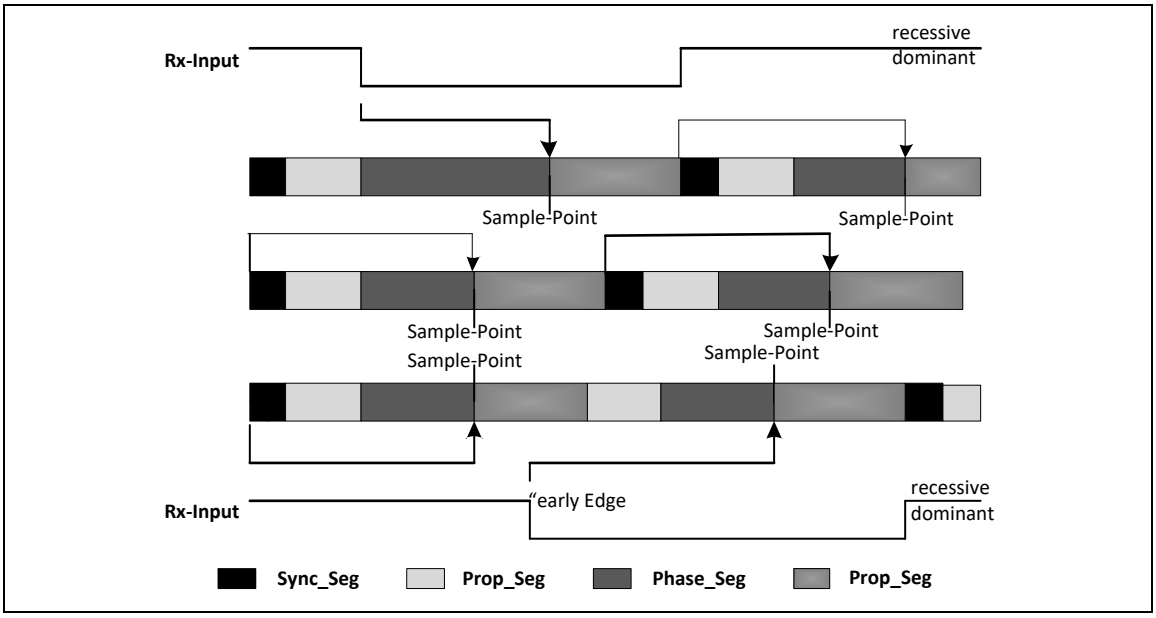


Figure 6-151 Synchronization on "late" and "early" Edges

In the first example an edge from recessive to dominant occurs at the end of Prop_Seg. The edge is "late" since it occurs after the Sync_Seg. Reacting to the "late" edge, Phase_Seg1 is lengthened so that the distance from the edge to the Sample Point is the same as it would have been from the Sync_Seg to the Sample Point if no edge had occurred. The phase error of this "late" edge is less than SJW, so it is fully compensated and the edge from dominant to recessive at the end of the bit, which is one nominal bit time long, occurs in the Sync_Seg.

In the second example an edge from recessive to dominant occurs during Phase_Seg2. The edge is "early" since it occurs before a Sync_Seg. Reacting to the "early" edge, Phase_Seg2 is shortened and Sync_Seg is omitted, so that the distance from the edge to the Sample Point is the same as it would have been from an Sync_Seg to the Sample Point if no edge had occurred. As in the previous example, the magnitude of this "early" edge's phase error is less than SJW, so it is fully compensated.

The Phase Buffer Segments are lengthened or shortened temporarily only; at the next bit time, the segments return to their nominal programmed values.

In these examples, the bit timing is seen from the point of view of the CAN implementation's state machine, where the bit time starts and ends at the Sample Points. The state machine omits Sync_Seg when synchronising on an "early" edge because it cannot subsequently redefine that time quantum of Phase_Seg2 where the edge occurs to be the Sync_Seg.

The examples in the Figure 6-152shows how short dominant noise spikes are filtered by synchronisations. In both examples the spike starts at the end of Prop_Seg and has the length of (Prop_Seg + Phase_Seg1).

In the first example, the Synchronization Jump Width is greater than or equal to the phase error of the spike's edge from recessive to dominant. Therefore the Sample Point is shifted after the end of the spike; a recessive bus level is sampled.

In the second example, SJW is shorter than the phase error, so the Sample Point cannot be shifted far enough; the dominant spike is sampled as actual bus level.

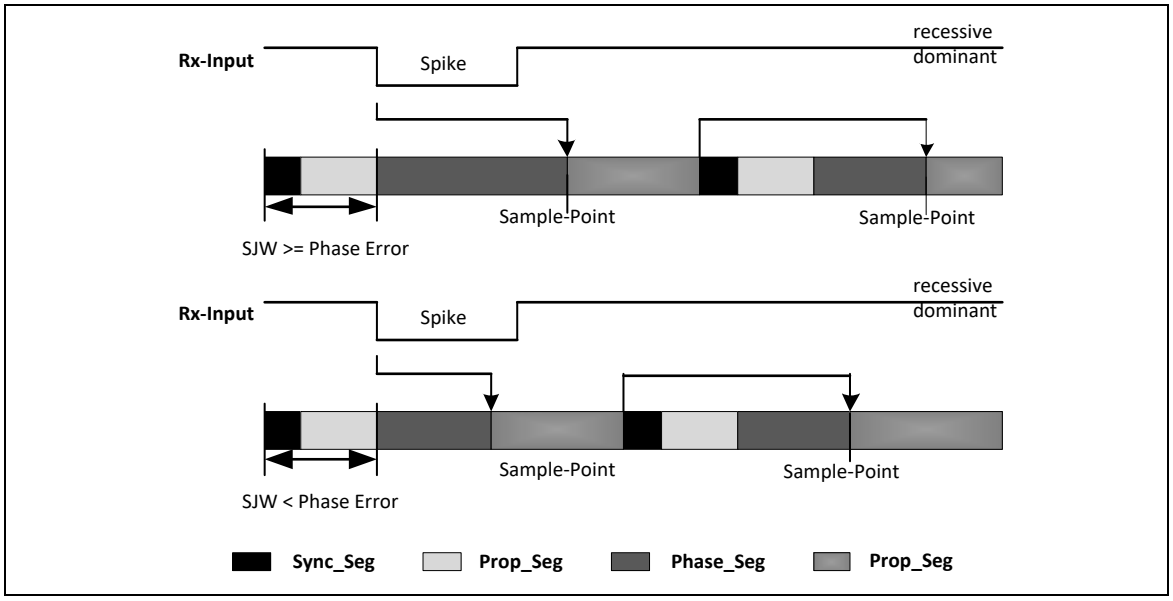


Figure 6-152 Filtering of Short Dominant Spikes

6.16.7.19 Oscillator Tolerance Range

The oscillator tolerance range was increased when the CAN protocol was developed from version 1.1 to version 1.2 (version 1.0 was never implemented in silicon). The option to synchronize on edges from dominant to recessive became obsolete, only edges from recessive to dominant are considered for synchronization. The protocol update to version 2.0 (A and B) had no influence on the oscillator tolerance.

The tolerance range df for an oscillator frequency f_{osc} around the nominal frequency f_{nom} is:

$$(1 - d_f) \cdot f_{nom} <= f_{osc} <= (1 + d_f) \cdot f_{nom}$$

It depends on the proportions of Phase_Seg1, Phase_Seg2, SJW and the bit time. The maximum tolerance df is the defined by two conditions (both shall be met):

$$\text{I: } d_f <= \frac{\min(\text{Phase\_Seg1}, \text{Phase\_Seg2})}{2 * (13 * \text{bit\_time} - \text{Phase\_Seg2})}$$

$$\text{II: } d_f <= \frac{\text{SJW}}{20 * \text{bit\_time}}$$

Note: These conditions base on the APB cock = f_{osc}.

It has to be considered that SJW may not be larger than the smaller of the Phase Buffer Segments and that the Propagation Time Segment limits that part of the bit time that may be used for the Phase Buffer Segments.

The combination Prop_Seg = 1 and Phase_Seg1 = Phase_Seg2 = SJW = 4 allows the largest possible oscillator tolerance of 1.58%. This combination with a Propagation Time Segment of only 10% of the bit time is not suitable for short bit times; it can be used for bit rates of up to 125 Kbit/s (bit

time = 8us) with a bus length of 40 m.

6.16.7.20 Configuring the CAN Protocol Controller

In most CAN implementations and also in the C_CAN, the bit timing configuration is programmed in two register bytes. The sum of Prop_Seg and Phase_Seg1 (as TSEG1 (CAN_BTIME[11:8])) is combined with Phase_Seg2 (as TSEG2 (CAN_BTIME[14:12])) in one byte, SJW (CAN_BTIME[7:6]) and BRP (CAN_BTIME[5:0]) are combined in the other byte.

In these bit timing registers, the four components TSEG1, TSEG2, SJW and BRP have to be programmed to a numerical value that is one less than its functional value. Therefore, instead of values in the range of [1..n], values in the range of [0..n-1] are programmed. That way, e.g. SJW (functional range of [1..4]) is represented by only two bits.

Therefore the length of the bit time is (programmed values) [TSEG1 + TSEG2 + 3] t_q or (functional values) [Sync_Seg + Prop_Seg + Phase_Seg1 + Phase_Seg2] t_q.

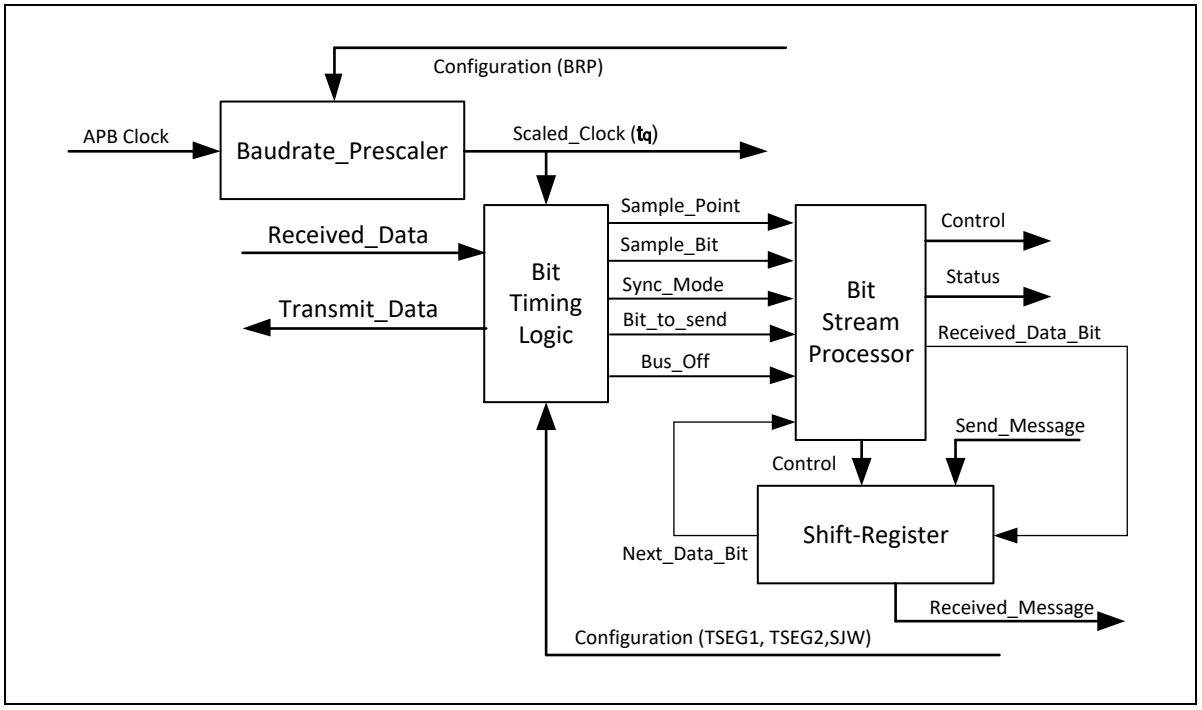


Figure 6-153 Structure of the CAN Core's CAN Protocol Controller

The data in the bit timing registers is the configuration input of the CAN protocol controller. The Baud Rate Prescaler (configured by BRP) defines the length of the time quantum, the basic time unit of the bit time; the Bit Timing Logic (configured by TSEG1, TSEG2 and SJW) defines the number of time quanta in the bit time.

The processing of the bit time, the calculation of the position of the Sample Point, and occasional synchronizations are controlled by the BTL (Bit Timing Logic) state machine, which is evaluated once each time quantum. The rest of the CAN protocol controller, the BSP (Bit Stream Processor) state machine is evaluated once each bit time, at the Sample Point.

The Shift Register sends the messages serially and parallelizes received messages. It's loading and shifting is controlled by the BSP.

The BSP translates messages into frames and vice versa. It generates and discards the enclosing fixed format bits, inserts and extracts stuff bits, calculates and checks the CRC code, performs the error management, and decides which type of synchronization is to be used. It is evaluated at the Sample Point and processes the sampled bus input bit. The time that is needed to calculate the next

bit to be sent after the Sample point (e.g. data bit, CRC (Cyclic Redundancy Check) bit, stuff bit, error flag or idle) is called the Information Processing Time (IPT).

The IPT is application specific but may not be longer than 2 t_q; the IPT for the C_CAN is 0 t_q. Its length is the lower limit of the programmed length of Phase_Seg2. In case of a synchronization, Phase_Seg2 may be shortened to a value less than IPT, which does not affect bus timing.

6.16.7.21 Calculating Bit Timing Parameters

Usually, the calculation of the bit timing configuration starts with a desired bit rate or bit time. The resulting bit time (1/bit rate) must be an integer multiple of the APB clock period.

The bit time may consist of 4 to 25 time quanta, the length of the time quantum t_q is defined by the Baud Rate Prescaler with t_q = (Baud Rate Prescaler)/f_{apb_clk}. Several combinations may lead to the desired bit time, allowing iterations of the following steps.

First part of the bit time to be defined is the Prop_Seg. Its length depends on the delay time measured in the APB clock. A maximum bus length as well as a maximum node delay has to be defined for expandible CAN bus systems. The resulting time for Prop_Seg is converted into time quanta (rounded up to the nearest integer multiple of t_q).

The Sync_Seg is 1 t_q long (fixed), leaving (bit time – Prop_Seg – 1) t_q for the two Phase Buffer Segments. If the number of remaining t_q is even, the Phase Buffer Segments have the same length, Phase_Seg2 = Phase_Seg1, else Phase_Seg2 = Phase_Seg1 + 1.

The minimum nominal length of Phase_Seg2 has to be regarded as well. Phase_Seg2 may not be shorter than the IPT of the CAN controller, which, depending on the actual implementation, is in the range of [0..2] t_q.

The length of the Synchronization Jump Width is set to its maximum value, which is the minimum of 4 and Phase_Seg1.

The oscillator tolerance range necessary for the resulting configuration is calculated by the formulas given in Section “Oscillator Tolerance Range”.

If more than one configuration is possible, that configuration allowing the highest oscillator tolerance range should be chosen.

CAN nodes with different system clocks require different configurations to come to the same bit rate. The calculation of the propagation time in the CAN network, based on the nodes with the longest delay time, is done once for the whole network.

The oscillator tolerance range of the CAN systems is limited by that node with the lowest tolerance range.

The calculation may shows that bus length or bit rate have to be decreased or that the stability of the oscillator frequency has to be increased in order to find a protocol compliant configuration of the CAN bit timing. The resulting configuration is written into the Bit Timing Register: (Phase_Seg2-1) & (Phase_Seg1+Prop_Seg-1) & (SynchronisationJumpWidth-1) & (Prescaler-1)

Example for Bit Timing at High Baud Rate

In this example, the frequency of APB_CLK is 10 MHz, BRP (CAN_BTIME[5:0]) is 0, and the bit rate is 1 Mbit/s.

t_q	100 ns		= t_{APB_CLK}
delay of bus driver	50 ns		
delay of receiver circuit	30 ns		
delay of bus line (40m)	220 ns		
t_{Prop}	600 ns		= 6 • t_q
t_{SJW}	100 ns		= 1 • t_q
t_{Tseg1}	700 ns		= t_{Prop} + t_{SJW}
t_{Tseg2}	200 ns		= Information Processing Time + 1 • t_q
$t_{Sync-Seg}$	100 ns		= 1 • t_q
bit time	1000 ns		= $t_{Sync-Seg}$ + t_{Tseg1} + t_{Tseg2}
toleranceforAPB_CLK	0.39	%	$= \frac{Min(PB1, PB2)}{2 \times 13 \times (bit\ time - PB2))}$

$$= \frac{0.1\mu s}{2x(13x(1\mu s - 0.2\mu s))}$$

In this example, the concatenated bit time parameters are $(2\text{-}1)_3$ & $(7\text{-}1)_4$ & $(1\text{-}1)_2$ & $(1\text{-}1)_6$, and the Bit Timing Register is programmed to 0x1600.

Example for Bit Timing at Low Baud Rate

In this example, the frequency of APB_CLK is 2 MHz, BRP (CAN_BTIME[5:0]) is 1, and the bit rate is 100 Kbit/s.

t_q 1 us = 2 • t_{APB_CLK}

delay of bus driver 200 ns

delay of receiver circuit 80 ns

delay of bus line (40m) 220 ns

t_{Prop} 1 us = 1 • t_q

t_{SJW} 4 us = 4 • t_q

t_{Tseg1} 5 us = t_{Prop} + t_{SJW}

t_{Tseg2} 4 us = Information Processing Time + 3 • t_q

$t_{Sync\text{-}Seg}$ 1 us = 1 • t_q

bit time 10 us = $t_{Sync\text{-}Seg}$ + t_{Tseg1} + t_{Tseg2}

toleranceforAPB_CLK 1.58 %

$$= \frac{Min(PB1, PB2)}{2 \times 13 \times (bit\ time - PB2))}$$

$$= \frac{4us}{2 \times (13 \times (10us - 4us))}$$

In this example, the concatenated bit time parameters are $(4\text{-}1)_3$ & $(5\text{-}1)_4$ & $(4\text{-}1)_2$ & $(2\text{-}1)_6$, and the Bit Timing Register is programmed to 0x34C1.

6.16.7.22 CAN Interface Reset State

After the hardware reset, the C_CAN registers hold the reset values which are given in the register description in register map chapter.

Additionally the bus-off state is reset and the output CAN_TX is set to recessive (HIGH). The value 0x0001 (Init = '1') in the CAN Control Register enables the software initialization. The C_CAN does not influence the CAN bus until the application software resets the Init bit (CAN_CON[0]) to '0'.

The data stored in the Message RAM is not affected by a hardware reset. After powered on, the contents of the Message RAM are undefined.

6.16.8 Register Map

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

Register	Offset	R/W	Description	Reset Value
CAN Base Address: **CAN_BA = 0x4018_0000**				
CAN_CON	CAN_BA+0x00	R/W	Control Register	0x0000_0001
CAN_STATUS	CAN_BA+0x04	R/W	Status Register	0x0000_0000
CAN_ERR	CAN_BA+0x08	R	Error Counter Register	0x0000_0000
CAN_BTIME	CAN_BA+0x0C	R/W	Bit Timing Register	0x0000_2301
CAN_IIDR	CAN_BA+0x10	R	Interrupt Identifier Register	0x0000_0000
CAN_TEST	CAN_BA+0x14	R/W	Test Register	0x0000_00x0 [1]
CAN_BRPE	CAN_BA+0x18	R/W	Baud Rate Prescaler Extension Register	0x0000_0000
CAN_IF1_CREQ	CAN_BA+0x20	R/W	IF1 Command Request Register	0x0000_0001
CAN_IF2_CREQ	CAN_BA+0x80	R/W	IF2 Command Request Register	0x0000_0001
CAN_IF1_CMASK	CAN_BA+0x24	R/W	IF1 Command Mask Register	0x0000_0000
CAN_IF2_CMASK	CAN_BA+0x84	R/W	IF2 Command Mask Register	0x0000_0000
CAN_IF1_MASK1	CAN_BA+0x28	R/W	IF1 Mask1 Register	0x0000_FFFF
CAN_IF2_MASK1	CAN_BA+0x88	R/W	IF2 Mask1 Register	0x0000_FFFF
CAN_IF1_MASK2	CAN_BA+0x2C	R/W	IF1 Mask2 Register	0x0000_FFFF
CAN_IF2_MASK2	CAN_BA+0x8C	R/W	IF2 Mask2 Register	0x0000_FFFF
CAN_IF1_ARB1	CAN_BA+0x30	R/W	IF1Arbitration1 Register	0x0000_0000
CAN_IF2_ARB1	CAN_BA+0x90	R/W	IF2 Arbitration1 Register	0x0000_0000
CAN_IF1_ARB2	CAN_BA+0x34	R/W	IF1 Arbitration2 Register	0x0000_0000
CAN_IF2_ARB2	CAN_BA+0x94	R/W	IF2 Arbitration2 Register	0x0000_0000
CAN_IF1_MCON	CAN_BA+0x38	R/W	IF1 Message Control Register	0x0000_0000
CAN_IF2_MCON	CAN_BA+0x98	R/W	IF2 Message Control Register	0x0000_0000
CAN_IF1_DAT_A1	CAN_BA+0x3C	R/W	IF1 DataA1 Register	0x0000_0000
CAN_IF1_DAT_A2	CAN_BA+0x40	R/W	IF1 DataA2 Register	0x0000_0000
CAN_IF1_DAT_B1	CAN_BA+0x44	R/W	IF1 DataB1 Register	0x0000_0000
CAN_IF1_DAT_B2	CAN_BA+0x48	R/W	IF1 DataB2 Register	0x0000_0000
CAN_IF2_DAT_A1	CAN_BA+0x9C	R/W	IF2 DataA1 Register	0x0000_0000

CAN_IF2_DAT_A2	CAN_BA+0xA0	R/W	IF2 DataA2 Register	0x0000_0000
CAN_IF2_DAT_B1	CAN_BA+0xA4	R/W	IF2 DataB1 Register	0x0000_0000
CAN_IF2_DAT_B2	CAN_BA+0xA8	R/W	IF2 DataB2 Register	0x0000_0000
CAN_TXREQ1	CAN_BA+0x100	R	Transmission Request Register1	0x0000_0000
CAN_TXREQ2	CAN_BA+0x104	R	Transmission Request Register2	0x0000_0000
CAN_NDAT1	CAN_BA+0x120	R	New Data Register 1	0x0000_0000
CAN_NDAT2	CAN_BA+0x124	R	New Data Register 2	0x0000_0000
CAN_IPND1	CAN_BA+0x140	R	Interrupt Pending Register1	0x0000_0000
CAN_IPND2	CAN_BA+0x144	R	Interrupt Pending Register2	0x0000_0000
CAN_MVLD1	CAN_BA+0x160	R	Message Valid Register1	0x0000_0000
CAN_MVLD2	CAN_BA+0x164	R	Message Valid Register2	0x0000_0000
CAN_WU_EN	CAN_BA+0x168	R/W	Wake-Up Enable Register	0x0000_0000
CAN_WU_STATUS	CAN_BA+0x16C	R/W	Wake-Up Status Register	0x0000_0000

Note: 1. 0x00 & 0br0000000, where r signifies the actual value of the CAN_RX

5. Ifn: The two sets of Message Interface Registers – IF1 and IF2, have identical function

CAN Register Map for Each Bit Function

Addr Offset	Register Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
00h	CAN_CON	Reserved								Test	CCE	DAR	Res	EIE	SIE	IE	Init
04h	CAN_STATUS	Reserved								Boff	Ewarn	Epass	RxOk	TxOk	LEC		
08h	CAN_ERR	RP	REC6-0							TEC7-0							
0Ch	CAN_BTIME	Res	Tseg2			Tseg1				SJW		BRP					
10h	CAN_IIDR	IntId15-8								IntId7-0							
14h	CAN_TEST	Reserved								Rx	Tx1	Tx0	Lback	Silent	Basic	Reserved	
18h	CAN_BRPE	Reserved												BRPE			
20h	CAN_IF1_CREQ	Busy	Reserved									Message Number					
24h	CAN_IF1_CMASK	Reserved								WR/RD	Mask	Arb	Control	ClrIntPnd	TxRqst/	Data A	Data B
28h	CAN_IF1_MASK1	Msk15-0															
2Ch	CAN_IF1_MASK2	MXtd	Mdir	Res	Msk28-16												
30h	CAN_IF1_ARB1	ID15-0															
34h	CAN_IF1_ARB2	MsgVal	Xtd	Dir	ID28-16												

Addr Offset	Register Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
38h	CAN_IF1_MCON	NewDat	MsgLst	IntPnd	Umask	TxIE	RxIE	RmtEn	TxRqst	EoB	Reserved			DLC3-0			
3Ch	CAN_IF1_DAT_A1	Data(1)								Data(0)							
40h	CAN_IF1_DAT_A2	Data(3)								Data(2)							
44h	CAN_IF1_DAT_B1	Data(5)								Data(4)							
48h	CAN_IF1_DAT_B2	Data(7)								Data(6)							
80h	CAN_IF2_CREQ	Busy	Reserved									Message Number					
84h	CAN_IF2_CMASK	Reserved								WR/RD	Mask	Arb	Control	ClrIntPnd	TxRqst/	Data A	Data B
88h	CAN_IF2_MASK1	Msk15-0															
8Ch	CAN_IF2_MASK2	MXtd	Mdir	Res.	Msk28-16												
90h	CAN_IF2_ARB1	ID15-0															
94h	CAN_IF2_ARB2	MsgVal	Xtd	Dir	ID28-16												
98h	CAN_IF2_MCON	NewDat	MsgLst	IntPnd	Umask	TxIE	RxIE	RmtEn	TxRqst	EoB	Reserved			DLC3-0			
9Ch	CAN_IF2_DAT_A1	Data(1)								Data(0)							

Addr Offset	Register Name	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
A0h	CAN_IF2_DAT_A2	Data(3)								Data(2)							
A4h	CAN_IF2_DAT_B1	Data(5)								Data(4)							
A8h	CAN_IF2_DAT_B2	Data(7)								Data(6)							
100h	CAN_TXREQ1	TxRqst16-1															
104h	CAN_TXREQ2	TxRqst32-17															
120h	CAN_NDAT1	NewDat16-1															
124h	CAN_NDAT2	NewDat32-17															
140h	CAN_IPND1	IntPnd16-1															
144h	CAN_IPND2	IntPnd32-17															
160h	CAN_MVLD1	MsgVal16-1															
164h	CAN_MVLD2	MsgVal32-17															
168h	CAN_WU_EN	Reserved															WAKUP_EN
16Ch	CAN_WU_STATUS	Reserved															WAKUP_STS
170h	CAN_RAM_CEN	Reserved															RAM_CEN
Others	Reserved	Reserved															

Table 6-27 CAN Register Map for Each Bit Function

Note: Reserved bits are read as 0’ except for Ifn Mask 2 Register where they are read as ’1’.

Res. = Reserved

6.16.9 Register Description

The C_CAN allocates an address space of 256 bytes. The registers are organized as 16-bit registers.

The two sets of interface registers (IF1 and IF2) control the software access to the Message RAM. They buffer the data to be transferred to and from the RAM, avoiding conflicts between software accesses and message reception/transmission.

CAN Control Register (CAN_CON)

Register	Offset	R/W	Description	Reset Value
CAN_CON	CAN_BA+0x00	R/W	Control Register	0x0000_0001

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**Test**	**Test Mode Enable Bit** 0 = Normal Operation. 1 = Test Mode.
[6]	**CCE**	**Configuration Change Enable Bit** 0 = No write access to the Bit Timing Register. 1 = Write access to the Bit Timing Register (CAN_BTIME) allowed. (while Init bit (CAN_CON[0]) = 1).
[5]	**DAR**	**Automatic Re-transmission Disable Bit** 0 = Automatic Retransmission of disturbed messages Enabled. 1 = Automatic Retransmission Disabled.
[4]	**Reserved**	Reserved.
[3]	**EIE**	**Error Interrupt Enable Bit** 0 = Disabled – No Error Status Interrupt will be generated. 1 = Enabled – A change in the bits Boff (CAN_STATUS[7]) or Ewarn (CAN_STATUS[6]) in the Status Register will generate an interrupt.
[2]	**SIE**	**Status Change Interrupt Enable Bit** 0 = Disabled – No Status Change Interrupt will be generated. 1 = Enabled – An interrupt will be generated when a message transfer is successfully completed or a CAN bus error is detected.
[1]	**IE**	**Module Interrupt Enable Bit** 0 = Interrupt Disabled. 1 = Interrupt Enabled.
[0]	**Init**	**Init Initialization** 0 = Normal Operation. 1 = Initialization is started.

Note: The bus-off recovery sequence (see CAN Specification Rev. 2.0) cannot be shortened by setting or resetting the Init bit (CAN_CON[0]). If the device goes in the bus-off state, it will set Init of its own accord, stopping all bus activities. Once Init has been cleared by the CPU, the device will then wait for 129 occurrences of Bus Idle (129 * 11 consecutive recessive bits) before resuming normal operations. At the end of the bus-off recovery sequence, the Error Management Counters will be reset.

During the waiting time after resetting Init, each time a sequence of 11 recessive bits has been monitored, a Bit0 Error code is written to the Status Register, enabling the CPU to readily check up whether the CAN bus is stuck at dominant or continuously disturbed and to monitor the proceeding of the bus-off recovery sequence.

CAN Status Register (CAN_STATUS)

Register	Offset	R/W	Description	Reset Value
CAN_STATUS	CAN_BA+0x04	R/W	Status Register	0x0000_0000

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**BOFF**	**Bus-off Status (Read Only)** 0 = The CAN module is not in bus-off state. 1 = The CAN module is in bus-off state.
[6]	**Ewarn**	**Error Warning Status (Read Only)** 0 = Both error counters are below the error warning limit of 96. 1 = At least one of the error counters in the EML has reached the error warning limit of 96.
[5]	**Epass**	**Error Passive (Read Only)** 0 = The CAN Core is error active. 1 = The CAN Core is in the error passive state as defined in the CAN Specification.
[4]	**RxOK**	**Received a Message Successfully** 0 = No message has been successfully received since this bit was last reset by the CPU. This bit is never reset by the CAN Core. 1 = A message has been successfully received since this bit was last reset by the CPU (independent of the result of acceptance filtering).
[3]	**TxOK**	**Transmitted a Message Successfully** 0 = Since this bit was reset by the CPU, no message has been successfully transmitted. This bit is never reset by the CAN Core. 1 = Since this bit was last reset by the CPU, a message has been successfully (error free and acknowledged by at least one other node) transmitted.
[2:0]	**LEC**	**Last Error Code (Type of the Last Error to Occur on the CAN Bus)** The LEC field holds a code, which indicates the type of the last error to occur on the CAN bus. This field will be cleared to '0' when a message has been transferred (reception or transmission) without error. The unused code '7' may be written by the CPU to check for updates. The Table 6-28describes the error code.

Error Code	Meanings
0	No Error
1	Stuff Error: More than 5 equal bits in a sequence have occurred in a part of a received message where this is not allowed.
2	Form Error: A fixed format part of a received frame has the wrong format.
3	AckError: The message this CAN Core transmitted was not acknowledged by another node.
4	Bit1Error: During the transmission of a message (with the exception of the arbitration field), the device wanted to send a recessive level (bit of logical value '1'), but the monitored bus value was dominant.
5	Bit0Error: During the transmission of a message (or acknowledge bit, or active error flag, or overload flag), though the device wanted to send a dominant level (data or identifier bit logical value '0'), but the monitored Bus value was recessive. During bus-off recovery, this status is set each time a sequence of 11 recessive bits has been monitored. This enables the CPU to monitor the proceedings of the bus-off recovery sequence (indicating the bus is not stuck at dominant or continuously disturbed).
6	CRCError: The CRC check sum was incorrect in the message received, the CRC received for an incoming message does not match with the calculated CRC for the received data.
7	Unused: When the LEC shows the value '7', no CAN bus event was detected since the CPU wrote this value to the LEC.

Table 6-28Last Error Code

Status Interrupts

A Status Interrupt is generated by bits Boff (CAN_STATUS[7]) and Ewarn (CAN_STATUS[6]) (Error Interrupt) or by RxOk (CAN_STATUS[4]), TxOk (CAN_STATUS[3]) and LEC (CAN_STATUS[2:0]) (Status Change Interrupt) assumed that the corresponding enable bits in the CAN Control Register are set. A change of bit Epass (CAN_STATUS[5]) or a write to RxOk, TxOk or LEC will never generate a Status Interrupt.

Reading the Status Register will clear the Status Interrupt value (8000h) in the Interrupt Register, if it is pending.

CAN Error Counter Register (CAN_ERR)

Register	Offset	R/W	Description	Reset Value
CAN_ERR	CAN_BA+0x08	R	Error Counter Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**RP**	**Receive Error Passive** 0 = The Receive Error Counter is below the error passive level. 1 = The Receive Error Counter has reached the error passive level as defined in the CAN Specification.
[14:8]	**REC**	**Receive Error Counter** Actual state of the Receive Error Counter. Values between 0 and 127.
[7:0]	**TEC**	**Transmit Error Counter** Actual state of the Transmit Error Counter. Values between 0 and 255.

Bit Timing Register (CAN_BTIME)

Register	Offset	R/W	Description	Reset Value
CAN_BTIME	CAN_BA+0x0C	R/W	Bit Timing Register	0x0000_2301

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved	Tseg2			Tseg1			
7	6	5	4	3	2	1	0
SJW		BRP					

Bits	Description	
[31:15]	**Reserved**	Reserved.
[14:12]	**Tseg2**	**Time Segment After Sample Point** 0x0-0x7: Valid values for Tseg2 are [0…7]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
[11:8]	**Tseg1**	**Time Segment Before the Sample Point Minus Sync_Seg** 0x01-0x0F: valid values for Tseg1 are [1…15]. The actual interpretation by the hardware of this value is such that one more than the value programmed is used.
[7:6]	**SJW**	**(Re)Synchronization Jump Width** 0x0-0x3: Valid programmed values are [0…3]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.
[5:0]	**BRP**	**Baud Rate Prescaler** 0x01-0x3F: The value by which the oscillator frequency is divided for generating the bit time quanta. The bit time is built up from a multiple of this quanta. Valid values for the Baud Rate Prescaler are [0…63]. The actual interpretation by the hardware of this value is such that one more than the value programmed here is used.

Note: With a module clock APB_CLK of 8 MHz, the reset value of 0x2301 configures the C_CAN for a bit rate of 500 Kbit/s. The registers are only writable if bits CCE (CAN_CON[6]) and Init (CAN_CON[0]) are set.

Interrupt Identify Register (CAN_IIDR)

Register	Offset	R/W	Description	Reset Value
CAN_IIDR	CAN_BA+0x10	R	Interrupt Identifier Register	0x0000_0000

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

Bits	Description	
[15:0]	IntId	**Interrupt Identifier (Indicates the Source of the Interrupt)** If several interrupts are pending, the CAN Interrupt Register will point to the pending interrupt with the highest priority, disregarding their chronological order. An interrupt remains pending until the application software has cleared it. If IntId is different from 0x0000 and IE (CAN_Ifn_MCON[1]) is set, the IRQ interrupt signal to the EIC is active. The interrupt remains active until IntId is back to value 0x0000 (the cause of the interrupt is reset) or until IE is reset. The Status Interrupt has the highest priority. Among the message interrupts, the Message Object' s interrupt priority decreases with increasing message number. A message interrupt is cleared by clearing the Message Object's IntPnd bit (CAN_Ifn_MCON[13]). The Status Interrupt is cleared by reading the Status Register.

IntId Value	Meanings
0x0000	No Interrupt is Pending
0x0001-0x0020	Number of Message Object which caused the interrupt.
0x0021-0x7FFF	Unused
0x8000	Status Interrupt
0x8001-0xFFFF	Unused

Table 6-29Source of Interrupts

Test Register (CAN_TEST)

Register	Offset	R/W	Description	Reset Value
CAN_TEST	CAN_BA+0x14	R/W	Test Register (Register Map Note 1)	0x0000_0080

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

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**Rx**	**Monitors the Actual Value of CAN_RX Pin (Read Only)*(1)** 0 = The CAN bus is dominant (CAN_RX = '0'). 1 = The CAN bus is recessive (CAN_RX = '1').
[6:5]	**Tx**	**Tx[1:0]: Control of CAN_TX Pin** 00 = Reset value, CAN_TX pin is controlled by the CAN Core. 01 = Sample Point can be monitored at CAN_TX pin. 10 = CAN_TX pin drives a dominant ('0') value. 11 = CAN_TX pin drives a recessive ('1') value.
[4]	**Lback**	**Loop Back Mode Enable Bit** 0 = Loop Back Mode is Disabled. 1 = Loop Back Mode is Enabled.
[3]	**Silent**	**Silent Mode** 0 = Normal operation. 1 = The module is in Silent Mode.
[2]	**Basic**	**Basic Mode** 0 = Basic Mode Disabled. 1= IF1 Registers used as Tx Buffer, IF2 Registers used as Rx Buffer.
[1:0]	**Reserved**	Reserved.

Reset value: 0000 0000 R000 0000 b (R:current value of RX pin)

Note: Write access to the Test Register is enabled by setting the Test bit (CAN_CON[7]). The different test functions may be combined, but Tx[1-0] "00" (CAN_TEST[6:5]) disturbs message transfer..

Baud Rate Prescaler Extension REGISTER (CAN_BRPE)

Register	Offset	R/W	Description	Reset Value
CAN_BRPE	CAN_BA+0x18	R/W	Baud Rate Prescaler Extension Register	0x0000_0000

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

Bits	Description	
[31:4]	Reserved	Reserved.
[3:0]	BRPE	**BRPE: Baud Rate Prescaler Extension** 0x00-0x0F: By programming BRPE, the Baud Rate Prescaler can be extended to values up to 1023. The actual interpretation by the hardware is that one more than the value programmed by BRPE (MSBs) and BTIME (LSBs) is used.

Message Interface Register Sets

There are two sets of Interface Registers, which are used to control the CPU access to the Message RAM. The Interface Registers avoid conflict between the CPU accesses to the Message RAM and CAN message reception and transmission by buffering the data to be transferred. A complete Message Object or parts of the Message Object may be transferred between the Message RAM and the Ifn Message Buffer registers in one single transfer.

The function of the two interface register sets is identical except for the Basic test mode. They can be used the way one set of registers is used for data transfer to the Message RAM while the other set of registers is used for the data transfer from the Message RAM, allowing both processes to be interrupted by each other. The Table 6-30provides an overview of the two Interface Register sets.

Each set of Interface Registers consists of Message Buffer Registers controlled by their own Command Registers. The Command Mask Register specifies the direction of the data transfer and which parts of a Message Object will be transferred. The Command Request Register is used to select a Message Object in the Message RAM as target or source for the transfer and to start the action specified in the Command Mask Register.

Address	IF1 Register Set	Address	IF2 Register Set
CAN_BA+0x20	IF1 Command Request	CAN_BA+0x80	IF2 Command Request
CAN_BA+0x24	IF1 Command Mask	CAN_BA+0x84	IF2 Command Mask
CAN_BA+0x28	IF1 Mask 1	CAN_BA+0x88	IF2 Mask 1
CAN_BA+0x2C	IF1 Mask 2	CAN_BA+0x8C	IF2 Mask 2
CAN_BA+0x30	IF1 Arbitration 1	CAN_BA+0x90	IF2 Arbitration 1
CAN_BA+0x34	IF1 Arbitration 2	CAN_BA+0x94	IF2 Arbitration 2
CAN_BA+0x38	IF1 Message Control	CAN_BA+0x98	IF2 Message Control
CAN_BA+0x3C	IF1 Data A 1	CAN_BA+0x9C	IF2 Data A 1
CAN_BA+0x40	IF1 Data A 2	CAN_BA+0xA0	IF2 Data A 2
CAN_BA+0x44	IF1 Data B 1	CAN_BA+0xA4	IF2 Data B 1
CAN_BA+0x48	IF1 Data B 2	CAN_BA+0xA8	IF2 Data B 2

Table 6-30IF1 and IF2 Message Interface Register

Ifn Command Request Register (CAN_Ifn_CREQ)

Register	Offset	R/W	Description	Reset Value
CAN_IFN_CREQ	CAN_BA+0x20 + (0x60 *(n-1))	R/W	Ifn (Register Map Note 2) Command Request Registers	0x0000_0001

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**Busy**	**Busy Flag** 0 = Read/write action has finished. 1 = Writing to the Ifn Command Request Register is in progress. This bit can only be read by the software.
[14:6]	**Reserved**	Reserved.
[5:0]	**Message Number**	**Message Number** 0x01-0x20: Valid Message Number, the Message Object in the Message RAM is selected for data transfer. 0x00: Not a valid Message Number, interpreted as 0x20. 0x21-0x3F: Not a valid Message Number, interpreted as 0x01-0x1F.

A message transfer is started as soon as the application software has written the message number to the Command Request Register. With this write operation, the Busy bit (CAN_Ifn_CREQ[15]) is automatically set to notify the CPU that a transfer is in progress. After a waiting time of 3 to 6 APB_CLK periods, the transfer between the Interface Register and the Message RAM is completed. The Busy bit is cleared.

Note: When a Message Number that is not valid is written into the Command Request Register, the Message Number will be transformed into a valid value and that Message Object will be transferred.

Ifn Command Mask Register (CAN_Ifn_CMASK)

The control bits of the Ifn Command Mask Register specify the transfer direction and select which of the Ifn Message Buffer Registers are source or target of the data transfer.

Register	Offset	R/W	Description	Reset Value
CAN_IFN_CMASK	CAN_BA+0x24 + (0x60 *(n-1))	R/W	Ifn Command Mask Registers	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
WR/RD	Mask	Arb	Control	ClrIntPnd	TxRqst/NewDat	DAT_A	DAT_B

Bits	Description	
[31:8]	Reserved	Reserved.
[7]	WR/RD	**Write / Read Mode** 0 = Read: Transfer data from the Message Object addressed by the Command Request Register into the selected Message Buffer Registers. 1 = Write: Transfer data from the selected Message Buffer Registers to the Message Object addressed by the Command Request Register.
[6]	Mask	**Access Mask Bits** Write Operation: 0 = Mask bits unchanged. 1 = Transfer Identifier Mask + Mdir + MXtd to Message Object. Read Operation: 0 = Mask bits unchanged. 1 = Transfer Identifier Mask + Mdir + MXtd to Ifn Message Buffer Register.
[5]	Arb	**Access Arbitration Bits** Write Operation: 0 = Arbitration bits unchanged. 1 = Transfer Identifier + Dir (CAN_Ifn_ARB2[13]) + Xtd (CAN_Ifn_ARB2[14]) + MsgVal (CAN_Ifn_ARB2[15]) to Message Object. Read Operation: 0 = Arbitration bits unchanged. 1 = Transfer Identifier + Dir + Xtd + MsgVal to Ifn Message Buffer Register.
[4]	Control	**Control Access Control Bits** Write Operation: 0 = Control Bits unchanged. 1 = Transfer Control Bits to Message Object.

		Read Operation: 0 = Control Bits unchanged. 1 = Transfer Control Bits to Ifn Message Buffer Register.
[3]	**ClrIntPnd**	**Clear Interrupt Pending Bit** Write Operation: When writing to a Message Object, this bit is ignored. Read Operation: 0 = IntPnd bit (CAN_Ifn_MCON[13]) remains unchanged. 1 = Clear IntPnd bit in the Message Object.
[2]	**TxRqst/NewDat**	**Access Transmission Request Bit When Write Operation** 0 = TxRqst bit unchanged. 1 = Set TxRqst bit. **Note:** If a transmission is requested by programming bit TxRqst/NewDat in the Ifn Command Mask Register, bit TxRqst in the Ifn Message Control Register will be ignored. Access New Data Bit when Read Operation. 0 = NewDat bit remains unchanged. 1 = Clear NewDat bit in the Message Object. **Note:** A read access to a Message Object can be combined with the reset of the control bits IntPnd and NewDat. The values of these bits transferred to the Ifn Message Control Register always reflect the status before resetting these bits.
[1]	**DAT_A**	**Access Data Bytes [3:0]** Write Operation: 0 = Data Bytes [3:0] unchanged. 1 = Transfer Data Bytes [3:0] to Message Object. Read Operation: 0 = Data Bytes [3:0] unchanged. 1 = Transfer Data Bytes [3:0] to Ifn Message Buffer Register.
[0]	**DAT_B**	**Access Data Bytes [7:4]** Write Operation: 0 = Data Bytes [7:4] unchanged. 1 = Transfer Data Bytes [7:4] to Message Object. Read Operation: 0 = Data Bytes [7:4] unchanged. 1 = Transfer Data Bytes [7:4] to Ifn Message Buffer Register.

Ifn Mask 1 Register (CAN_Ifn_MASK1)

Register	Offset	R/W	Description	Reset Value
CAN_IFN_MASK1	CAN_BA+0x28 + (0x60 *(n-1))	R/W	Ifn Mask 1 Registers	0x0000_FFFF

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**Msk[15:0]**	**Identifier Mask 15-0** 0 = The corresponding bit in the identifier of the message object cannot inhibit the match in the acceptance filtering. 1 = The corresponding identifier bit is used for acceptance filtering.

Ifn Mask 2 Register (CAN_Ifn_MASK2)

Register	Offset	R/W	Description	Reset Value
CAN_IFN_MASK2	CAN_BA+0x2C + (0x60 *(n-1))	R/W	Ifn Mask 2 Registers	0x0000_FFFF

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**MXtd**	**Mask Extended Identifier** 0 = The extended identifier bit (IDE) has no effect on the acceptance filtering. 1 = The extended identifier bit (IDE) is used for acceptance filtering. **Note:** When 11-bit (“standard”) Identifiers are used for a Message Object, the identifiers of received Data Frames are written into bits ID28 to ID18 (CAN_Ifn_ARB2[12:2]). For acceptance filtering, only these bits together with mask bits Msk28 to Msk18 (CAN_Ifn_MASK2[12:2]) are considered.
[14]	**Mdir**	**Mask Message Direction** 0 = The message direction bit (Dir (CAN_Ifn_ARB2[13])) has no effect on the acceptance filtering. 1 = The message direction bit (Dir) is used for acceptance filtering.
[13]	**Reserved**	Reserved.
[12:0]	**Msk[28:16]**	**Identifier Mask 28-16** 0 = The corresponding bit in the identifier of the message object cannot inhibit the match in the acceptance filtering. 1 = The corresponding identifier bit is used for acceptance filtering.

Ifn Arbitration 1 Register (CAN_Ifn_ARB1)

Register	Offset	R/W	Description	Reset Value
CAN_IFN_ARB1	CAN_BA+0x30 + (0x60 *(n-1))	R/W	Ifn Arbitration 1 Registers	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**ID[15:0]**	**Message Identifier 15-0** ID28 – ID0, 29-bit Identifier (“Extended Frame”). ID28 – ID18, 11-bit Identifier (“Standard Frame”)

Ifn Arbitration 2 Register (CAN_Ifn_ARB2)

Register	Offset	R/W	Description	Reset Value
CAN_IFN_ARB2	CAN_BA+0x34 + (0x60 *(n-1))	R/W	Ifn Arbitration 2 Registers	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**MsgVal**	**Message Valid** 0 = The Message Object is ignored by the Message Handler. 1 = The Message Object is configured and should be considered by the Message Handler. **Note:** The application software must reset the MsgVal bit of all unused Messages Objects during the initialization before it resets bit Init (CAN_CON[0]). This bit must also be reset before the identifier Id28-0 (CAN_Ifn_ARB1/2), the control bits Xtd (CAN_Ifn_ARB2[14]), Dir (CAN_Ifn_ARB2[13]), or the Data Length Code DLC3-0 (CAN_Ifn_MCON[3:0]) are modified, or if the Messages Object is no longer required.
[14]	**Xtd**	**Extended Identifier** 0 = The 11-bit (“standard”) Identifier will be used for this Message Object. 1 = The 29-bit (“extended”) Identifier will be used for this Message Object.
[13]	**Dir**	**Message Direction** 0 = Direction is receive. On TxRqst, a Remote Frame with the identifier of this Message Object is transmitted. On reception of a Data Frame with matching identifier, that message is stored in this Message Object. 1 = Direction is transmit. On TxRqst, the respective Message Object is transmitted as a Data Frame. On reception of a Remote Frame with matching identifier, the TxRqst bit (CAN_Ifn_CMASK[2]) of this Message Object is set (if RmtEn (CAN_Ifn_MCON[9]) = one).
[12:0]	**ID[28:16]**	**Message Identifier 28-16** ID28 – ID0, 29-bit Identifier (“Extended Frame”). ID28 – ID18, 11-bit Identifier (“Standard Frame”)

Ifn Message Control Register (CAN_Ifn_MCON)

Register	Offset	R/W	Description	Reset Value
CAN_IFN_MCON	CAN_BA+0x38 + (0x60 *(n-1))	R/W	Ifn Message Control Registers	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
NewDat	MsgLst	IntPnd	Umask	TxIE	RxIE	RmtEn	TxRqst
7	6	5	4	3	2	1	0
EoB	Reserved			DLC			

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**NewDat**	**New Data** 0 = No new data has been written into the data portion of this Message Object by the Message Handler since last time this flag was cleared by the application software. 1 = The Message Handler or the application software has written new data into the data portion of this Message Object.
[14]	**MsgLst**	Message Lost (only valid for Message Objects with direction = receive). 0 = No message lost since last time this bit was reset by the CPU. 1 = The Message Handler stored a new message into this object when NewDat was still set, the CPU has lost a message.
[13]	**IntPnd**	**Interrupt Pending** 0 = This message object is not the source of an interrupt. 1 = This message object is the source of an interrupt. The Interrupt Identifier in the Interrupt Register will point to this message object if there is no other interrupt source with higher priority.
[12]	**Umask**	**Use Acceptance Mask** 0 = Mask ignored. 1 = Use Mask (Msk28-0, MXtd, and Mdir) for acceptance filtering. **Note:** If the Umask bit is set to one, the Message Object's mask bits have to be programmed during initialization of the Message Object before MsgVal bit (CAN_Ifn_ARB2[15]) is set to one.
[11]	**TxIE**	**Transmit Interrupt Enable Bit** 0 = IntPnd (CAN_Ifn_MCON[13]) will be left unchanged after the successful transmission of a frame. 1 = IntPnd will be set after a successful transmission of a frame.
[10]	**RxIE**	**Receive Interrupt Enable Bit** 0 = IntPnd (CAN_Ifn_MCON[13]) will be left unchanged after a successful reception of a frame. 1 = IntPnd will be set after a successful reception of a frame.
[9]	**RmtEn**	**Remote Enable Bit**

		0 = At the reception of a Remote Frame, TxRqst (CAN_Ifn_MCON[8]) is left unchanged. 1 = At the reception of a Remote Frame, TxRqst is set.
[8]	**TxRqst**	**Transmit Request** 0 = This Message Object is not waiting for transmission. 1 = The transmission of this Message Object is requested and is not yet done.
[7]	**EoB**	**End of Buffer** 0 = Message Object belongs to a FIFO Buffer and is not the last Message Object of that FIFO Buffer. 1 = Single Message Object or last Message Object of a FIFO Buffer. **Note:** This bit is used to concatenate two or more Message Objects (up to 32) to build a FIFO Buffer. For single Message Objects (not belonging to a FIFO Buffer), this bit must always be set to one.
[6:4]	**Reserved**	Reserved.
[3:0]	**DLC**	**Data Length Code** 0-8: Data Frame has 0-8 data bytes. 9-15: Data Frame has 8 data bytes **Note:** The Data Length Code of a Message Object must be defined the same as in all the corresponding objects with the same identifier at other nodes. When the Message Handler stores a data frame, it will write the DLC to the value given by the received message. Data(0): 1st data byte of a CAN Data Frame Data(1): 2nd data byte of a CAN Data Frame Data(2): 3rd data byte of a CAN Data Frame Data(3): 4th data byte of a CAN Data Frame Data(4): 5th data byte of a CAN Data Frame Data(5): 6th data byte of a CAN Data Frame Data(6): 7th data byte of a CAN Data Frame Data(7): 8th data byte of a CAN Data Frame **Note:** The Data(0)byte is the first data byte shifted into the shift register of the CAN Core during a reception while the Data(7) byte is the last. When the Message Handler stores a Data Frame, it will write all the eight data bytes into a Message Object. If the Data Length Code is less than 8, the remaining bytes of the Message Object will be overwritten by unspecified values.

Ifn Data A1 Register (CAN_Ifn_DAT_A1)

Register	Offset	R/W	Description	Reset Value
CAN_IFN_DAT_A1	CAN_BA+0x3C + (0x60 *(n-1))	R/W	Ifn Data A1 Registers (Register Map Note 3)	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**Data(1)**	**Data Byte 1** 2nd data byte of a CAN Data Frame
[7:0]	**Data(0)**	**Data Byte 0** 1st data byte of a CAN Data Frame

Ifn Data A2 Register (CAN_Ifn_DAT_A2)

Register	Offset	R/W	Description	Reset Value
CAN_IFN_DAT_A2	CAN_BA+0x40 + (0x60 *(n-1))	R/W	Ifn Data A2 Registers (Register Map Note 3)	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**Data(3)**	**Data Byte 3** 4th data byte of CAN Data Frame
[7:0]	**Data(2)**	**Data Byte 2** 3rd data byte of CAN Data Frame

Ifn Data B1 Register (CAN_Ifn_DAT_B1)

Register	Offset	R/W	Description	Reset Value
CAN_IFN_DAT_B1	CAN_BA+0x44 + (0x60 *(n-1))	R/W	Ifn Data B1 Registers (Register Map Note 3)	0x0000_0000

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

Bits	Description	
[31:16]	Reserved	Reserved.
[15:8]	Data(5)	**Data Byte 5** 6th data byte of CAN Data Frame
[7:0]	Data(4)	**Data Byte 4** 5th data byte of CAN Data Frame

Ifn Data B2 Register (CAN_Ifn_DAT_B2)

Register	Offset	R/W	Description	Reset Value
CAN_IFN_DAT_B2	CAN_BA+0x48 + (0x60 *(n-1))	R/W	Ifn Data B2 Registers (Register Map Note 3)	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**Data(7)**	**Data Byte 7** 8th data byte of CAN Data Frame.
[7:0]	**Data(6)**	**Data Byte 6** 7th data byte of CAN Data Frame.

In a CAN Data Frame, Data [0] is the first, Data [7] is the last byte to be transmitted or received. In CAN's serial bit stream, the MSB of each byte will be transmitted first.

Message Object in the Message Memory

There are 32 Message Objects in the Message RAM. To avoid conflicts between application software access to the Message RAM and CAN message reception and transmission, the CPU cannot directly access the Message Objects, these accesses are handled through the Ifn Interface Registers. The Table 6-31provides an overview of the structures of a Message Object.

Message Object												
Umask	Msk [28:0]	MXtd	Mdir	EoB	NewDat		MsgLst	RxIE	TxIE	IntPnd	RmtEn	TxRqst
MsgVal	ID [28:0]	Xtd	Dir	DLC [3:0]	Data(0)	Data(1)	Data(2)	Data(3)	Data(4)	Data(5)	Data(6)	Data(7)

Table 6-31Structure of a Message Object in the Message Memory

The Arbitration Registers ID28-0 (CAN_Ifn_ARB1/2), Xtd (CAN_Ifn_ARB2[14]) and Dir (CAN_Ifn_ARB2[13]) are used to define the identifier and type of outgoing messages and are used (together with the mask registers Msk28-0 (CAN_Ifn_MASK1/2), MXtd (CAN_Ifn_MASK2[15]) and Mdir (CAN_Ifn_MASK2[14])) for acceptance filtering of incoming messages. A received message is stored in the valid Message Object with matching identifier and Direction = receive (Data Frame) or Direction = transmit (Remote Frame). Extended frames can be stored only in Message Objects with Xtd = one, standard frames in Message Objects with Xtd = zero. If a received message (Data Frame or Remote Frame) matches with more than one valid Message Object, it is stored into that with the lowest message number.

Message Handler Registers

All Message Handler registers are read only. Their contents (TxRqst(CAN_Ifn_MCON[8]), NewDat (CAN_Ifn_MCON[15]), IntPnd(CAN_Ifn_MCON[13]) and MsgVal (CAN_Ifn_ARB2[15]) bits of each Message Object and the Interrupt Identifier) are status information provided by the Message Handler FSM.

Transmission Request Register 1 (CAN_TXREQ1)

These registers hold the TxRqst bits of the 32 Message Objects. By reading the TxRqst bits, the software can check which Message Object in a Transmission Request is pending. The TxRqst bit of a specific Message Object can be set/reset by the application software through the Ifn Message Interface Registers or by the Message Handler after reception of a Remote Frame or after a successful transmission.

Register	Offset	R/W	Description	Reset Value
CAN_TXREQ1	CAN_BA+0x100	R	Transmission Request Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
TxRqst16-9							
7	6	5	4	3	2	1	0
TxRqst8-1							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**TxRqst16-1**	**Transmission Request Bits 16-1(of All Message Objects)** 0 = This Message Object is not waiting for transmission. 1 = The transmission of this Message Object is requested and is not yet done. These bits are read only.

Transmission Request Register 2 (CAN_TXREQ2)

Register	Offset	R/W	Description	Reset Value
CAN_TXREQ2	CAN_BA+0x104	R	Transmission Request Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
TxRqst32-25							
7	6	5	4	3	2	1	0
TxRqst24-17							

Bits	Description	
[31:16]	Reserved	Reserved.
[15:0]	TxRqst32-17	**Transmission Request Bits 32-17 (of All Message Objects)** 0 = This Message Object is not waiting for transmission. 1 = The transmission of this Message Object is requested and is not yet done. These bits are read only.

New Data Register 1 (CAN_NDAT1)

These registers hold the NewDat bits of the 32 Message Objects. By reading out the NewDat bits, the software can check for which Message Object the data portion was updated. The NewDat bit of a specific Message Object can be set/reset by the software through the Ifn Message Interface Registers or by the Message Handler after reception of a Data Frame or after a successful transmission.

Register	Offset	R/W	Description	Reset Value
CAN_NDAT1	CAN_BA+0x120	R	New Data Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
NewData16-9							
7	6	5	4	3	2	1	0
NewData8-1							

Bits	Description	
[31:16]	Reserved	Reserved.
[15:0]	NewData16-1	**New Data Bits 16-1(of All Message Objects)** 0 = No new data has been written into the data portion of this Message Object by the Message Handler since the last time this flag was cleared by the application software. 1 = The Message Handler or the application software has written new data into the data portion of this Message Object.

New Data Register 2 (CAN_NDAT2)

Register	Offset	R/W	Description	Reset Value
CAN_NDAT2	CAN_BA+0x124	R	New Data Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
NewData32-25							
7	6	5	4	3	2	1	0
NewData24-17							

Bits	Description	
[31:16]	Reserved	Reserved.
[15:0]	NewData32-17	**New Data Bits 32-17 (of All Message Objects)** 0 = No new data has been written into the data portion of this Message Object by the Message Handler since the last time this flag was cleared by the application software. 1 = The Message Handler or the application software has written new data into the data portion of this Message Object.

Interrupt Pending Register 1 (CAN_IPND1)

These registers contain the IntPnd bits of the 32 Message Objects. By reading the IntPnd bits, the software can check for which Message Object an interrupt is pending. The IntPnd bit of a specific Message Object can be set/reset by the application software through the Ifn Message Interface Registers or by the Message Handler after reception or after a successful transmission of a frame. This will also affect the value of IntId in the Interrupt Register.

Register	Offset	R/W	Description	Reset Value
CAN_IPND1	CAN_BA+0x140	R	Interrupt Pending Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
IntPnd16-9							
7	6	5	4	3	2	1	0
IntPnd8-1							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**IntPnd16-1**	**Interrupt Pending Bits 16-1(of All Message Objects)** 0 = This message object is not the source of an interrupt. 1 = This message object is the source of an interrupt.

Interrupt Pending Register 2 (CAN_IPND2)

Register	Offset	R/W	Description	Reset Value
CAN_IPND2	CAN_BA+0x144	R	Interrupt Pending Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
IntPnd32-25							
7	6	5	4	3	2	1	0
IntPnd24-17							

Bits	Description	
[31:16]	Reserved	Reserved.
[15:0]	IntPnd32-17	**Interrupt Pending Bits 32-17(of All Message Objects)** 0 = This message object is not the source of an interrupt. 1 = This message object is the source of an interrupt.

Message Valid Register 1 (CAN_MVLD1)

These registers hold the MsgVal bits of the 32 Message Objects. By reading the MsgVal bits, the application software can check which Message Object is valid. The MsgVal bit of a specific Message Object can be set/reset by the application software via the Ifn Message Interface Registers.

Register	Offset	R/W	Description	Reset Value
CAN_MVLD1	CAN_BA+0x160	R	Message Valid Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
MsgVal16- 9							
7	6	5	4	3	2	1	0
MsgVal8-1							

Bits	Description	
[31:16]	Reserved	Reserved.
[15:0]	MsgVal16-1	**Message Valid Bits 16-1(of All Message Objects)(Read Only)** 0 = This Message Object is ignored by the Message Handler. 1 = This Message Object is configured and should be considered by the Message Handler. Ex. CAN_MVLD1[0] means Message object No.1 is valid or not. If CAN_MVLD1[0] is set, message object No.1 is configured.

Message Valid Register 2 (CAN_MVLD2)

Register	Offset	R/W	Description	Reset Value
CAN_MVLD2	CAN_BA+0x164	R	Message Valid Register 2	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
MsgVal32-25							
7	6	5	4	3	2	1	0
MsgVal24-17							

Bits	Description	
[31:16]	Reserved	Reserved.
[15:0]	MsgVal32-17	**Message Valid Bits 32-17 (of All Message Objects)(Read Only)** 0 = This Message Object is ignored by the Message Handler. 1 = This Message Object is configured and should be considered by the Message Handler. Ex.CAN_MVLD2[15] means Message object No.32 is valid or not. If CAN_MVLD2[15] is set, message object No.32 is configured.

Wake-up Enable Control Register (CAN_WU_EN)

Register	Offset	R/W	Description	Reset Value
CAN_WU_EN	CAN_BA+0x168	R/W	Wake-up Enable Control Register	0x0000_0000

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

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	WAKUP_EN	**Wake-up Enable Bit** 0 = The wake-up function Disabled. 1 = The wake-up function Enabled. **Note:** User can wake-up system when there is a falling edge in the CAN_Rx pin..

Wake-up Status Register (CAN_WU_STATUS)

Register	Offset	R/W	Description	Reset Value
CAN_WU_STATUS	CAN_BA+0x16C	R/W	Wake-up Status Register	0x0000_0000

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

Bits	Description	
[31:1]	Reserved	Reserved.
[0]	WAKUP_STS	**Wake-up Status** 0 = No wake-up event occurred. 1 = Wake-up event occurred. **Note:** This bit can be cleared by writing '0'.

6.17 Hardware Divider (HDIV)

6.17.1 Overview

The hardware divider is useful to the high performance application. The hardware divider is a signed, integer divider with quotient and remainder outputs.

6.17.2 Features

- Supports Signed (two's complement) integer calculation.
- Supports 32-bit dividend with 16-bit divisor calculation capacity.
- Supports 32-bit quotient and 16-bit remainder outputs.
- Supports divided by 0 warning flag.
- 7 HCLK clocks taken for one cycle calculation.
- Software triggered with finish flag.
- Supports triggered by MDU under background automatically.

6.17.3 Functional Description

The following figure shows the HDIV clock source control.

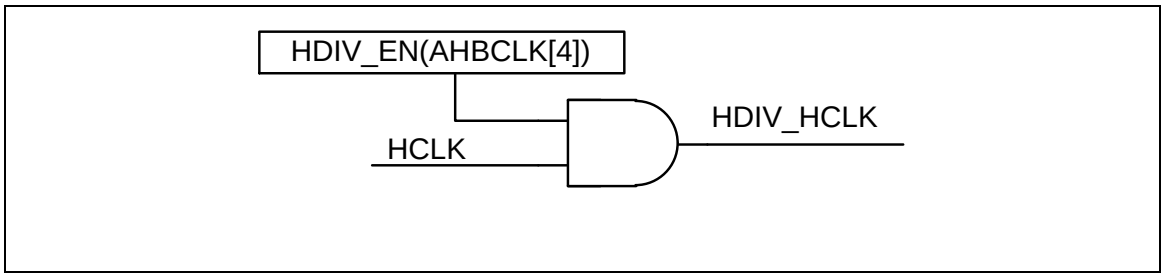


Figure 6-154 HDIV Clock Source Control

To use hardware divider, it needs to set dividend first. Then set divisor and the hardware divider will trigger calculation automatically after divisor written. The calculation results including the quotient and remainder could be got by reading DIVQUO and DIVREM register. If CPU reads DIVQUO or DIVREM before hardware divider calculation finishing, CPU will be held until hardware divider finishing the calculation. Therefore, CPU can always get valid results after trigger one hardware divider calculation without software delay.

DIV0 (DIVSTS[0]) flag of will be set if divisor is 0.

The dividend is 32-bit signed integer and divisor is 16-bit signed integer. The quotient is 32-bit signed integer and the remainder is 16-bit signed integer.

The Figure 6-155shows the operation flow of hardware divider. To calculation X / Y, CPU needs to write X to DIVIDEND register, and then write Y to DIVISOR. CPU can read DIVQUO and DIVREM registers to get calculation results after DIVISOR been written.

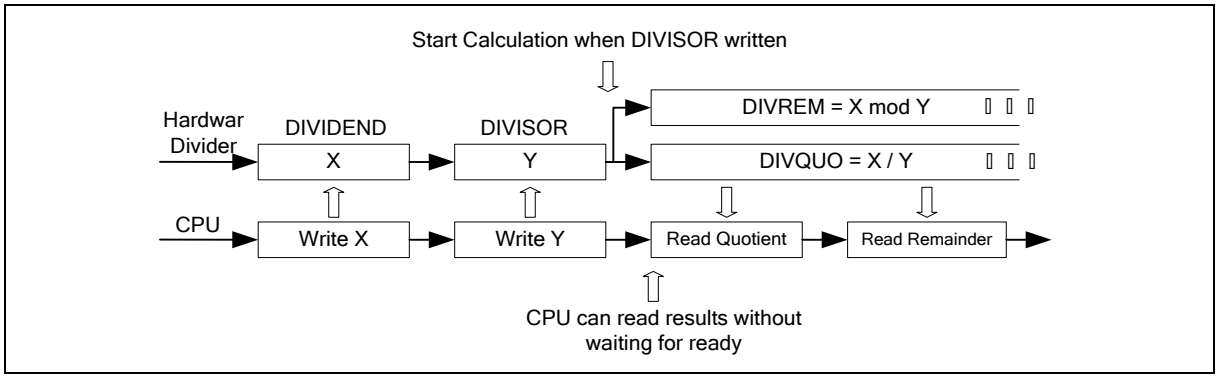


Figure 6-155 Hardware Divider Operation Flow

6.17.4 Register Map

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

Register	Offset	R/W	Description	Reset Value
HDIV Base Address: **HDIV_BA = 0x5001_4000**				
DIVIDEND	HDIV_BA+0x04	R/W	DividendSource Register(Signed 32-bit)	0x0000_0000
DIVISOR	HDIV_BA+0x08	R/W	DivisorSource Resister (Signed 16-bit)	0x0000_FFFF
DIVQUO	HDIV_BA+0x0C	R	QuotientResult Resister (Signed 32-bit)	0x0000_0000
DIVREM	HDIV_BA+0x10	R	ReminderResult Register (Signed 16-bit)	0x0000_0000
DIVSTS	HDIV_BA+0x14	R/W	Divider Status Register	0x0000_0001

6.17.5 Register Description

Divider Dividend Source Register (DIVIDEND)

Register	Offset	R/W	Description	Reset Value
DIVIDEND	HDIV_BA+0x04	R/W	DividendSource Register (Signed 32-bit)	0x0000_0000

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

Bits	Description	
[31:0]	**Dividend**	**Dividend Source** This register is given the dividend (signed 32-bit) of divider before calculation starts.

Divider Divisor Source Register (DIVISOR)

Register	Offset	R/W	Description	Reset Value
DIVISOR	HDIV_BA+0x08	R/W	DivisorSource Resister (Signed 16-bit)	0x0000_FFFF

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**Divisor**	**Divisor Source** This register is given the divisor of divider before calculation starts. **Note:** When this register is written, hardware divider will start calculation.

Divider Quotient Result Register (DIVQUO)

Register	Offset	R/W	Description	Reset Value
DIVQUO	HDIV_BA+0x0C	R	QuotientResult Resister (Signed 32-bit)	0x0000_0000

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

Bits	Description	
[31:0]	**Quotient**	**Quotient Result** This register holds the quotient (signed 32-bit) result of divider after calculation completed.

Divider Reminder Result Register (DIVREM)

Register	Offset	R/W	Description	Reset Value
DIVREM	HDIV_BA+0x10	R	ReminderResult Register (Signed 16-bit)	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**Reminder**	**Reminder Result** This register holds the reminder (signed 16-bit) result of divider after calculation completed.

Divider Status Register (DIVSTS)

Register	Offset	R/W	Description	Reset Value
DIVSTS	HDIV_BA+0x14	R/W	Divider Status 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					**DIVFF**	**DIV0**	**DIV_FINISH**

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**DIVFF**	**Divider Operation Finish Flag** When divider calculation has finished, this bit is set to 1. This bit is cleared to 0 by writing 1 to it through software
[1]	**DIV0**	**Divisor Zero Warning** 0 = The divisor is not 0. 1 = The divisor is 0. This register is read only.
[0]	**DIV_FINISH**	**Divider Operation Finished** 0 = The divider calculation not finished yet. 1 = The divider calculation finished. This register is read only.

6.18 Motor Drive Unit(MDU)

6.18.1 Overview

An efficient motor control scheme, Field Orientated Control (FOC), is widely applied to a three phases AC Induction Motor (ACIM) and Permanent Magnet Synchronous Motor (PMSM). This device provides a motor drive unit (MDU) which performs the Field Orientated Control (FOC) through hardware and offers the Space Vector PWM duty timing to PWM unit0 to produce the PWM signals to power inverter.

6.18.2 Features

- Clarke and Inverse Clarke Transformation
- Park and Inverse Park Transformation
- Twobuilt-in PI (Proportional and Integral) controllers
- Space-vector PWM (SVPWM) timing generator
- Arithmetic operation in fixed point Q-15 format
- Auto update PWM Unit0 duty registers

6.18.3 MDU Architecture

The Motor Drive Unit comprises several function blocks including Clarke, Park, Inverse Clarke and Inverse Park transformations, two PI controllers and SVPWM timing generator.

The Figure 6–156 and Figure 6–157illustrate the MDU clock source control and the architecture of Motor Drive Unit Controller.

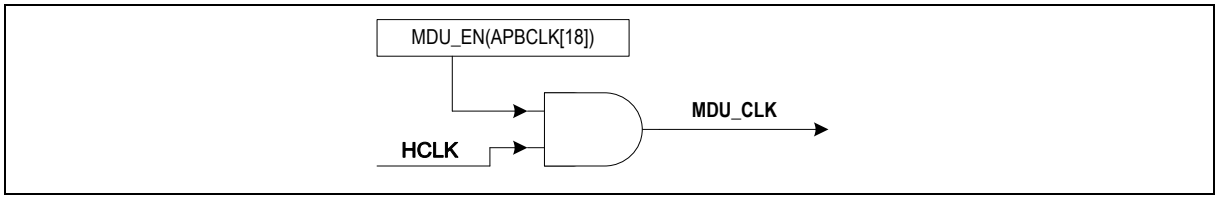


Figure 6–156 MDU Clock Source Control

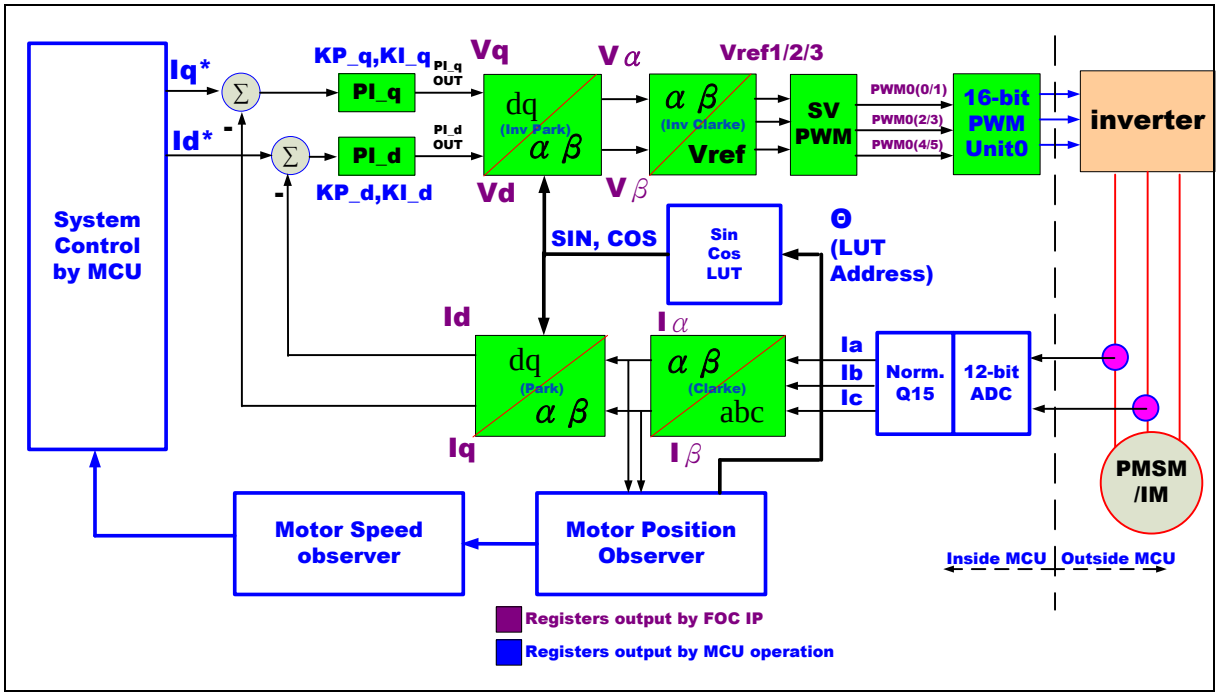


Figure 6–157 Motor Drive Unit Architecture

6.18.4 Basic Configuration

The peripheral clock source of MDU can be enabled in MDU_EN (APBCLK[18]).

6.18.5 Operation of Motor Drive Unit

The Field Orientated Control (FOC) is based on projections which transform a three phase time and speed dependent system into a two co-ordinate (d and q co-ordinates) time invariant system.

All the arithmetical values in MDU, except KP0 and KP1, are normalized in a 16-bit Q15 format with one sign bit in MSB. KP0 and KP1 are represented in 18-bit I2Q15 format.

6.18.5.1 Clarke and Inverse Clarke Transformation

The Clarke transformation function block transforms the three variables registers Ia, Ib and Ic from abc-axis to stationary αβ-axis and produces two variables stored in registers Ialfa and Ibeta. The inverse Clarke transformation takes the inverse operation.

6.18.5.2 Park and Inverse Park Transformation

The Park transformation function block transforms the two variables Valfa and Vbeta from stationary αβ-axis to rotating **dq**-axis and produces two variables stored in registers Vd and Vq. The inverse Park transformation takes the inverse operation.

6.18.5.3 PI Controller

The PI controller executes current control by proportional and integral scheme to obtain a proper controlled phase voltage applied to the Motor.

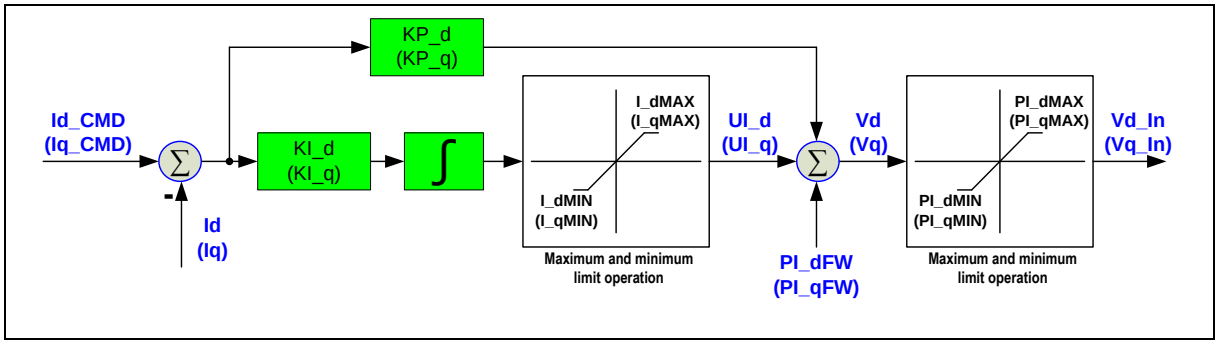


Figure 6–158 PI Controller Architecture

6.18.5.4 SVPWM Timing Generator

If the control bit SVPWM_DIS (MDUCON[17]) is cleared, the inverse Clarke Transformation is integrated into the Space Vector PWM function block to produce three PWM duties which are applied to PWM unit0 duty double buffer registers. Consequently, PWM unit 0 may output the frequency and amplitude adjustable sinusoidal signal modulated by SVPWM scheme.

If the control bit SVPWM_DIS is set, it will produce 3 balanced sinusoidal PWM (SPWM) duties for PWM modules.

6.18.6 Register Map

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

Register	Offset	R/W	Description	Reset Value
MDU Base Address: **MDUx_BA = 0x401D_0000 + (0x0400 * x)** **x = 0, 1** **MDUG_BA = 0x401D_0800**				
IA	MDUx_BA+0x00	R/W	Phase A Current Register, Represented in Q-15 Format	0x0000_0000
IB	MDUx_BA+0x04	R/W	Phase B Current Register, Represented in Q-15 Format	0x0000_0000
IC	MDUx_BA+0x08	R/W	Phase C Current Register, Represented in Q-15 Format	0x0000_0000
IALFA	MDUx_BA+0x0C	R/W	Clarke Transformation Output Register, the Current Along alpha-axis	0x0000_0000
IBETA	MDUx_BA+0x10	R/W	Clarke Transformation Output Register, the Current Along beta-axis	0x0000_0000
SIN	MDUx_BA+0x14	R/W	Trigonometric Value of SIN Register, Represented in Q-15 Format	0x0000_0000
COS	MDUx_BA+0x18	R/W	Trigonometric Value of COS Register, Represented in Q-15 Format	0x0000_0000
ID	MDUx_BA+0x1C	R/W	Park Transformation Output Register, the Current Along d-axis.	0x0000_0000
IQ	MDUx_BA+0x20	R/W	Park Transformation Output Register, the Current Along q-axis.	0x0000_0000
ID_CMD	MDUx_BA+0x24	R/W	PI_d Controller Command Input Register,Represented in Q-15 Format.	0x0000_0000
IQ_CMD	MDUx_BA+0x28	R/W	PI_q Controller Command Input Register,Represented in Q-15 Format.	0x0000_0000
KP_D	MDUx_BA+0x2C	R/W	Parameter KP for PI_d Controller,Represented in 18-bit I2Q-15 (Q2.15) Format	0x0000_0000
KI_D	MDUx_BA+0x30	R/W	Parameter KI for PI_d Controller,Represented in 18-bit I2Q-15 (Q2.15) Format	0x0000_0000
KP_Q	MDUx_BA+0x34	R/W	Parameter KP for PI_q Controller,Represented in 18-bit I2Q-15 (Q2.15) Format	0x0000_0000
KI_Q	MDUx_BA+0x38	R/W	Parameter KI for PI_q Controller,Represented in 18-bit I2Q-15 (Q2.15) Format	0x0000_0000
PI_DFW	MDUx_BA+0x3C	R/W	PI_d Controller Feed Forward Data Register	0x0000_0000
PI_QFW	MDUx_BA+0x40	R/W	PI_q Controller Feed Forward Data Register	0x0000_0000
UI_D	MDUx_BA+0x44	R/W	D-axis I Control Output Data Register	0x0000_0000
UI_Q	MDUx_BA+0x48	R/W	Q-axis I Control Output Data Register	0x0000_0000
VD	MDUx_BA+0x4C	R/W	PI_d Controller Output of d-axis,Represented in Q-15 Format	0x0000_0000
VQ	MDUx_BA+0x50	R/W	PI_q Controller Output of q-axis,Represented in Q-15 Format	0x0000_0000
VALFA	MDUx_BA+0x54	R/W	Inverse Park Transformation Output, the Voltage Along the alpha-axis	0x0000_0000

Register	Offset	R/W	Description	Reset Value
MDU Base Address: **MDUx_BA = 0x401D_0000 + (0x0400 * x)** **x = 0, 1** **MDUG_BA = 0x401D_0800**				
VBETA	MDUx_BA+0x58	R/W	Inverse Park Transformation Output, the Voltage Along the beta-axis	0x0000_0000
VREF1	MDUx_BA+0x5C	R/W	Inverse Clarke Transformation Output Register	0x0000_0000
VREF2	MDUx_BA+0x60	R/W	Inverse Clarke Transformation Output Register	0x0000_0000
VREF3	MDUx_BA+0x64	R/W	Inverse Clarke Transformation Output Register	0x0000_0000
SVPDATA0	MDUx_BA+0x68	R/W	SVPWM output Data Register (it is an unsigned 16-bit value and denotes Taon time)	0x0000_0000
SVPDATA2	MDUx_BA+0x6C	R/W	SVPWM output Data Register (it is an unsigned 16-bit value and denotes Tbon time)	0x0000_0000
SVPDATA4	MDUx_BA+0x70	R/W	SVPWM output Data Register (it is an unsigned 16-bit value and denotes Tcon time)	0x0000_0000
SVPDATA0N	MDUx_BA+0x74	R/W	An unsigned16-bit data Normalizes SVPDATA0 to PWMNORM and within the limit between PWMMAX and PWMMIN	0x0000_0000
SVPDATA2N	MDUx_BA+0x78	R/W	An unsigned 16-bit data Normalizes SVPDATA2 to PWMNORM and within the limit between PWMMAX and PWMMIN	0x0000_0000
SVPDATA4N	MDUx_BA+0x7C	R/W	An unsigned 16-bit data Normalizes SVPDATA4 to PWMNORM and within the limit between PWMMAX and PWMMIN	0x0000_0000
MDUCON	MDUx_BA+0x80	R/W	Motor Driver Unit Control Register	0x0000_0000
MDUSTS	MDUx_BA+0x84	R/W	Motor Driver Unit Status Register	0x0000_0700
PI_DMAX	MDUx_BA+0x100	R/W	Maximum limit value of PI_d Controller in Q-15 Format	0x0000_7FFF
PI_DMIN	MDUx_BA+0x104	R/W	Minimum limit value of PI_d Controller in Q-15 Format	0x0000_8000
PI_QMAX	MDUx_BA+0x108	R/W	Maximum limit value of PI_q Controller in Q-15 Format	0x0000_7FFF
PI_QMIN	MDUx_BA+0x10C	R/W	Minimum limit value of PI_q Controller in Q-15 Format	0x0000_8000
I_DMAX	MDUx_BA+0x110	R/W	Maximum limit value of integral_d Controller in Q-15 Format	0x0000_7FFF
I_DMIN	MDUx_BA+0x114	R/W	Minimum limit value of integral_d Controller in Q-15 Format	0x0000_8000
I_QMAX	MDUx_BA+0x118	R/W	Maximum limit value of integral_q Controller in Q-15 Format	0x0000_7FFF
I_QMIN	MDUx_BA+0x11C	R/W	Minimum limit value of integral_q Controller in Q-15 Format	0x0000_8000
VD_IN	MDUx_BA+0x120	R	Inverse Park Transformation Input of d_axis represented in Q-15 Format	0x0000_0000
VQ_IN	MDUx_BA+0x124	R	Inverse Park Transformation Input of q_axis represented in Q-15 Format	0x0000_0000
PWMNORM	MDUx_BA+0x128	R/W	An Unsigned 16-bit PWM Normalization Value	0x0000_0000
PWMMAX	MDUx_BA+0x12C	R/W	An unsigned 16-bit Maximum Limit Value of SVPWM Output	0x0000_FFFF
PWMMIN	MDUx_BA+0x130	R/W	An unsigned 16-bit Minimum Limit Value of SVPWM Output	0x0000_0000
ID_ERR	MDUx_BA+0x134	R	Error of Id current = Id_CMD – Id; Read only	0x0000_0000

Register	Offset	R/W	Description	Reset Value
MDU Base Address: **MDUx_BA = 0x401D_0000 + (0x0400 * x)** **x = 0, 1** **MDUG_BA = 0x401D_0800**				
IQ_ERR	MDUx_BA+0x138	R	Error of Id current = Iq_CMD – Iq; Read only.	0x0000_0000
MDUSCON	MDUG_BA+0x00	R/W	MDU Switch Control	0x0000_0000
MDUSSTS	MDUG_BA+0x04	R/W	MDU Switch Status	0x0000_0000

6.18.7 Register Description

Phase A/B/C Current Register (Ia / Ib / Ic)

Register	Offset	R/W	Description	Reset Value
IA	MDUx_BA+0x00	R/W	Phase A Current Register, Represented in Q-15 Format	0x0000_0000
IB	MDUx_BA+0x04	R/W	Phase B Current Register, Represented in Q-15 Format	0x0000_0000
IC	MDUx_BA+0x08	R/W	Phase C Current Register, Represented in Q-15 Format	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**DATA**	**Motor Phase Current Data Register** This register stores the phase current data represented in Q-15 format. The MCU must update registers Ia, Ib and Ic in the beginning of a FOC process flow.

Clarke Transformation Output Register (Ialfa / Ibeta)

Register	Offset	R/W	Description	Reset Value
IALFA	MDUx_BA+0x0C	R/W	Clarke Transformation Output Register, the Current Along alpha-axis	0x0000_0000
IBETA	MDUx_BA+0x10	R/W	Clarke Transformation Output Register, the Current Along beta-axis	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**DATA**	**Clarke Transformation Output Data Register** This data content is produced by Clarke Transformation function block and stored in Q-15 format. It is also writable by software.

Trigonometric Value Register (SIN / COS)

Register	Offset	R/W	Description	Reset Value
SIN	MDUx_BA+0x14	R/W	Trigonometric Value of SIN Register,Represented in Q-15 Format	0x0000_0000
COS	MDUx_BA+0x18	R/W	Trigonometric Value of COS Register,Represented in Q-15 Format	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**DATA**	**Trigonometric Value of SIN / COS Data Register** The trigonometric value of SIN and COS, which is represented in Q-15 format, must be preset before operating Park and inverse Park transformation.

Park Transformation Output Register (Id / Iq)

Register	Offset	R/W	Description	Reset Value
ID	MDUx_BA+0x1C	R/W	Park Transformation Output Register, the Current Along d-axis.	0x0000_0000
IQ	MDUx_BA+0x20	R/W	Park Transformation Output Register, the Current Along q-axis.	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**DATA**	**Park Transformation Output Data Register** This data content is produced by Park Transformation function block and stored in Q-15 format. It is also writable by software.

PI Controller Command Input Register (Id_CMD / Iq_CMD)

Register	Offset	R/W	Description	Reset Value
ID_CMD	MDUx_BA+0x24	R/W	PI_d Controller Command Input Register,Represented in Q-15 Format.	0x0000_0000
IQ_CMD	MDUx_BA+0x28	R/W	PI_q Controller Command Input Register,Represented in Q-15 Format.	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**CMD**	**PI Controller Command Input Register** The register content stands for the PI controller command input. It must be normalized in Q-15 format.

PI Controller Parameter Register (KP_q / KI_q / KP_d / KI_d)

Register	Offset	R/W	Description	Reset Value
KP_D	MDUx_BA+0x2C	R/W	Parameter KP for PI_d Controller,Represented in 18-bit I2Q-15 (Q2.15) Format	0x0000_0000
KI_D	MDUx_BA+0x30	R/W	Parameter KI for PI_d Controller,Represented in 18-bit I2Q-15 (Q2.15) Format	0x0000_0000
KP_Q	MDUx_BA+0x34	R/W	Parameter KP for PI_q Controller,Represented in 18-bit I2Q-15 (Q2.15) Format	0x0000_0000
KI_Q	MDUx_BA+0x38	R/W	Parameter KI for PI_q Controller,Represented in 18-bit I2Q-15 (Q2.15) Format	0x0000_0000

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

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17:0]	**DATA**	**PI Controller Parameter Register** The KP_d and KP_q stand for the proportional parameter of PI controller, the KI_d and KI_q stand for the integral parameter of PI controller. The register data value must be stored in Q-15 format.

PI Controller Feed Forward Data Register (PI_dFW / PI_qFW)

Register	Offset	R/W	Description	Reset Value
PI_DFW	MDUx_BA+0x3C	R/W	PI_d Controller Feed Forward Data Register	0x0000_0000
PI_QFW	MDUx_BA+0x40	R/W	PI_q Controller Feed Forward 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**
DATA							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
DATA							

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**DATA**	**PI Controller Feed Forward Data Register** Software presets the feed forward value of the PI controller in this register.

Integral Controller Output Data Register (UI_d / UI_q)

Register	Offset	R/W	Description	Reset Value
UI_D	MDUx_BA+0x44	R/W	D-axis I Control Output Data Register	0x0000_0000
UI_Q	MDUx_BA+0x48	R/W	Q-axis I Control Output Data Register	0x0000_0000

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

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20:0]	**DATA**	**Integral Controller Output Data Register** The PI function block content I (Integral) controller and P (Propotional) controller, and I controller output data will be used as the next PI control flow. Here is the register to save the output data of I controller and will update when PI controller finished once operation. **Here, the register content is a Q-15 format value (PI_Q20_EN = 0) or Q-20 format value(PI_Q20_EN = 1).**

PI Controller Output Register (Vd / Vq)

Register	Offset	R/W	Description	Reset Value
VD	MDUx_BA+0x4C	R/W	PI_d Controller Output of d-axis,Represented in Q-15 Format	0x0000_0000
VQ	MDUx_BA+0x50	R/W	PI_q Controller Output of q-axis,Represented in Q-15 Format	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**DATA**	**PI Controller Output Data Register (Inverse Park Transformation Input)** This data content is produced by PI controller function block and stored in Q-15 format. It is also writable by software.

Inverse Park Transformation Output Register (Valfa / Vbeta)

Register	Offset	R/W	Description	Reset Value
VALFA	MDUx_BA+0x54	R/W	Inverse Park Transformation Output, the Voltage Along the alpha-axis	0x0000_0000
VBETA	MDUx_BA+0x58	R/W	Inverse Park Transformation Output, the Voltage Along the beta-axis	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**DATA**	**Inverse Park Transformation Output Data Register** This data content is produced by inverse Park Transformation function block and stored in Q-15 format. It is also writable by software.

Inverse Clarke Transformation Output Register (Valfa / Vbeta)

Register	Offset	R/W	Description	Reset Value
VREF1	MDUx_BA+0x5C	R/W	Inverse Clarke Transformation Output Register	0x0000_0000
VREF2	MDUx_BA+0x60	R/W	Inverse Clarke Transformation Output Register	0x0000_0000
VREF3	MDUx_BA+0x64	R/W	Inverse Clarke Transformation Output Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**DATA**	**Inverse Clarke Transformation Output Data Register** This data content is produced by inverse Clarke Transformation function block and stored in Q-15 format. It is also writable by software.

SVPWM Output Data Register (SVPWM0 / SVPWM2 / SVPWM4)

Register	Offset	R/W	Description	Reset Value
SVPDATA0	MDUx_BA+0x68	R/W	SVPWM output Data Register (it is an unsigned 16-bit value and denotes Taon time)	0x0000_0000
SVPDATA2	MDUx_BA+0x6C	R/W	SVPWM output Data Register (it is an unsigned 16-bit value and denotes Tbon time)	0x0000_0000
SVPDATA4	MDUx_BA+0x70	R/W	SVPWM output Data Register (it is an unsigned 16-bit value and denotes Tcon time)	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**DATA**	**SVPWM Output Data Register** This data content is produced by SVPWM function block and stored in Q-15 format. It is also writable by software.

Normalized SVPWM Output Register (SVPWM0N / SVPWM2N / SVPWM4N)

Register	Offset	R/W	Description	Reset Value
SVPDATA0N	MDUx_BA+0x74	R/W	An unsigned16-bit data Normalizes SVPDATA0 to PWMNORM and within the limit between PWMMAX and PWMMIN	0x0000_0000
SVPDATA2N	MDUx_BA+0x78	R/W	An unsigned 16-bit data Normalizes SVPDATA2 to PWMNORM and within the limit between PWMMAX and PWMMIN	0x0000_0000
SVPDATA4N	MDUx_BA+0x7C	R/W	An unsigned 16-bit data Normalizes SVPDATA4 to PWMNORM and within the limit between PWMMAX and PWMMIN	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**DATA**	**Normalized SVPDATA Register** The SVPWM function block will normalizes the value of SVP0/2/4 to the range between 0000h and the value of PWMNORM and limit the PWM duty value within PWMMAX and PWMMIN. If SVPWM function block auto mode is active, each new SVPDATA0N/2N/4N will be reload to PWM unit 0 duty registers PWM0/2/4 after SVPWM operation is completed if PWMRUN and LOAD bits are enabled in PWM unit 0. Here, the register content is a 16-bit unsigned integer.

Motor Driver Unit Control Register (MDUCON)

Register	Offset	R/W	Description	Reset Value
MDUCON	MDUx_BA+0x80	R/W	Motor Driver Unit Control Register	0x0000_0000

31	30	29	28	27	26	25	24
MDU_IE	SVPWM_IE	INVCK_IE	INVPK_IE	PIQ_IE	PID_IE	PK_IE	CK_IE
23	22	21	20	19	18	17	16
Reserved			PI_Q20_EN	Reserved		SVPWM_DIS	ATCLRFG
15	14	13	12	11	10	9	8
Reserved	SVPWMAUTO	INVCKAUTO	INVPKAUTO	PIQAUTO	PIDAUTO	PKAUTO	Reserved
7	6	5	4	3	2	1	0
Reserved	SVPWMSTR	INVCKSTR	INVPKSTR	PIQSTR	PIDSTR	PKSTR	CKSTR

Bits	Description	
[31]	**MDU_IE**	**MDU Interrupt Enable Bit** 0 = MDU interrupt Disabled. 1 = MDU interrupt Enabled.
[30]	**SVPWM_IE**	**Clark Block Interrupt Enable Bit** 0 = SVPWM block interrupt Disabled. 1 = SVPWM block interrupt Enabled. Interrupt will generate if MDUINT_EN and SVPWM_IE is set to 1 and SVPWMCPF is set
[29]	**INVCK_IE**	**Inv-clark Block Interrupt Enable Bit** 0 = Inv-Clark block interrupt Disabled. 1 = Inv-Clark block interrupt Enabled.
[28]	**INVPK_IE**	**Inv-park Block Interrupt Enable Bit** 0 = Inv-Park block interrupt Disabled. 1 = Inv-Park block interrupt Enabled.
[27]	**PIQ_IE**	**Q-axis PI Block Interrupt Enable Bit** 0 = Q-axis PI controller block interrupt Disabled. 1 = Q-axis PI controller block interrupt Enabled.
[26]	**PID_IE**	**D-axis PIBlock Interrupt Enable Bit** 0 = D-axis PI controller block interrupt Disabled. 1 = D-axis PI controller block interrupt Enabled.
[25]	**PK_IE**	**Park Block Interrupt Enable Bit** 0 = Park block interrupt Disabled. 1 = Park block interrupt Enabled.
[24]	**CK_IE**	**Clark Block Interrupt Enable Bit** 0 = Clark block interrupt Disabled. 1 = Clark block interrupt Enabled.

Bits	Description	
[23:21]	**Reserved**	Reserved.
[20]	**PI_Q20_EN**	**PI Controller Q20 Format Enable Bit** 0 = PI controller Q20 format Disabled. 1 = PI controller Q20 format Enabled.
[19:18]	**Reserved**	Reserved.
[17]	**SVPWM_DIS**	**SVPWM Block Function Disable Bit** 0 = SVPWM function block produces the Space Vector PWM timing for PWM duty registers. 1 = SVPWM operation Disabled.
[16]	**ATCLRFG**	**Auto-clear Flag Control Bit** When this bit is set to high, the complete flag of **each function block in auto-mode** is cleared automatically after SVPWM function block has updated PWM duty registers. The auto-clear flag function helps the FOC control flow runs smoothly without the need of software clearing flags. 0 = Auto-clear flag function Disabled. 1 = Auto-clear flag function Enabled.
[15]	**Reserved**	Reserved.
[14]	**SVPWMAUTO**	**SVPWM Auto-mode Enable Bit** Setting this bit will enable auto-mode of the function block. If the auto-mode is active, the SVPWM function block is allowed to accept the start trigger from Inverse Clarke block and SVPWM function proceeds completed, it will automatically update the duty registers in PWM unit0 and set the reload bit (LOAD). 0 = Auto-mode Disabled. 1 = Auto-mode Enabled.
[13]	**INVCKAUTO**	**Inverse Clarke Transformation Auto-mode Enable Bit** Setting this bit will enable auto-mode of the function block. If the auto-mode is active, the current function block is allowed to accept the previous start trigger and will trigger the start bit of the next function block after current function is completed. 0 = Auto-mode Disabled. 1 = Auto-mode Enabled.
[12]	**INVPKAUTO**	**Inverse Park Transformation Auto-mode Enable Bit** Setting this bit will enable auto-mode of the function block. If the auto-mode is active, the current function block is allowed to accept the previous start trigger and will trigger the start bit of the next function block after current function is completed. 0 = Auto-mode Disabled. 1 = Auto-mode Enabled.
[11]	**PIQAUTO**	**D-axis PI Controller Auto-mode Enable Bit** Setting this bit will enable auto-mode of the function block. If the auto-mode is active, the current function block is allowed to accept the previous start trigger and will trigger the start bit of the next function block after current function is completed. 0 = Auto-mode Disabled. 1 = Auto-mode Enabled.

Bits	Description	
[10]	**PIDAUTO**	**D-axis PI Controller Auto-mode Enable Bit** Setting this bit will enable auto-mode of the function block. If the auto-mode is active, the current function block is allowed to accept the previous start trigger and will trigger the start bit of the next function block after current function is completed. 0 = Auto-mode Disabled. 1 = Auto-mode Enabled.
[9]	**PKAUTO**	**Park Transformation Auto-mode Enable Bit** Setting this bit will enable auto-mode of the function block. If the auto-mode is active, the current function block is allowed to accept the previous start trigger and will trigger the start bit of the next function block after current function is completed. 0 = Auto-mode Disabled. 1 = Auto-mode Enabled.
[7:8]	**Reserved**	Reserved.
[6]	**SVPWMSTR**	**SVPWM Start Bit (Write Only, Read 0)** Setting this bit to high will start proceeding SVPWM timing calculating and it is automatically cleared when the process is completed by hardware. Writing it to 0 will stop and reset the function block. 0 = No action. 1 = Start the process.
[5]	**INVCKSTR**	**Inverse Clarke Transformation Start Bit (Write Only, Read 0)** Setting this bit to high will start proceeding Inverse Clarke Transformation and it is automatically cleared when the transform process is completed by hardware. Writing it to 0 will stop and reset the function block. 0 = No action. 1 = Start the process.
[4]	**INVPKSTR**	**Inverse Park Transformation Start Bit (Write Only, Read 0)** Setting this bit to high will start proceeding Inverse Park Transformation and it is automatically cleared when the transform process is completed by hardware. Writing it to 0 will stop and reset the function block. 0 = No action. 1 = Start the process.
[3]	**PIQSTR**	**Q-axis PI Controller Start Bit (Write Only, Read 0)** Setting this bit to high will start proceeding Q-axis PI Control and it is automatically cleared when the process is completed by hardware. Writing it to 0 will stop and reset the function block. 0 = No action. 1 = Start the process.
[2]	**PIDSTR**	**D-axis PI Controller Start Bit (Write Only, Read 0)** Setting this bit to high will start proceeding D-axis PI Control and it is automatically cleared when the process is completed by hardware. Writing it to 0 will stop and reset the function block. 0 = No action. 1 = Start the process.

Bits	Description	
[1]	**PKSTR**	**Park Transformation Start Bit (Write Only, Read 0)** Setting this bit to high will start proceeding Park Transformation and it is automatically cleared when the transform process is completed by hardware. Writing it to 0 will stop and reset the function block. 0 = No action. 1 = Start the process.
[0]	**CKSTR**	**Clarke Transformation Start Bit (Write Only, Read 0)** Setting this bit to high will start proceeding Clarke Transformation and it is automatically cleared when the transform process is completed by hardware. Writing it to 0 will stop and reset the function block. 0 = No action. 1 = Start the process.

Motor Driver Unit Status Register (MDUSTS)

Register	Offset	R/W	Description	Reset Value
MDUSTS	MDUx_BA+0x84	R/W	Motor Driver Unit Status Register	0x0000_0700

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved					**ZONE**		
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**SVPWMCPF**	**INVCKCPF**	**INVPKCPF**	**PI_dCPF**	**PI_qCPF**	**PKCPF**	**CKCPF**

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10:8]	**ZONE**	**Zone Number Indicator** A 360 degree space is equally divided into six zones where each zone is with 60 degree space. The SVPWM function block will internally produce the zone number during the operation which indicates the which zone the motor electric angle stays in the moment.
[7]	**Reserved**	Reserved.
[6]	**SVPWMCPF**	**SVPWM Complete Flag** When the function block completes the process, the corresponding complete flag will be set by hardware. This flag can be cleared by a writing one to itself. **Note:** If both the control bits ATCLRFG (Auto-clear flag control bit) and SVPWMAUTO (Auto-mode enable bit) are set to high, the complete flag is cleared automatically after SVPWM function block has set PWM LOAD bit to high.
[5]	**INVCKCPF**	**Inverse Clarke Transformation Complete Flag** When the function block completes the process, the corresponding complete flag will be set by hardware. This flag can be cleared by a writing one to itself. **Note:** If both the control bits ATCLRFG (Auto-clear flag control bit) and INVCKAUTO (Auto-mode enable bit) are set to high, the complete flag is cleared automatically after SVPWM function block has set PWM LOAD bit to high.
[4]	**INVPKCPF**	**Inverse Park Transformation Complete Flag** When the function block completes the process, the corresponding complete flag will be set by hardware. This flag can be cleared by a writing one to itself. **Note:** If both the control bits ATCLRFG (Auto-clear flag control bit) and INVPKAUTO (Auto-mode enable bit) are set to high, the complete flag is cleared automatically after SVPWM function block has set PWM LOAD bit to high.
[3]	**PI_dCPF**	**PI_D Controller Complete Flag** When the function block completes the process, the corresponding complete flag will be set by hardware. This flag can be cleared by a writing one to itself. **Note:** If both the control bits ATCLRFG (Auto-clear flag control bit) and PI_dAUTO (Auto-mode enable bit) are set to high, the complete flag is cleared automatically after SVPWM function block has set PWM LOAD bit to high.

Bits	Description	
[2]	**PI_qCPF**	**PI_Q Controller Complete Flag** When the function block completes the process, the corresponding complete flag will be set by hardware. This flag can be cleared by a writing one to itself. **Note:** If both the control bits ATCLRFG (Auto-clear flag control bit) and PI_qAUTO (Auto-mode enable bit) are set to high, the complete flag is cleared automatically after SVPWM function block has set PWM LOAD bit to high.
[1]	**PKCPF**	**Park Transformation Complete Flag** When the function block completes the process, the corresponding complete flag will be set by hardware. This flag can be cleared by a writing one to itself. **Note:** If both the control bits ATCLRFG (Auto-clear flag control bit) and PKAUTO (Auto-mode enable bit) are set to high, the complete flag is cleared automatically after SVPWM function block has set PWM LOAD bit to high.
[0]	**CKCPF**	**Clarke Transformation Complete Flag** When the function block completes the process, the corresponding complete flag will be set by hardware. This flag can be cleared by a writing one to itself. **Note:** If both the control bits ATCLRFG (Auto-clear flag control bit) and CKAUTO (Auto-mode enable bit) are set to high, the complete flag is cleared automatically after SVPWM function block has set PWM LOAD bit to high.

Limitation of PI Controller Output (PI_dMAX / PI_qMAX)

Register	Offset	R/W	Description	Reset Value
PI_DMAX	MDUx_BA+0x100	R/W	Maximum limit value of PI_d Controller in Q-15 Format	0x0000_7FFF
PI_QMAX	MDUx_BA+0x108	R/W	Maximum limit value of PI_q Controller in Q-15 Format	0x0000_7FFF

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**MAX**	**Maximum Limit of PI Controller and I Controller Output** The 16-bit value represented in Q-15 format limits the maximum value of the PI controller and I controller output.

Limitation of PI Controller Output (PI_dMIN / PI_qMIN)

Register	Offset	R/W	Description	Reset Value
PI_DMIN	MDUx_BA+0x104	R/W	Minimum limit value of PI_d Controller in Q-15 Format	0x0000_8000
PI_QMIN	MDUx_BA+0x10C	R/W	Minimum limit value of PI_q Controller in Q-15 Format	0x0000_8000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**MIN**	**Minimum Limit of PI Controller and I Controller Output** The 16-bit value represented in Q-15 format limits the minimum value of the PI controller and I controller output.

Limitation of Integral Controller Output (I_dMAX / I_qMAX)

Register	Offset	R/W	Description	Reset Value
I_DMAX	MDUx_BA+0x110	R/W	Maximum limit value of integral_d Controller in Q-15 Format	0x0000_7FFF
I_QMAX	MDUx_BA+0x118	R/W	Maximum limit value of integral_q Controller in Q-15 Format	0x0000_7FFF

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

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20:0]	**MAX**	**Maximum Limit of PI Controller and I Controller Output** The 16-bit value represented in Q-15 format limits the maximum value of the integral controller output. **Here, the register content is a Q-15 format value (PI_Q20_EN = 0) or Q-20 format value(PI_Q20_EN = 1).**

Limitation of Integral Controller Output (I_dMIN / I_qMIN)

Register	Offset	R/W	Description	Reset Value
I_DMIN	MDUx_BA+0x114	R/W	Minimum limit value of integral_d Controller in Q-15 Format	0x0000_8000
I_QMIN	MDUx_BA+0x11C	R/W	Minimum limit value of integral_q Controller in Q-15 Format	0x0000_8000

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

Bits	Description	
[31:21]	**Reserved**	Reserved.
[20:0]	**MIN**	**Minimum Limit of PI Controller and I Controller Output** The 16-bit value represented in Q-15 format limits the minimum value of the integral controller output.

Inverse Park Transformation Input Register (Vd_In / Vq_In)

Register	Offset	R/W	Description	Reset Value
VD_IN	MDUx_BA+0x120	R	Inverse Park Transformation Input of d_axis represented in Q-15 Format	0x0000_0000
VQ_IN	MDUx_BA+0x124	R	Inverse Park Transformation Input of q_axis represented in Q-15 Format	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**DATA**	**Inverse Park Transformation Input Data (Read Only)** This data content is produced by the PI controller with maximum and minimum limit operation and stored in Q-15 format. It is a read only register

PWM Normalization Data Register (PWMNORM)

Register	Offset	R/W	Description	Reset Value
PWMNORM	MDUx_BA+0x128	R/W	An Unsigned 16-bit PWM Normalization Value	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PWMNORM**	**PWM Normalization Data** The SVPWM function block will normalize the value of SVP0/2/4 to the range between 0000h and the value of PWMNORM and limit the PWM duty value within PWMMAX and PWMMIN. Here, the register content is a 16-bit unsigned integer.

Maximum Limit of SVPWM Output (PWMMAX)

Register	Offset	R/W	Description	Reset Value
PWMMAX	MDUx_BA+0x12C	R/W	An unsigned 16-bit Maximum Limit Value of SVPWM Output	0x0000_FFFF

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PWMMAX**	**Maximum Limit of SVPWM Output Data Register** The SVPWM function block will normalizes the value of SVP0/2/4 to the range between 0000h and the value of PWMNORM and limit the PWM duty value within PWMMAX and PWMMIN. Here, the register content is a 16-bit unsigned integer.

Minimum Limit of SVPWM Output (PWMMIN)

Register	Offset	R/W	Description	Reset Value
PWMMIN	MDUx_BA+0x130	R/W	An unsigned 16-bit Minimum Limit Value of SVPWM Output	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:0]	**PWMMIN**	**Maximum Limit of SVPWM Output Data Register** The SVPWM function block will normalize the value of SVP0/2/4 to the range between 0000h and the value of PWMNORM and limit the PWM duty value within PWMMAX and PWMMIN. Here, the register content is a 16-bit unsigned integer.

MDU Switch Control (MDUSCON)

Register	Offset	R/W	Description	Reset Value
MDUSCON	MDUG_BA+0x00	R/W	MDU Switch 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**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved							**MDUS**

Bits	Description	
[31:1]	**Reserved**	Reserved.
[0]	**MDUS**	**MDU Switch Control Register** 0 = MDU is controlled by MDU0 control register. 1 = MDU is controlled by MDU1 control register.

MDU Switch Control (MDUSSTS)

Register	Offset	R/W	Description	Reset Value
MDUSSTS	MDUG_BA+0x04	R/W	MDU Switch 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					**MDUACT**	**MDU1BUSY**	**MDU0BUSY**

Bits	Description	
[31:3]	**Reserved**	Reserved.
[2]	**MDUACT**	**MDU Active Set** 0 = MDU0 is active. 1 = MDU1 is active.
[1]	**MDU1BUSY**	**MDU1 Busy Register** 0 = MDU1 is idle. 1 = MDU1 is busy.
[0]	**MDU0BUSY**	**MDU0 Busy Register** 0 = MDU0 is idle. 1 = MDU0 is busy.

6.19 Enhanced Analog-to-Digital Converter (EADC)

6.19.1 Overview

The NuMicro®NM1530Series contains two 12-bit successive approximation analog-to-digital converters (SAR A/D converter) with 16 input channels. The two A/D converters ADCA and ADCB can be sampled with Simultaneous or Single Sampling mode. The A/D converters can be started by software, PWM triggers, timer0~3 overflow pulse triggers, ADINT0, ADINT1 interrupt EOC pulse trigger and external STADC pin input signal.

Note: The analog input port pins must be configured as input type before the EADC function is enabled.

6.19.2 Features

- Analog input voltage range: 0~V_{REF}(Max to 5.0V).
- 12-bit resolution and 10-bit accuracy is guaranteed.
- Up to 16 single-end analog input channels.
- Two SAR ADC converters.
- Four EADC interrupts with individual interrupt vector addresses.
- Maximum EADC clock frequency: 16MHz.
- Up to 1.6M SPS conversion rate, each of ADC converter conversion time less than 1.25μs.
- Two operating modes
  - Singlesampling mode: two ADC converters run at normal operation.
  - Simultaneous sampling mode: Allow two ADC converters can be sampled simultaneously.
- An A/D conversion can be started by:
  - Writing1 to ADST(ADSSTR[n]) bit (n = 0~15) through software
  - External pin STADC
  - Timer0~3 overflow pulse triggers
  - ADINT0, ADINT1 interrupt EOC pulse triggers
  - PWM triggers
- Conversion results are held in 16 data registers with valid and overrun indicators.
- Supports 8 SAMPLE control logic modules, each of them is configurable for EADC0 converter channel EADC0_CH0~7 and trigger source.
- S Supports 8 SAMPLE control logic modules, each of them is configurable for EADC1 converter channel EADC1_CH0~7 and trigger source.
- Channel EADC0_CH0 supports 2 input sources: external analog voltage and internal OPA0 Amplifier output voltage.
- Channel EADC1_CH0 supports 2 input sources: external analog voltage and internal OPA1 Amplifier output voltage.
- Channel EADC0_CH7 supports 4 input sources: external analog voltage, internal fixed band-gap voltage, internal temperature sensor output, and analog ground.

6.19.3 Block Diagram

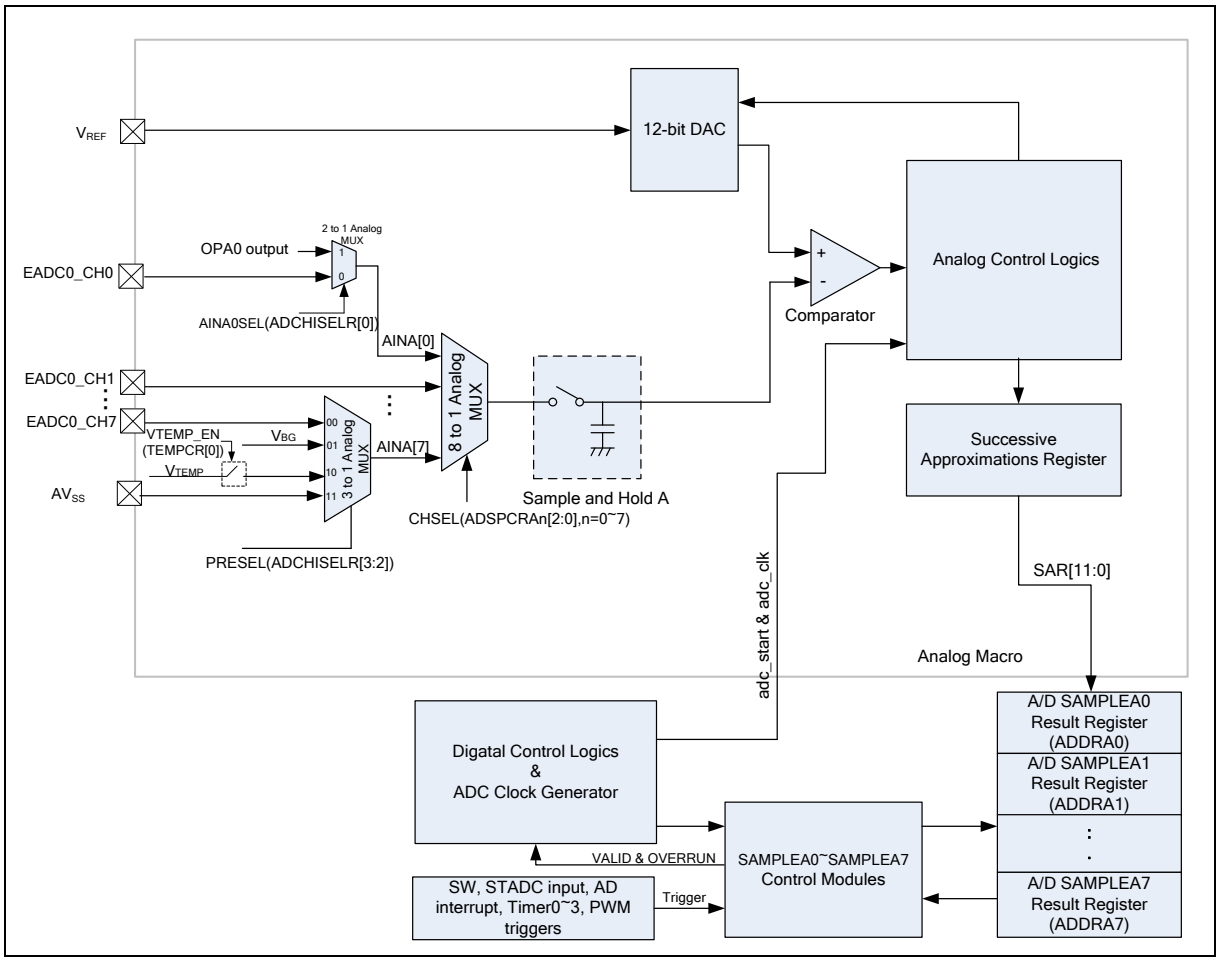


Figure 6–159 ADC0 Converter Block Diagram

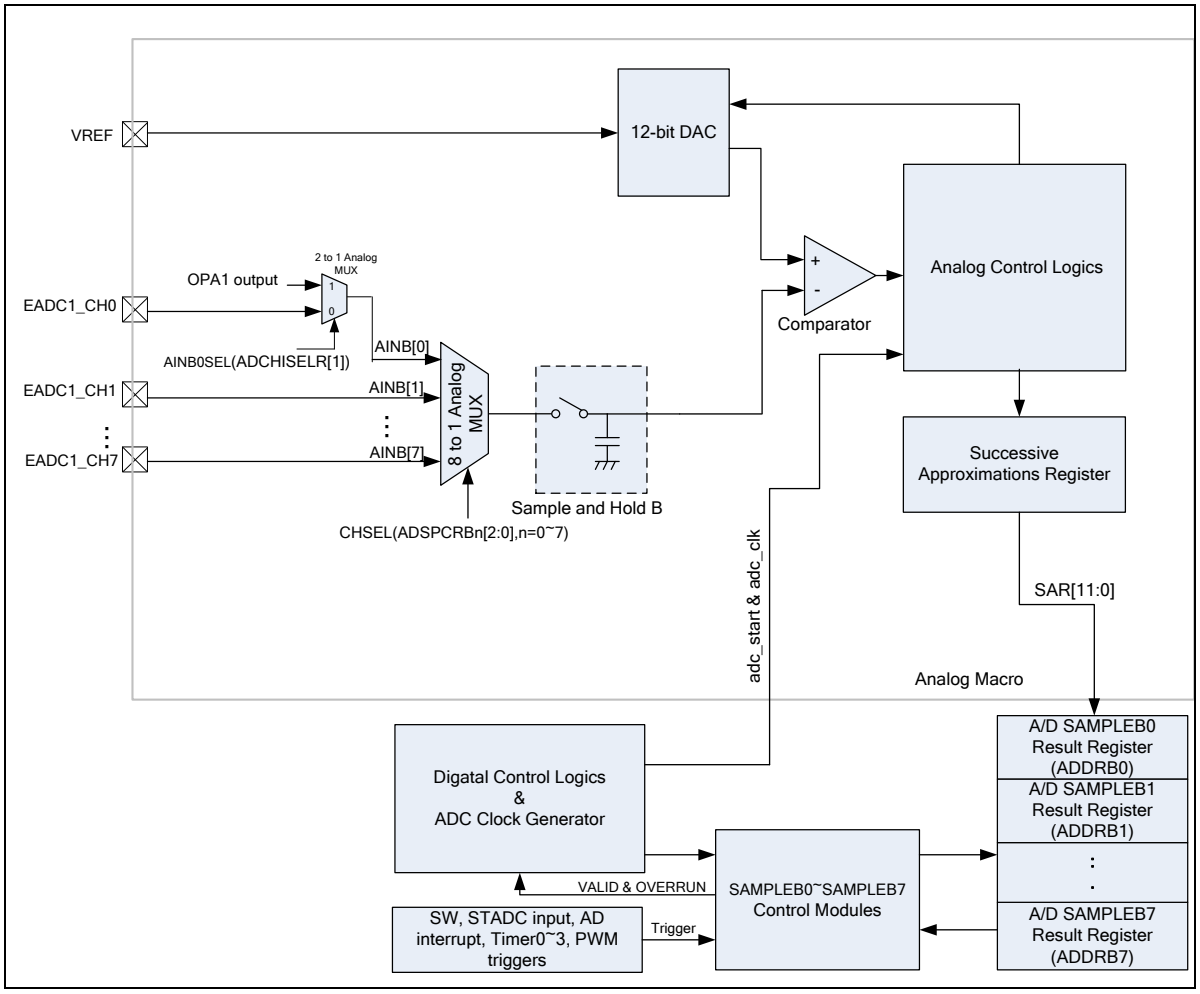


Figure 6–160 ADC1 Converter Block Diagram

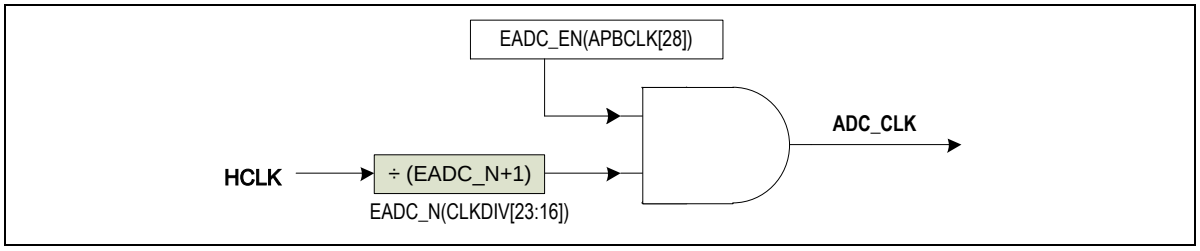


Figure 6–161 ADC Clock Control

6.19.4 Operation Procedure

There are two ADC converters, and each ADC converter consists of an eight-channel input select function and eight SAMPLE control modules, the A/D converter operates by successive approximation with 12-bit resolution.

The A/D operation is based on SAMPLEA0~7(ADCA converter) and SAMPLEB0~7(ADCB converter) control logic modules, and each of them has its configuration to decide which trigger source to start the conversion and which channel to convert. Different SAMPLE modules can be configured for the same channel, trigger source. It provides user a flexible means to get the over-sampling results.

The ADC conversion trigger sources are listed below:

- Writing 1 to ADST(ADSSTR[n]) bit (n = 0~15) by software
- External pin STADC
- Timer0~3 overflow pulse triggers
- ADINT0, ADINT1 ADC interrupt EOC pulse triggers
- PWM triggers

The ADINT0, ADINT1 interrupt pulses are generated whenever the specific SAMPLE A/D EOC (End of conversion) pulse is generated. The ADINT0, 1 interrupt pulse triggers can be fed back to trigger another A/D conversion, and is useful if a continuous scan conversion is needed.

6.19.4.1 ADC Clock Generator

The maximum sampling rate is up to 800 kHz and the conversion time is less than 1.25μs. It needs 20 EADC clocks to complete an A/D conversion. The EADC engine clock source is from HCLK clock, the EADC clock frequency is divided by an 8-bit pre-scalar with the formula:

The EADC clock frequency = (HCLK) / (EADC_N+1).

Where the 8-bit EADC_N is located in register CLKDIV[23:16].

In generally, software can set EADC_N to get 16 MHz or slightly less.

6.19.4.2 ADC Single Sampling Mode

When an ADC conversion is performed on the SAMPLEx specified single channel, the operations are as follows:

- A/D conversion is started when the ADST(ADSSTR[n]) bit in ADSSTR is set to 1 by software or other trigger inputs.
- When A/D conversion is finished, the 12-bit result is stored in the EADC data register ADDRx corresponding to the SAMPLEx.
- On completion of conversion, the ADFn bit in ADSR is set to 1 and EADC interrupt (ADINTn) is requested if the ADIEn bit is set to 1.
- The ADST(ADSSTR[n]) bit remains 1 during A/D conversion. When A/D conversion ends, the ADST(ADSSTR[n]) bit is automatically cleared to 0 and the A/D converter will do another pending conversion.

Note: If software or other trigger enables more than one channel in single or simultaneous sampling mode, the SAMPLE specified channel with highest priority is converted and other enabled channels will be pended.

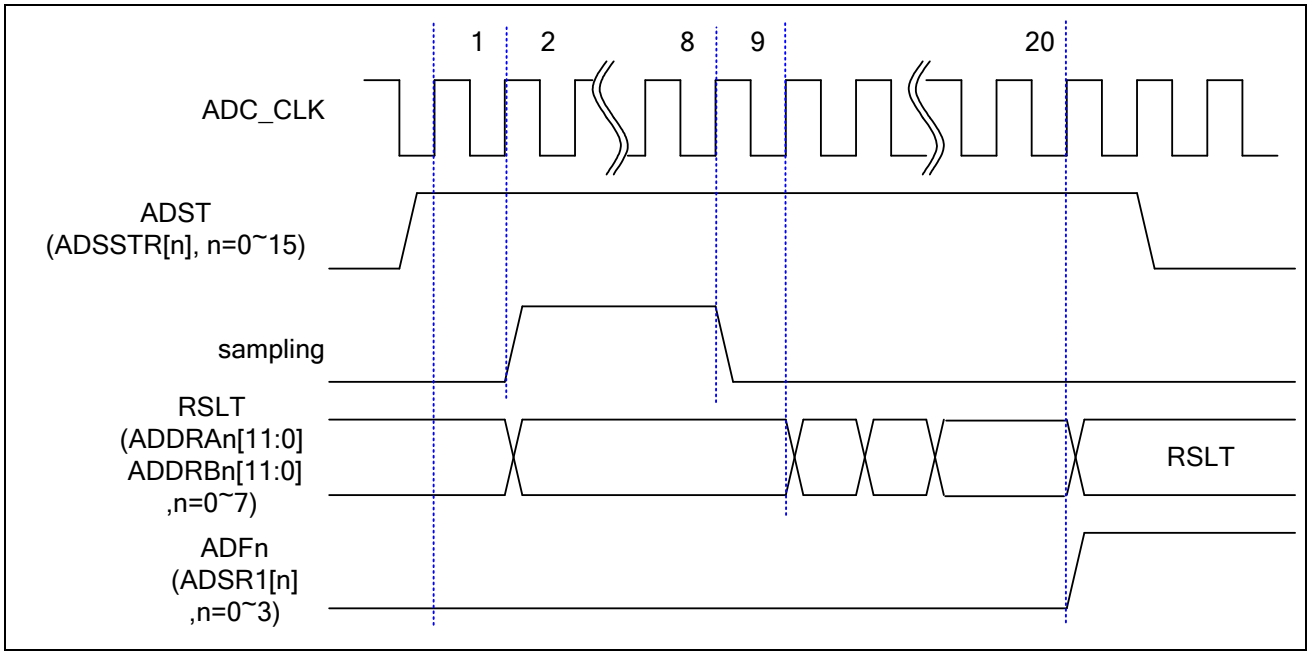


Figure 6–162 Single Sampling Mode Conversion Timing Diagram

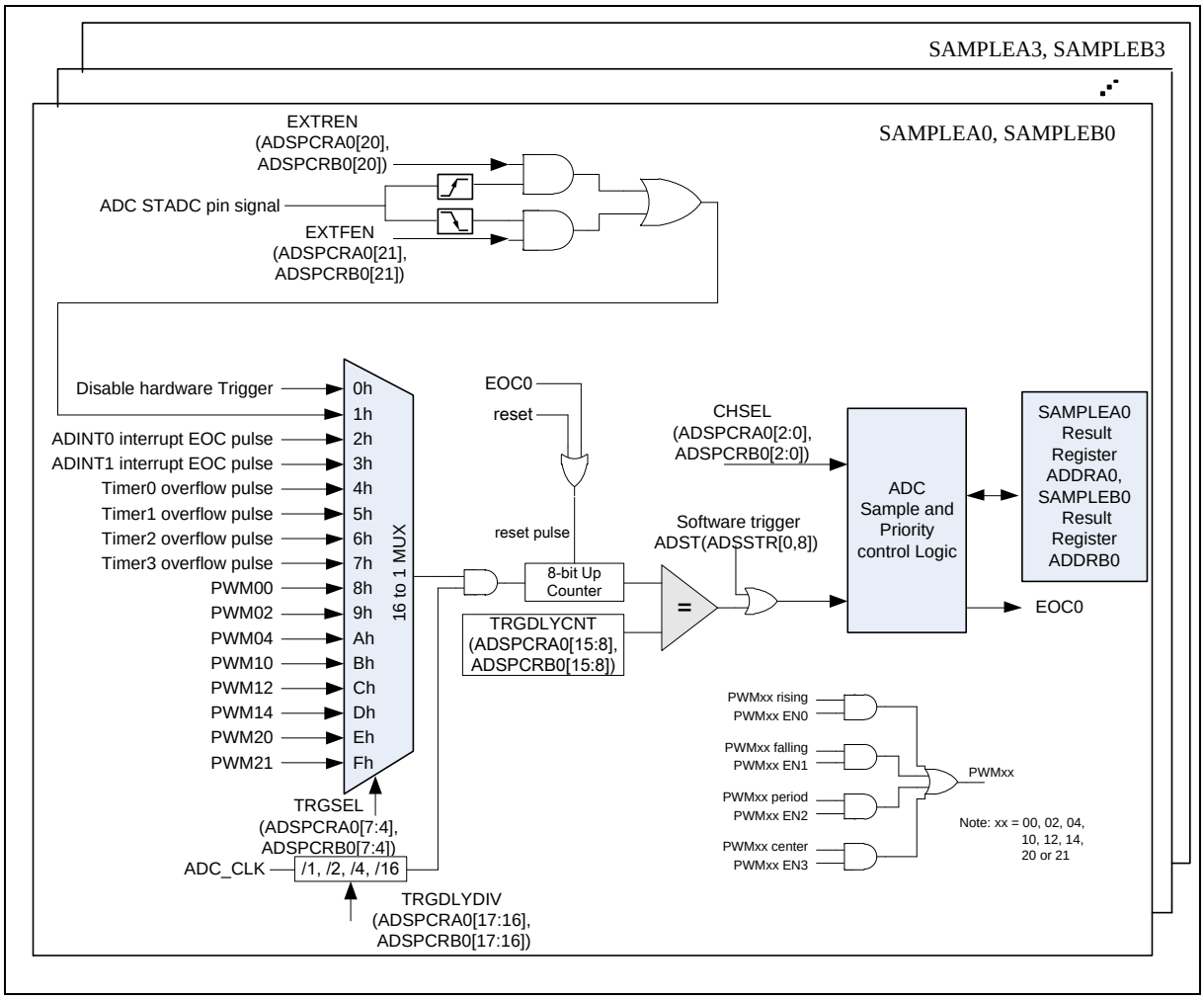


Figure 6–163 SAMPLEA0~3 and SAMPLEB0~3 Control Block Diagram

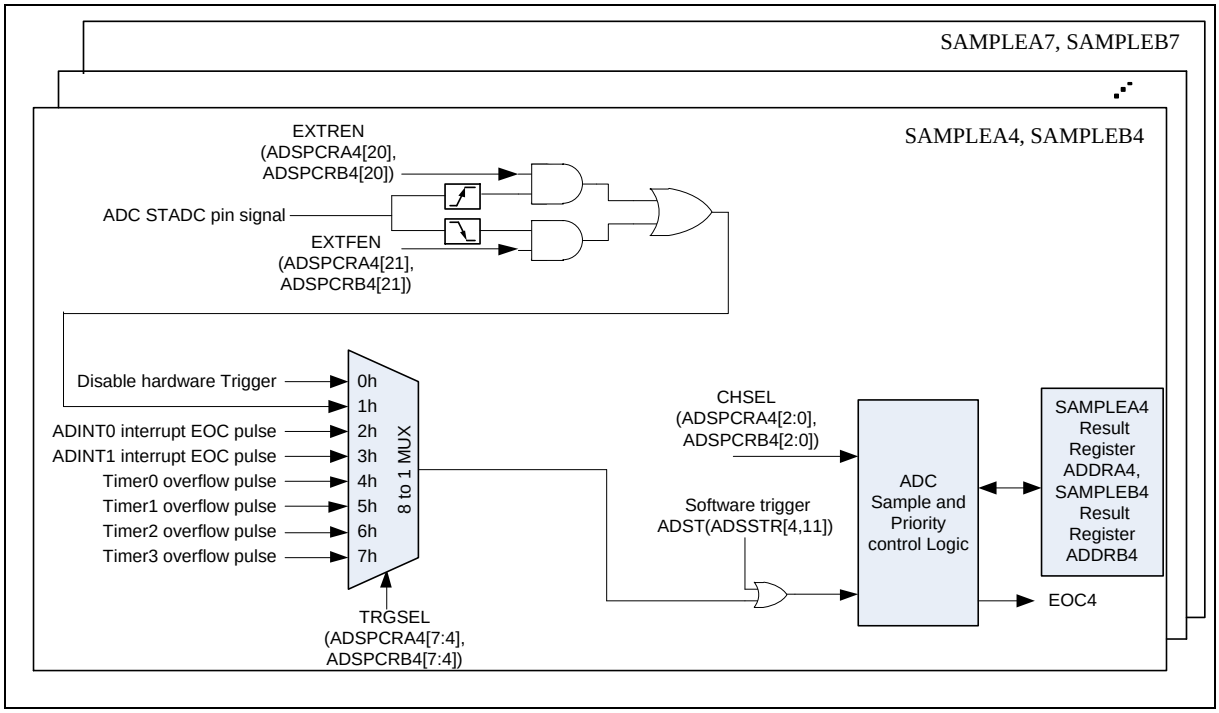


Figure 6–164 SAMPLEA4~7 and SAMPLEB4~7 Control Block Diagram

6.19.4.3 ADC Simultaneous Sampling Mode

The NuMicro®NM1530 has two ADCs that allow two different ADC channels to be simultaneously sampled. The priority rules are same as the single sampling mode.

The same numbered of SAMPLEA and the SAMPLEB are coupled together. For example, SAMPLEA0 and SAMPLEB0 are coupled when simultaneous sampling mode enable bit SIMUSEL0(ADSMSELR[0] bit = 1.

The ADC simultaneous sampling mode conversion operations are described below:

1. Only SAMPLEA trigger can start a pair of conversions.
2. SAMPLEA assign an A-channel and SAMPLEB also assign a B-channel in simultaneous sampling mode. The SAMPLEB specified trigger will be ignored.
3. The A-channel conversion result is stored in specific SAMPLEA ADDRA register, and the conversion result of the B-channel is placed in the same numbered SAMPLEB ADDRB register.

For example:

If SIMUSEL2(ADSMSELR[2]) bit = 1 and SAMPLEA2 is configured to sample EADC0_CH4 (CHSEL(ADSPCRA2[2:0]) = 4), it defines the SAMPLEA2 and same numbered SAMPLEB2 are coupled at simultaneous sampling conversion mode, if SAMPLEB2 is configured to sample EADC1_CH3 (CHSEL(ADSPCRB2[2:0]) = 3), the pair of ADC conversion channels are EADC0_CH4 and EADC1_CH3.

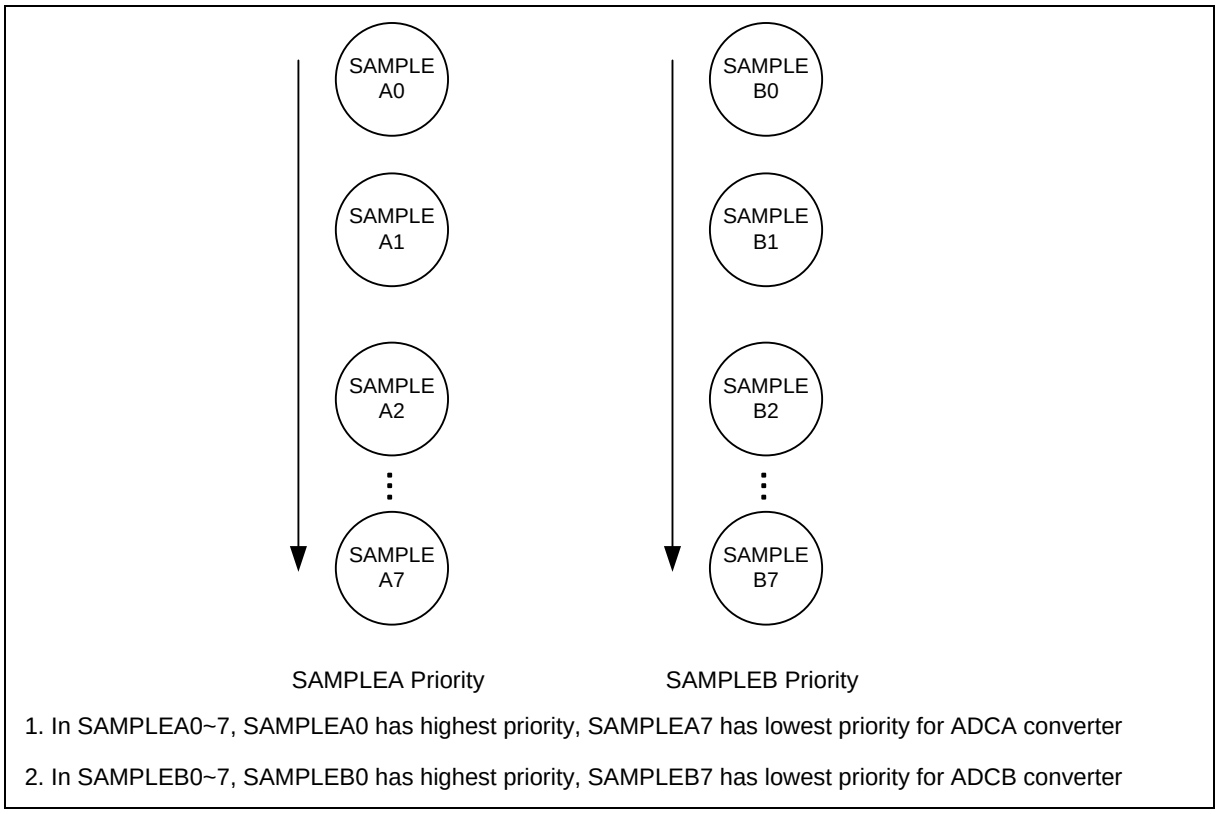
After a channel pair (AINA[4], AINB[3]) of ADC conversion completes, the results of those two ADC conversions will be placed in registers ADDRA2 and ADDRB2.

6.19.4.4 ADC Conversion Priority

There are two priority groups of converter for determining the conversion order when multiple SAMPLE trigger flags are set at the same time. SAMPLEA0~7 priority group is for ADCA converter, and SAMPLEB0~7 priority group is for ADCB converter.

The SAMPLE with lower number has higher priority than the higher number SAMPLE, if two SAMPLEs are triggered at the same time; the SAMPLE with lower number will start to convert ADC

first.



1. In SAMPLEA0~7, SAMPLEA0 has highest priority, SAMPLEA7 has lowest priority for ADCA converter
2. In SAMPLEB0~7, SAMPLEB0 has highest priority, SAMPLEB7 has lowest priority for ADCB converter

Figure 6–165 SAMPLE Conversion Priority Arbitrator Diagram

6.19.4.5 ADC Start Synchronous with PWM

Besides starting ADC conversionby software, ADINT0, 1 interrupt pulse, external pin STADC, and Timer0~3 overflow pulse, this device has a new feature to allow PWM channels to trigger the ADC start. User may configure the PWM trigger types: rising, falling PWM edge, period point of PWM or center point of PWM (Center-aligned mode only) to trigger ADC start. The device also allows user to configure the amount of delay prior to ADC start after hardware detected the PWM edge. User can configure the trigger delay time by settingTRGDLYCNT and TRGDLYDIV bits in ADSPCRAn and ADSPCRBn(n=0~3) registers. The figure below shows the programmable delay time for PWM-triggered ADC to start conversion.

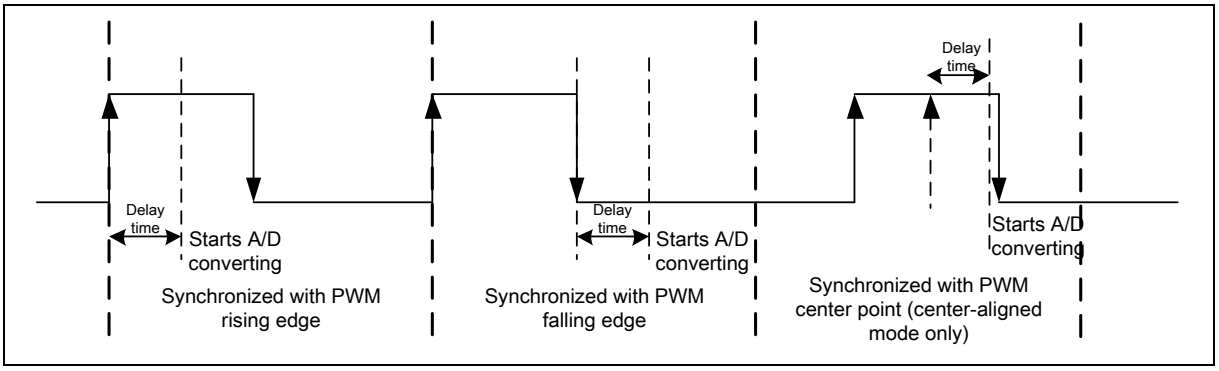


Figure 6–166 PWM-triggered ADC Start

6.19.4.6 ADC SAMPLE End of Conversion Interrupt Operation

There are 4 ADC interrupts ADINT0~3, and each of these interrupts has its own interrupt vector address and can be configured to select a specific SAMPLE EOC pulse (SAMPLEA0~7, SAMPLEB0~7 End of conversion pulses) as its interrupt trigger source. Each of them has its own interrupt overwritten flag ADFOVn(ADIFOVR[n], n=0~3). This flag set to 1 if the corresponding interrupt is overwritten to 1.

When ADIE0(ADCR[2]) = 1 and ADINT0SRCTL[7:0]=0xFF, all of SAMPLEA EOC pulses can cause an ADINT0 interrupt.

If ADIE1(ADCR[3]) = 1 and ADINT1SRCTL[15:8]=0xFF, all of SAMPLEB EOC pulses also can cause an ADINT1 interrupt.

The ADINT0, ADINT1 interrupt pulses are generated whenever the specific SAMPLE A/D EOC (End of conversion) pulse is generated. It also can be the SAMPLE conversion trigger sources, and user can use it to do the ADC continuous scan conversion.

Example for “Continuous scan”:

Step1. If ADC SAMPLEA2’s EOCA2 pulse is selected as ADINT0 interrupt trigger (ADINT0SRCTL[7:0] = 0x04) and ADINT0 is selected as SAMPLEA0, B1, A2 hardware conversion trigger.

Step2. Set software trigger ADST(ADSSTR[2]) bit to 1 to start a SAMPLEA2 ADC conversion, after the conversion completes, it generates an EOCA2 pulse signal and ADINT0 interrupt pulse at end of SAMPLEA2 ADC conversion, ADINT0 interrupt pulse will trigger the SAMPLEA0, B1, A2 to start the ADC conversions.

Step3. ADINT0 interrupt pulse repeats to trigger SAMPLEA0, B1, A2 ADC conversions automatically.

Step4. Clear TRGSEL(ADSPCRA2[7:4]) to 0 to disable SAMPLEA2 module’s ADINT0 interrupt pulse hardware trigger, if needs to stop the continuous scan.

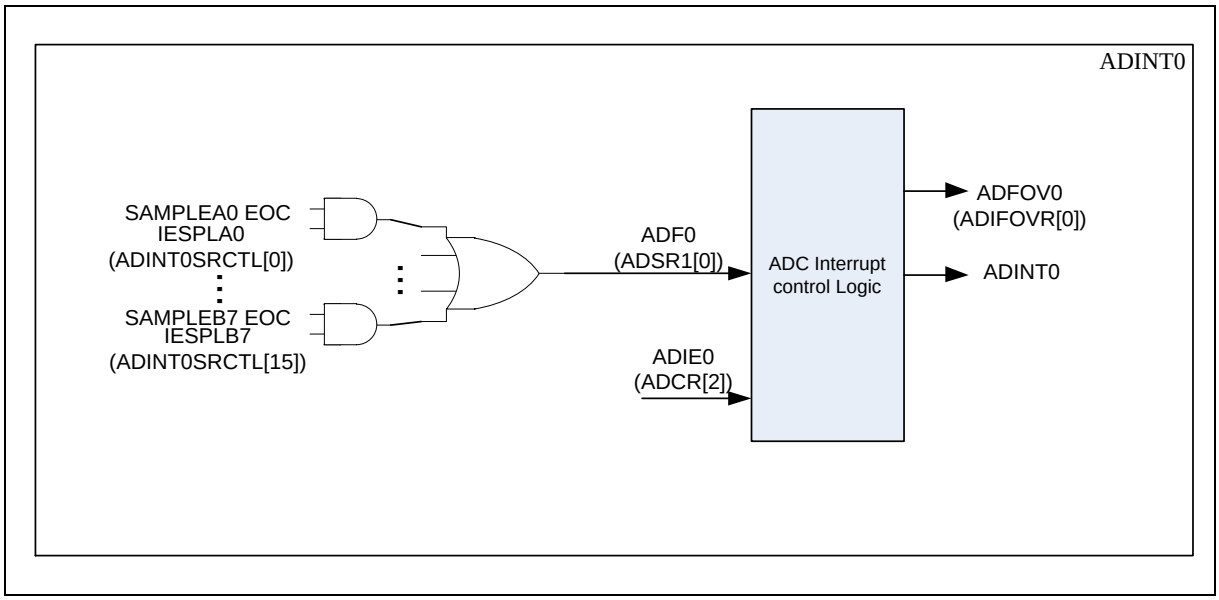


Figure 6–167 Specific SAMPLE A/D EOC Signal for ADINT0 Interrupt

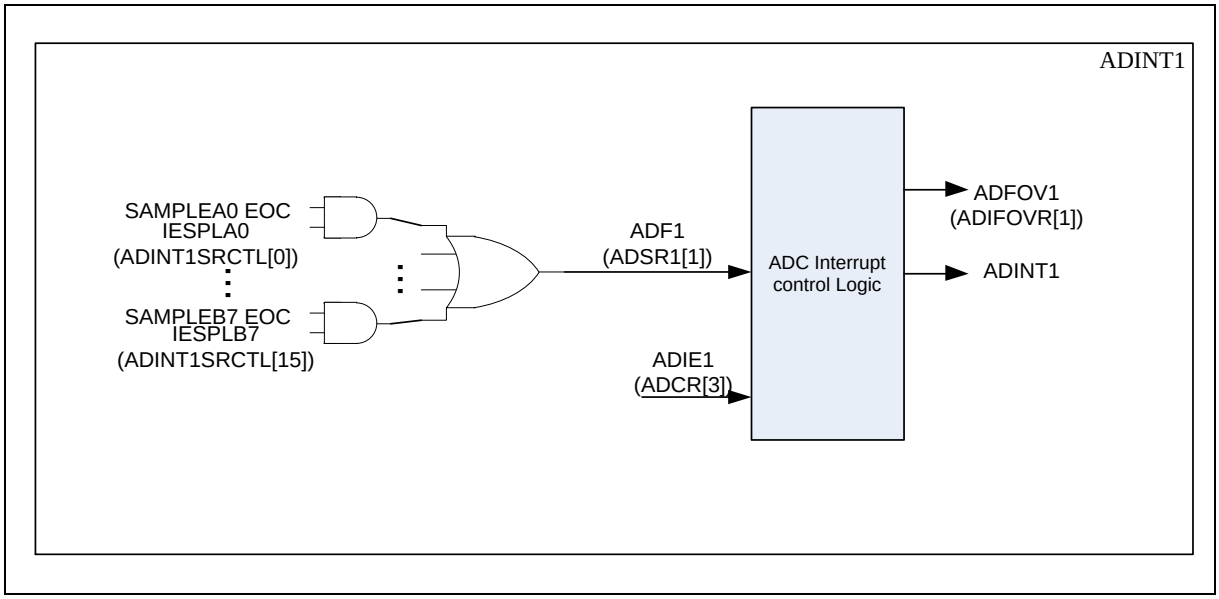


Figure 6–168 Specific SAMPLE A/D EOC Signal for ADINT1 Interrupt

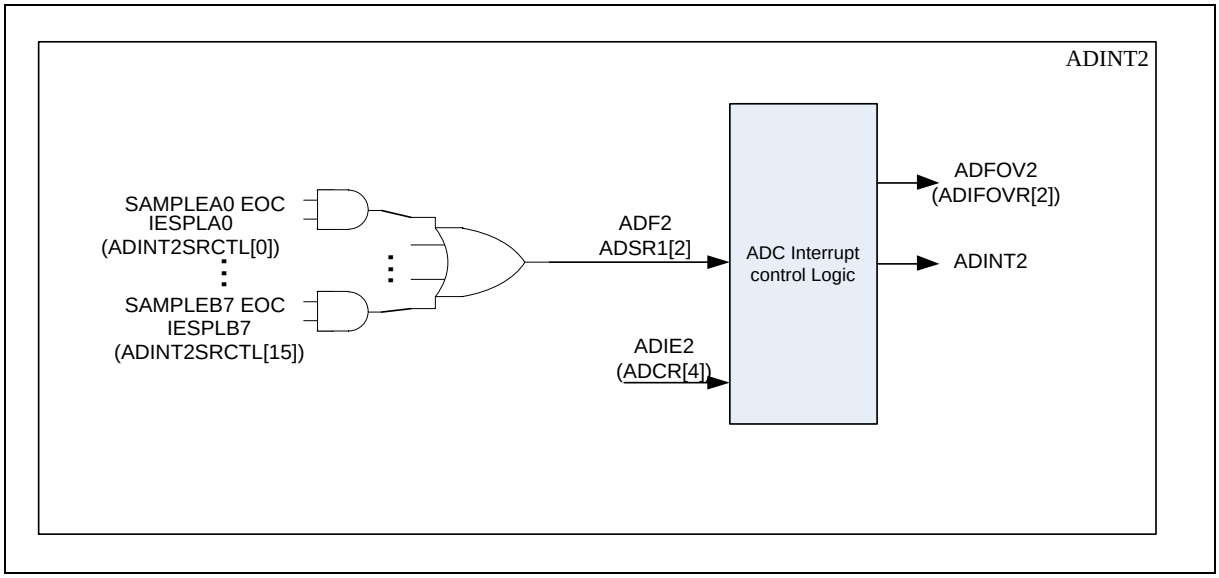


Figure 6–169 Specific SAMPLE A/D EOC Signal for ADINT2 Interrupt

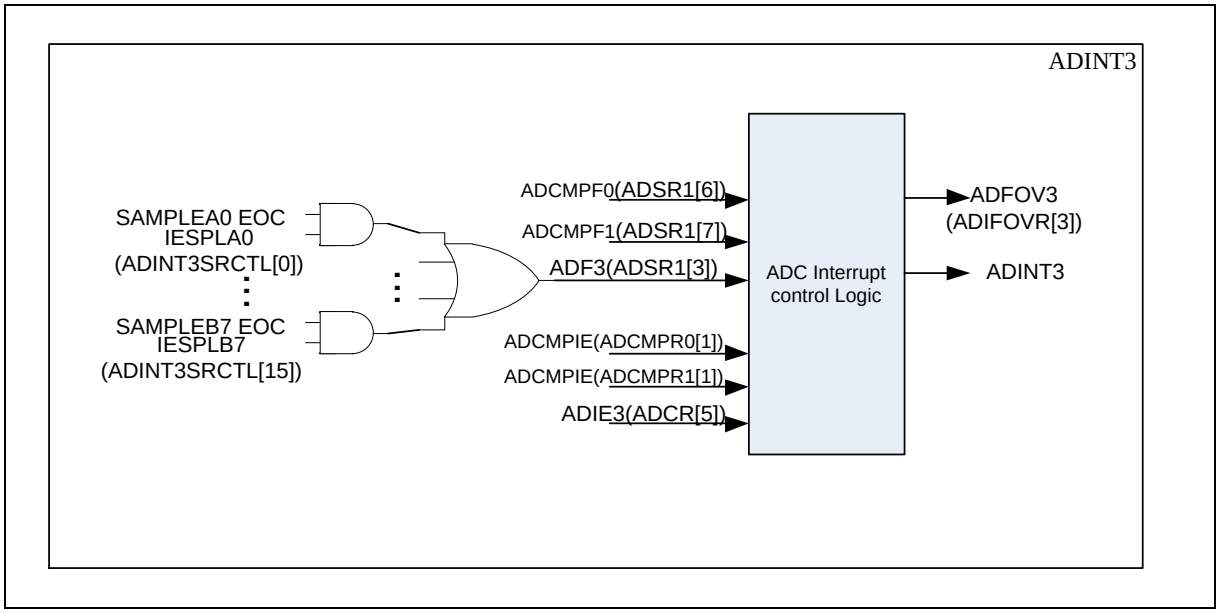


Figure 6–170 Specific SAMPLE A/D EOC Signal for ADINT3 Interrupt

6.19.4.7 Input Sampling and A/D Conversion Time

The A/D converter samples the analog input when A/D conversion start delay time (Td) has passed after ADST bit in ADSSTR is set to 1, and then starts conversion. Since the ADC clock is generated by HCLK divided by (EADC_N(CLKDIV[23:16])+1), the maximum delay time from user writeADST to A/D start sampling analog input time is two ADC clock cycles. The start delay time is shown below.

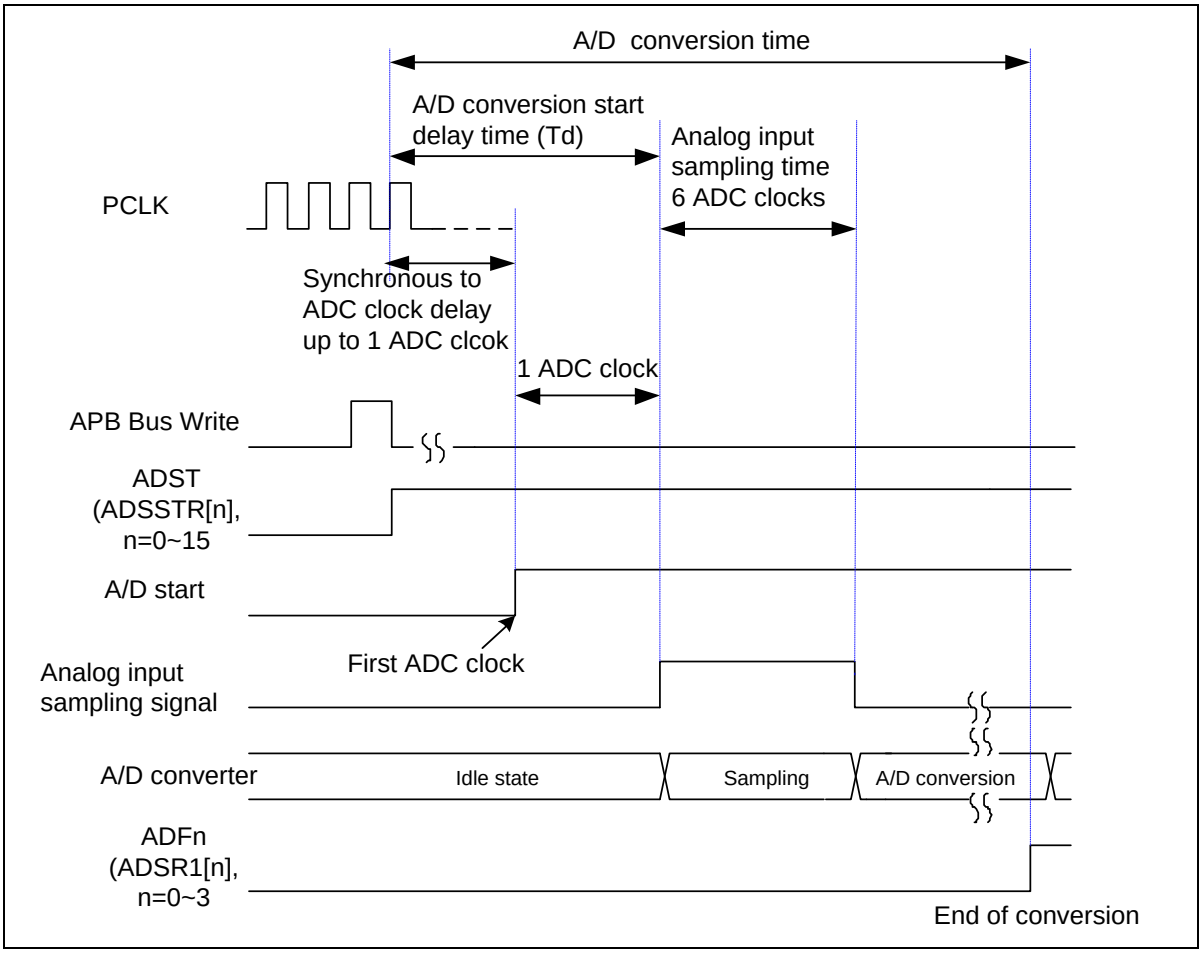


Figure 6–171 Conversion Start Delay Timing Diagram

A/D conversion can be triggered by external pin STADC request. Setting the TRGSEL(ADSPCRAm[7:4],ADSPCRBm[7:4],ADSPCRAn[6:4],ADSPCRBn[6:4], m=0~3, n=4~7) bits to 1 select external trigger input from the STADC pin. An 8-bit sampling counter is used to deglitch. If edge trigger condition is selected, the high and low state must be kept at least 4 HCLKs. The pulse that is shorter than this specification will be ignored.

6.19.4.8 A/D Extend Sampling Time

When A/D operation at high ADC clock rate, the sampling time of analog input voltage may not enough if the analog channel has heavy loading to cause fully charge time is longer. User can set A/D extend sampling time by writing ADAEST or ADBEST field in ADTCR register. The A/D extend sampling time is present between A/D controller judge which channel to be converting and A/D start to conversion. The range of extend sampling time is from 0 ~255 ADC clock. The extend sampling time is shown below.

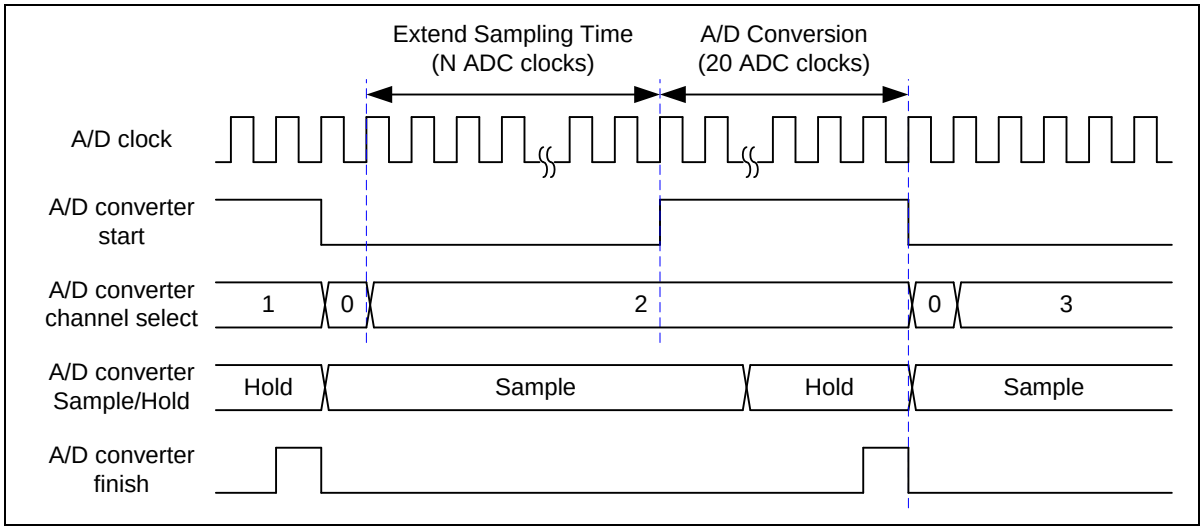


Figure 6–172 A/D Extend Sampling Timing Diagram

6.19.4.9 Conversion Result Monitored by Compare Mode

The NuMicro®NM1530controller provides two sets of compare register ADCMPR0/1 to monitor maximum two specified SAMPLEA0~3, SAMPLEB0~3 conversion results from A/D conversion module, refer to Figure below. Software can select which SAMPLE module result to be monitored by setting the CMPSMPL and CMPCOND bit in ADCMPR0/1 register used to check if the conversion result is less than the specified value or greater than (equal to) value specified in CMPD(ADCMPRn[27:16], n=0,1). When the conversion of the SAMPLE specified by CMPSMPL is completed, the comparing action will be triggered once automatically. When the compare result meets the setting, compare match counter will increase 1, when counter value reach the setting of (CMPMATCNT(ADCMPRn[11:8]+1n n=0, 1) then ADCMPFn(ADSR1[7:6], n=0, 1) bit will be set to 1, if ADCMPIE(ADCMPRn[1], n=0, 1) bit is set, an ADINT3 interrupt request is generated. User can use it to monitor the external analog input pin voltage transition. Detailed logics diagram is shown below.

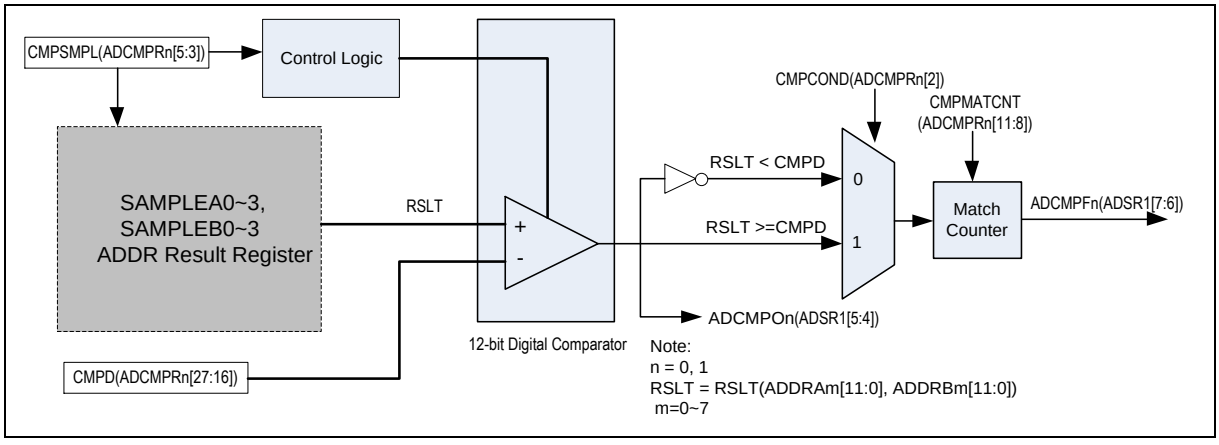


Figure 6–173 A/D Conversion Result Monitor Logics Diagram

6.19.4.10 Interrupt Sources

The A/D converter generates a conversion and ADFn(n=0~3) flag in the ADSR1 register upon the end of specific SAMPLE A/D conversion. If the ADIEn(n=0~3) bit in ADCR register is set, the conversion end interrupt request ADINTn(n=0~3) is generated.

If ADCMPIE(ADCMPRn[1], n=0, 1) bit is enabled, when A/D conversion result meets setting in ADCMPRn(n=0, 1) register, monitor interrupt is generated, ADINT3 will be set also. User can clear

ADCMPFn(ADSR1[7:6], n=0,1) and ADFn(ADSR1[3:0], n=0~3) flag to stop interrupt request.

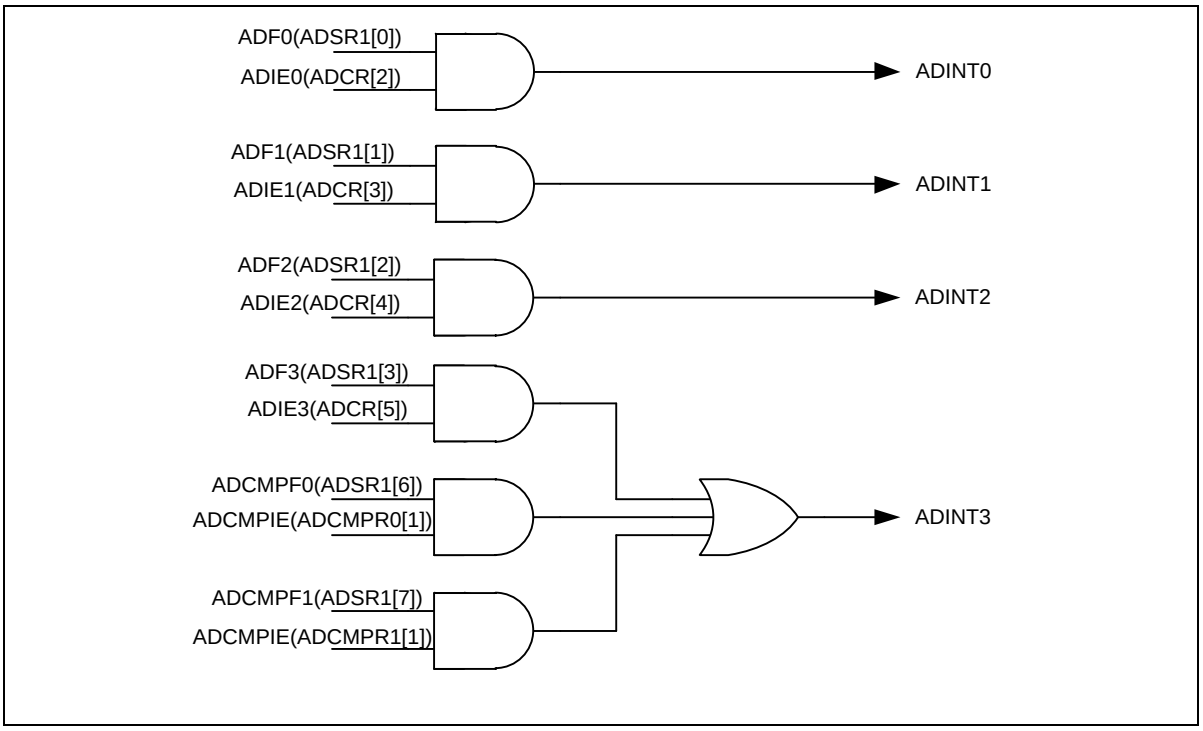


Figure 6–174 A/D Controller Interrupts

6.19.1 Register Map

R: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 0 can be written

Register	Offset	R/W	Description	Reset Value
EADC Base Address: **EADC_BA = 0x400E_0000**				
ADDRA0	EADC_BA+0x00	R	A/D Data Register 0 for SAMPLEA0	0x0000_0000
ADDRA1	EADC_BA+0x04	R	A/D Data Register 1 for SAMPLEA1	0x0000_0000
ADDRA2	EADC_BA+0x08	R	A/D Data Register 2 for SAMPLEA2	0x0000_0000
ADDRA3	EADC_BA+0x0C	R	A/D Data Register 3 for SAMPLEA3	0x0000_0000
ADDRA4	EADC_BA+0x10	R	A/D Data Register 4 for SAMPLEA4	0x0000_0000
ADDRA5	EADC_BA+0x14	R	A/D Data Register 5 for SAMPLEA5	0x0000_0000
ADDRA6	EADC_BA+0x18	R	A/D Data Register 6 for SAMPLEA6	0x0000_0000
ADDRA7	EADC_BA+0x1C	R	A/D Data Register 7 for SAMPLEA7	0x0000_0000
ADDRB0	EADC_BA+0x20	R	A/D Data Register 8 for SAMPLEB0	0x0000_0000
ADDRB1	EADC_BA+0x24	R	A/D Data Register 9 for SAMPLEB1	0x0000_0000
ADDRB2	EADC_BA+0x28	R	A/D Data Register 10 for SAMPLEB2	0x0000_0000
ADDRB3	EADC_BA+0x2C	R	A/D Data Register 11 for SAMPLEB3	0x0000_0000
ADDRB4	EADC_BA+0x30	R	A/D Data Register 12 for SAMPLEB4	0x0000_0000
ADDRB5	EADC_BA+0x34	R	A/D Data Register 13 for SAMPLEB5	0x0000_0000
ADDRB6	EADC_BA+0x38	R	A/D Data Register 14 for SAMPLEB6	0x0000_0000
ADDRB7	EADC_BA+0x3C	R	A/D Data Register 15 for SAMPLEB7	0x0000_0000
ADCR	EADC_BA+0x40	R/W	A/D Control Register	0x0000_0000
ADCHISELR	EADC_BA+0x44	R/W	A/D Channel Input Sources Select Register	0x0000_0000
ADSSTR	EADC_BA+0x48	W	A/D SAMPLE Software Start Register	0x0000_0000
ADSTPFR	EADC_BA+0x4C	R	A/D SAMPLE Start of Conversion Pending Flag Register	0x0000_0000
ADIFOVR	EADC_BA+0x50	R/W	A/D ADINT3~0 Interrupt Flag Over Run Register	0x0000_0000
ADSPOVFR	EADC_BA+0x54	R/W	A/D SAMPLE Start of Conversion Over Run Flag Register	0x0000_0000
ADSPCRA0	EADC_BA+0x58	R/W	A/D SAMPLEA0 Control Register	0x0000_0000
ADSPCRA1	EADC_BA+0x5C	R/W	A/D SAMPLEA1 Control Register	0x0000_0000
ADSPCRA2	EADC_BA+0x60	R/W	A/D SAMPLEA2 Control Register	0x0000_0000

Register	Offset	R/W	Description	Reset Value
EADC Base Address: **EADC_BA = 0x400E_0000**				
ADSPCRA3	EADC_BA+0x64	R/W	A/D SAMPLEA3 Control Register	0x0000_0000
ADSPCRA4	EADC_BA+0x68	R/W	A/D SAMPLEA4 Control Register	0x0000_0000
ADSPCRA5	EADC_BA+0x6C	R/W	A/D SAMPLEA5 Control Register	0x0000_0000
ADSPCRA6	EADC_BA+0x70	R/W	A/D SAMPLEA6 Control Register	0x0000_0000
ADSPCRA7	EADC_BA+0x74	R/W	A/D SAMPLEA7 Control Register	0x0000_0000
ADSPCRB0	EADC_BA+0x78	R/W	A/D SAMPLEB0 Control Register	0x0000_0000
ADSPCRB1	EADC_BA+0x7C	R/W	A/D SAMPLEB1 Control Register	0x0000_0000
ADSPCRB2	EADC_BA+0x80	R/W	A/D SAMPLEB2 Control Register	0x0000_0000
ADSPCRB3	EADC_BA+0x84	R/W	A/D SAMPLEB3 Control Register	0x0000_0000
ADSPCRB4	EADC_BA+0x88	R/W	A/D SAMPLEB4 Control Register	0x0000_0000
ADSPCRB5	EADC_BA+0x8C	R/W	A/D SAMPLEB5 Control Register	0x0000_0000
ADSPCRB6	EADC_BA+0x90	R/W	A/D SAMPLEB6 Control Register	0x0000_0000
ADSPCRB7	EADC_BA+0x94	R/W	A/D SAMPLEB7 Control Register	0x0000_0000
ADSMSELR	EADC_BA+0xA4	R/W	A/D SAMPLE Simultaneous Mode Select Register	0x0000_0000
ADCMPR0	EADC_BA+0xA8	R/W	A/D Result Compare Register 0	0x0000_0000
ADCMPR1	EADC_BA+0xAC	R/W	A/D Result Compare Register 1	0x0000_0000
ADSR0	EADC_BA+0xB0	R	A/D Status Register 0	0x0000_0000
ADSR1	EADC_BA+0xB4	R/W	A/D Status Register 1	0x0000_0000
ADTCR	EADC_BA+0xB8	R/W	A/D Timing Control Register	0x0000_0000
ADDRDBA0	EADC_BA+0x100	R	A/D Data Register double buffer for SAMPLEA0	0x0000_0000
ADDRDBA1	EADC_BA+0x104	R	A/D Data Register double buffer for SAMPLEA1	0x0000_0000
ADDRDBA2	EADC_BA+0x108	R	A/D Data Register double buffer for SAMPLEA2	0x0000_0000
ADDRDBA3	EADC_BA+0x10C	R	A/D Data Register double buffer for SAMPLEA3	0x0000_0000
ADDRDBB0	EADC_BA+0x120	R	A/D Data Register double buffer for SAMPLEB0	0x0000_0000
ADDRDBB1	EADC_BA+0x124	R	A/D Data Register double buffer for SAMPLEB1	0x0000_0000
ADDRDBB2	EADC_BA+0x128	R	A/D Data Register double buffer for SAMPLEB2	0x0000_0000
ADDRDBB3	EADC_BA+0x12C	R	A/D Data Register double buffer for SAMPLEB3	0x0000_0000
ADDBM	EADC_BA+0x130	R/W	A/D Double Buffer Mode select	0x0000_0000

Register	Offset	R/W	Description	Reset Value
EADC Base Address: **EADC_BA = 0x400E_0000**				
ADINT0SRCTL	EADC_BA+0x134	R/W	A/D interrupt 0 Source Enable Control Register.	0x0000_0000
ADINT1SRCTL	EADC_BA+0x138	R/W	A/D interrupt 1 Source Enable Control Register.	0x0000_0000
ADINT2SRCTL	EADC_BA+0x13C	R/W	A/D interrupt 2 Source Enable Control Register.	0x0000_0000
ADINT3SRCTL	EADC_BA+0x140	R/W	A/D interrupt 3 Source Enable Control Register.	0x0000_0000
SMPTRGA0	EADC_BA+0x144	R/W	A/D trigger condition for SAMPLEA0	0x0000_0000
SMPTRGA1	EADC_BA+0x148	R/W	A/D trigger condition for SAMPLEA1	0x0000_0000
SMPTRGA2	EADC_BA+0x14C	R/W	A/D trigger condition for SAMPLEA2	0x0000_0000
SMPTRGA3	EADC_BA+0x150	R/W	A/D trigger condition for SAMPLEA3	0x0000_0000
SMPTRGB0	EADC_BA+0x154	R/W	A/D trigger condition for SAMPLEB0	0x0000_0000
SMPTRGB1	EADC_BA+0x158	R/W	A/D trigger condition for SAMPLEB1	0x0000_0000
SMPTRGB2	EADC_BA+0x15C	R/W	A/D trigger condition for SAMPLEB2	0x0000_0000
SMPTRGB3	EADC_BA+0x160	R/W	A/D trigger condition for SAMPLEB3	0x0000_0000

6.19.2 Register Description

A/D Data Registers (ADDRA0~7, ADDRB0~7)

Register	Offset	R/W	Description	Reset Value
ADDRA0	EADC_BA+0x00	R	A/D Data Register 0 for SAMPLEA0	0x0000_0000
ADDRA1	EADC_BA+0x04	R	A/D Data Register 1 for SAMPLEA1	0x0000_0000
ADDRA2	EADC_BA+0x08	R	A/D Data Register 2 for SAMPLEA2	0x0000_0000
ADDRA3	EADC_BA+0x0C	R	A/D Data Register 3 for SAMPLEA3	0x0000_0000
ADDRA4	EADC_BA+0x10	R	A/D Data Register 4 for SAMPLEA4	0x0000_0000
ADDRA5	EADC_BA+0x14	R	A/D Data Register 5 for SAMPLEA5	0x0000_0000
ADDRA6	EADC_BA+0x18	R	A/D Data Register 6 for SAMPLEA6	0x0000_0000
ADDRA7	EADC_BA+0x1C	R	A/D Data Register 7 for SAMPLEA7	0x0000_0000
ADDRB0	EADC_BA+0x20	R	A/D Data Register 8 for SAMPLEB0	0x0000_0000
ADDRB1	EADC_BA+0x24	R	A/D Data Register 9 for SAMPLEB1	0x0000_0000
ADDRB2	EADC_BA+0x28	R	A/D Data Register 10 for SAMPLEB2	0x0000_0000
ADDRB3	EADC_BA+0x2C	R	A/D Data Register 11 for SAMPLEB3	0x0000_0000
ADDRB4	EADC_BA+0x30	R	A/D Data Register 12 for SAMPLEB4	0x0000_0000
ADDRB5	EADC_BA+0x34	R	A/D Data Register 13 for SAMPLEB5	0x0000_0000
ADDRB6	EADC_BA+0x38	R	A/D Data Register 14 for SAMPLEB6	0x0000_0000
ADDRB7	EADC_BA+0x3C	R	A/D Data Register 15 for SAMPLEB7	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**
Reserved				RSLT			
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
RSLT							

Bits	Description	
[31:18]	**Reserved**	Reserved.
[17]	**VALID**	**Valid Flag** 0 = Data in RSLT[11:0] bits is not valid. 1 = Data in RSLT[11:0] bits is valid. This bit is set to 1 when corresponding SAMPLE channel analog input conversion is completed and cleared by hardware after ADDR register is read.
[16]	**OVERRUN**	**over Run Flag** 0 = Data in RSLT[11:0] is the recent conversion result. 1 = Data in RSLT[11:0] is overwritten. If converted data in RSLT[11:0] has not been read before new conversion result is loaded to this register, OVERRUN is set to 1. It is cleared by hardware after ADDR register is read.
[15:12]	**Reserved**	Reserved.
[11:0]	**RSLT**	**A/D Conversion Result** This field contains 12-bit conversion result.

A/D Control Register (ADCR)

Register	Offset	R/W	Description	Reset Value
ADCR	EADC_BA+0x40	R/W	A/D 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		ADIE3	ADIE2	ADIE1	ADIE0	ADRESET	AD_EN

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**ADIE3**	**Specific SAMPLE A/D ADINT3 Interrupt Enable Bit** 0 = Specific SAMPLE A/D ADINT3 interrupt function Disabled. 1 = Specific SAMPLE A/D ADINT3 interrupt function Enabled. The A/D converter generates a conversion end ADF3 flag in ADSR1 register upon the end of specific SAMPLE A/D conversion. If ADIE3 bit is set then conversion end interrupt request ADINT3 is generated.
[4]	**ADIE2**	**Specific SAMPLE A/D ADINT2 Interrupt Enable Bit** 0 = Specific SAMPLE A/D ADINT2 interrupt function Disabled. 1 = Specific SAMPLE A/D ADINT2 interrupt function Enabled. The A/D converter generates a conversion end ADF2 flag in ADSR1 register upon the end of specific SAMPLE A/D conversion. If ADIE2 bit is set then conversion end interrupt request ADINT2 is generated.
[3]	**ADIE1**	**Specific SAMPLE A/D ADINT1 Interrupt Enable Bit** 0 = Specific SAMPLE A/D ADINT1 interrupt function Disabled. 1 = Specific SAMPLE A/D ADINT1 interrupt function Enabled. The A/D converter generates a conversion end ADF1 flag in ADSR1 register upon the end of specific SAMPLE A/D conversion. If ADIE1 bit is set then conversion end interrupt request ADINT1 is generated.
[2]	**ADIE0**	**Specific SAMPLE A/D ADINT0 Interrupt Enable Bit** 0 = Specific SAMPLE A/D ADINT0 interrupt function Disabled. 1 = Specific SAMPLE A/D ADINT0 interrupt function Enabled. The A/D converter generates a conversion end ADF0 flag in ADSR1 register upon the end of specific SAMPLE A/D conversion. If ADIE0 bit is set then conversion end interrupt request ADINT0 is generated.

Bits	Description	
[1]	**ADRESET**	**A/D Converter Control Circuits Reset** 0 = Writing 0 has no effect. 1 = Writing 1 will cause ADC control circuits reset to initial state, but not change the ADC registers value. ADRESET bit remains 1 during ADC reset, when ADC reset end, the ADRESET bit is automatically cleared to 0.
[0]	**AD_EN**	**A/D Converter Enable Bit** 0 = A/D converter Disabled. 1 = A/D converter Enabled. Before starting the A/D conversion function, this bit should be set to 1. Clear it to 0 to disable A/D converter analog circuit power consumption.

A/D Channel Input Select Register (ADCHISELR)

Register	Offset	R/W	Description	Reset Value
ADCHISELR	EADC_BA+0x44	R/W	A/D Channel Input Sources 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				PRESEL		AINB0SEL	AINA0SEL

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3:2]	**PRESEL**	**A/D Channel AINA[7] Analog Input Selection** 00 = Analog Input Channel AINA7. 01 = Band-gap (VBG) Analog Input. 10 = VTEMP Internal Temperature Sensor Analog Input. 11 = Analog ground.
[1]	**AINB0SEL**	**A/D Channel AINB[0] Analog Input Selection** 0 = AINB[0] pin P7.0E/EADC1_CH0 is selected as the A/D AINB[0] input signal. 1 = OP Amplifier1 output is selected as the A/D AINB[0] input signal.
[0]	**AINA0SEL**	**A/D Channel AINA[0] Analog Input Selection** 0 = AINA[0] pin P6.0/EADC0_CH0is selected as the ADC AINA[0] input signal. 1 = OP Amplifier0 output is selected as the ADC AINA[0] input signal.

A/D SAMPLE Software Start Register (ADSSTR)

Register	Offset	R/W	Description	Reset Value
ADSSTR	EADC_BA+0x48	W	A/D SAMPLE Software Start Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**ADST**	**A/D SAMPLEB7~0 Software Force to Start ADC Conversion Register** 0 = No effect. 1 = Cause an ADC conversion when the priority is given to SAMPLEB.
[7:0]	**ADST**	**A/D SAMPLEA7~0 Software Force to Start ADC Conversion Register** 0 = No effect. 1 = Cause an ADC conversion when the priority is given to SAMPLEA.

A/D SAMPLE Start of Conversion Pending Flag Register (ADSTPFR)

Register	Offset	R/W	Description	Reset Value
ADSTPFR	EADC_BA+0x4C	R	A/D SAMPLE Start of Conversion Pending Flag Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**STPF**	**A/D SAMPLEB7~0 Start Conversion Pending Flag** 0 = No pending conversion for SAMPLEB. 1 = SAMPLEBn ADC start of conversion is pending. This bit remains 1 during pending state, when the respective ADC conversion is end, the corresponding STPF bit is automatically cleared to 0. **Note:** n=0~7.
[7:0]	**STPF**	**A/D SAMPLEA7~0 Start Conversion Pending Flag** 0 = There is no pending conversion for SAMPLEA. 1 = SAMPLEAn ADC start of conversion is pending. This bit remains 1 during pending state, when the respective ADC conversion is end, the corresponding STPF bit is automatically cleared to 0. **Note:** n=0~7.

A/D Interrupt Flag Over Run Register (ADIFOVR)

Register	Offset	R/W	Description	Reset Value
ADIFOVR	EADC_BA+0x50	R/W	A/D ADINT3~0 Interrupt Flag Over Run 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				ADFOV3	ADFOV2	ADFOV1	ADFOV0

Bits	Description	
[31:4]	**Reserved**	Reserved.
[3]	**ADFOV3**	**A/D ADINT3 Interrupt Flag over Run Bit** 0 = ADINT3 interrupt flag is not over run. 1 = ADINT3 interrupt pulse received when ADF3 is 1. **Note:** This bit is cleared by writing 1 to 1.
[2]	**ADFOV2**	**A/D ADINT2 Interrupt Flag over Run Bit** 0 = ADINT2 interrupt flag is not over run. 1 = ADINT2 interrupt flag is overwrite to 1. **Note:** This bit is cleared by writing 1 to 1.
[1]	**ADFOV1**	**A/D ADINT1 Interrupt Flag over Run Bit** 0 = ADINT1 interrupt flag is not over run. 1 = ADINT1 interrupt flag is overwrite to 1. **Note:** This bit is cleared by writing 1 to 1.
[0]	**ADFOV0**	**A/D ADINT0 Interrupt Flag over Run Bit** 0 = ADINT0 interrupt flag is not over run. 1 = ADINT0 interrupt flag is overwrite to 1. **Note:** This bit is cleared by writing 1 to 1.

A/D SAMPLE Over Run Flag Register (ADSPOVFR)

Register	Offset	R/W	Description	Reset Value
ADSPOVFR	EADC_BA+0x54	R/W	A/D SAMPLE Start of Conversion Over Run Flag Register	0x0000_0000

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

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15:8]	**SPOVF**	**A/D SAMPLEB7~SAMPLEB0 Start Conversion Overrun Flag** 0 = No SAMPLE event overrun. 1 = Indicates new SAMPLEBn event is generated while an old one event is pending. If there is a new trigger event comes when the SAMPLE is pending for the last trigger event, the overrun is happened and the SPOVF bit will be set as 1. **Note1:** It is cleared by writing 1. **Note2:** n = 0~7.
[7:0]	**SPOVF**	**A/D SAMPLEA7~SAMPLEA0 Start Conversion Overrun Flag** 0 = No SAMPLE event overrun. 1 = Indicates new SAMPLEAn event is generated while an old one event is pending. If there is a new trigger event comes when the SAMPLE is pending for the last trigger event, the overrun is happened and the SPOVF bit will be set as 1. **Note1:** It is cleared by writing 1. **Note2:** n = 0~7.

A/D SAMPLEA0~3, SAMPLEB0~3 Control Registers (ADSPCRx0 ~ ADSPCRx3)

Register	Offset	R/W	Description	Reset Value
ADSPCRA0	EADC_BA+0x58	R/W	A/D SAMPLEA0 Control Register	0x0000_0000
ADSPCRA1	EADC_BA+0x5C	R/W	A/D SAMPLEA1 Control Register	0x0000_0000
ADSPCRA2	EADC_BA+0x60	R/W	A/D SAMPLEA2 Control Register	0x0000_0000
ADSPCRA3	EADC_BA+0x64	R/W	A/D SAMPLEA3 Control Register	0x0000_0000
ADSPCRB0	EADC_BA+0x78	R/W	A/D SAMPLEB0 Control Register	0x0000_0000
ADSPCRB1	EADC_BA+0x7C	R/W	A/D SAMPLEB1 Control Register	0x0000_0000
ADSPCRB2	EADC_BA+0x80	R/W	A/D SAMPLEB2 Control Register	0x0000_0000
ADSPCRB3	EADC_BA+0x84	R/W	A/D SAMPLEB3 Control Register	0x0000_0000

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

Bits	Description	
[31:20]	**Reserved**	Reserved.
[21]	**EXTFEN**	**A/D External Trigger Falling Edge Enabled** 0 = Falling edge Disabled when A/D selects STADC as trigger source. 1 = Falling edge Enabled when A/D selects STADC as trigger source.
[20]	**EXTREN**	**A/D External Trigger Rising Edge Enabled** 0 = Rising edge Disabled when A/D selects STADC as trigger source. 1 = Rising edge Enabled when A/D selects STADC as trigger source.
[19:18]	**Reserved**	Reserved.
[17:16]	**TRGDLYDIV**	**A/D SAMPLE Start Conversion Trigger Delay Clock Divider Selection** Trigger delay clock frequency: 00 = ADC_CLK/1. 01 = ADC_CLK/2. 10 = ADC_CLK/4. 11 = ADC_CLK/16.

Bits	Description	
[15:8]	**TRGDLYCNT**	**A/D SAMPLE Start Conversion Trigger Delay Time** Trigger delay time =TRGDLYCNT x ADC_CLK x n (n=1, 2, 4, 16 from TRGDLYDIV setting).
[7:4]	**TRGSEL**	**A/D SAMPLE Start Conversion Trigger Source Selection** 0000 = Disable hardware trigger. 0001 = External trigger from STADC pin input. 0010 = ADC ADINT0 interrupt EOC pulse trigger. 0011 = ADC ADINT1 interrupt EOC pulse trigger. 0100 = Timer0 overflow pulse trigger. 0101 = Timer1 overflow pulse trigger. 0110 = Timer2 overflow pulse trigger. 0111 = Timer3 overflow pulse trigger. 1000 = PWM00 trigger. 1001 = PWM02 trigger. 1010 = PWM04 trigger. 1011 = PWM10 trigger. 1100 = PWM12 trigger. 1101 = PWM14 trigger. 1110 = PWM20 trigger. 1111 = PWM21 trigger.
3	**Reserved**	Reserved.
[2:0]	**CHSEL**	**A/D SAMPLEA,B Channel Selection** 000 = AINx[0]. 001 = AINx[1]. 010 = AINx[2]. 011 = AINx[3]. 100 = AINx[4]. 101 = AINx[5]. 110 = AINx[6]. 111 = AINx[7].

A/D SAMPLEA4~7, SAMPLEB4~7 Control Registers (ADSPCRx4 ~ ADSPCRx7)

Register	Offset	R/W	Description	Reset Value
ADSPCRA4	EADC_BA+0x68	R/W	A/D SAMPLEA4 Control Register	0x0000_0000
ADSPCRA5	EADC_BA+0x6C	R/W	A/D SAMPLEA5 Control Register	0x0000_0000
ADSPCRA6	EADC_BA+0x70	R/W	A/D SAMPLEA6 Control Register	0x0000_0000
ADSPCRA7	EADC_BA+0x74	R/W	A/D SAMPLEA7 Control Register	0x0000_0000
ADSPCRB4	EADC_BA+0x88	R/W	A/D SAMPLEB4 Control Register	0x0000_0000
ADSPCRB5	EADC_BA+0x8C	R/W	A/D SAMPLEB5 Control Register	0x0000_0000
ADSPCRB6	EADC_BA+0x90	R/W	A/D SAMPLEB6 Control Register	0x0000_0000
ADSPCRB7	EADC_BA+0x94	R/W	A/D SAMPLEB7 Control Register	0x0000_0000

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

Bits	Description	
[31:22]	**Reserved**	Reserved.
[21]	**EXTFEN**	**A/D External Trigger Falling Edge Enabled** 0 = Falling edge Disabled when A/D selects STADC as trigger source. 1 = Falling edge Enabled when A/D selects STADC as trigger source.
[20]	**EXTREN**	**A/D External Trigger Rising Edge Enabled** 0 = Rising edge Disabled when A/D selects STADC as trigger source. 1 = Rising edge Enabled when A/D selects STADC as trigger source.
[19:7]	**Reserved**	Reserved.

Bits	Description	
[6:4]	**TRGSEL**	**A/D SAMPLE Start Conversion Trigger Source Selection** 000 = Disable hardware trigger. 001 = External trigger from STADC pin input. 010 = ADC ADINT0 interrupt EOC pulse trigger. 011 = ADC ADINT1 interrupt EOC pulse trigger. 100 = Timer0 overflow pulse trigger. 101 = Timer1 overflow pulse trigger. 110 = Timer2 overflow pulse trigger. 111 = Timer3 overflow pulse trigger.
[3]	**Reserved**	Reserved.
[2:0]	**CHSEL**	**A/D SAMPLEA,B Channel Selection** 000 = AINx[0]. 001 = AINx[1]. 010 = AINx[2]. 011 = AINx[3]. 100 = AINx[4]. 101 = AINx[5]. 110 = AINx[6]. 111 = AINx[7].

A/D Simultaneous Sampling Mode Select Register (ADSMSELR)

Register	Offset	R/W	Description	Reset Value
ADSMSELR	EADC_BA+0xA4	R/W	A/D SAMPLE Simultaneous Mode 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
SIMUSEL7	SIMUSEL6	SIMUSEL5	SIMUSEL4	SIMUSEL3	SIMUSEL2	SIMUSEL1	SIMUSEL0

Bits	Description	
[31:8]	**Reserved**	Reserved.
[7]	**SIMUSEL7**	**A/D SAMPLEA7, SAMPLEB7 Simultaneous Sampling Mode Selection** 0 = SAMPLEA7, SAMPLEB7 are in single sampling mode, both SAMPLEA7 and SAMPLEB7's 3 bits of CHSEL define the ADC channels to be converted. 1 = SAMPLEA7, SAMPLEB7 are in simultaneous sampling mode, Only SAMPLEA7 can trigger both the ADC conversions of SAMPLEA7 and SAMPLEB7, SAMPLEB7 trigger select TRGSEL is ignored. If SAMPLEA7's CHSEL = 1, SAMPLEB7's CHSEL = 3, the pair of channels are AINA[1], AINB[3], they will do the ADC conversion at the same time.
[6]	**SIMUSEL6**	**A/D SAMPLEA6, SAMPLEB6 Simultaneous Sampling Mode Selection** 0 = SAMPLEA6, SAMPLEB6 are in single sampling mode, both SAMPLEA6 and SAMPLEB6's 3 bits of CHSEL define the ADC channels to be converted. 1 = SAMPLEA6, SAMPLEB6 are in simultaneous sampling mode, Only SAMPLEA6 can trigger both the ADC conversions of SAMPLEA6 and SAMPLEB6, SAMPLEB6 trigger select TRGSEL is ignored. If SAMPLEA6's CHSEL = 1, and SAMPLEB6's CHSEL = 3, the pair of channels are AINA[1], AINB[3], they will do the ADC conversion at the same time.
[5]	**SIMUSEL5**	**A/D SAMPLEA5, SAMPLEB5 Simultaneous Sampling Mode Selection** 0 = SAMPLEA5, SAMPLEB5 are in single sampling mode, both SAMPLEA5 and SAMPLEB5's 3 bits of CHSEL define the ADC channels to be converted. 1 = SAMPLEA5, SAMPLEB5 are in simultaneous sampling mode, Only SAMPLEA5 can trigger both the ADC conversions of SAMPLEA5 and SAMPLEB5, SAMPLEB5 trigger select TRGSEL is ignored. If SAMPLEA5's CHSEL = 1, and SAMPLEB5's CHSEL = 3, the pair of channels are AINA[1], AINB[3], they will do the ADC conversion at the same time.
[4]	**SIMUSEL4**	**A/D SAMPLEA4, SAMPLEB4 Simultaneous Sampling Mode Select Ion** 0 = SAMPLEA4, SAMPLEB4 are in single sampling mode, both SAMPLEA4 and SAMPLEB4's 3 bits of CHSEL define the ADC channels to be converted. 1 = SAMPLEA4, SAMPLEB4 are in simultaneous sampling mode, Only SAMPLEA4 can trigger both the ADC conversions of SAMPLEA4 and SAMPLEB4, SAMPLEB4 trigger select TRGSEL is ignored. If SAMPLEA4's CHSEL = 1, and SAMPLEB4's CHSEL = 3, the pair of channels are AINA[1], AINB[3], they will do the ADC conversion at the same time.

Bits	Description	
[3]	**SIMUSEL3**	**A/D SAMPLEA3, SAMPLEB3 Simultaneous Sampling Mode Select Ion** 0 = SAMPLEA3, SAMPLEB3 are in single sampling mode, both SAMPLEA3 and SAMPLEB3's 3 bits of CHSEL define the ADC channels to be converted. 1 = SAMPLEA3, SAMPLEB3 are in simultaneous sampling mode, Only SAMPLEA3 can trigger both the ADC conversions of SAMPLEA3 and SAMPLEB3, SAMPLEB3 trigger select TRGSEL is ignored. If SAMPLEA3's CHSEL = 1, and SAMPLEB3's CHSEL = 3, the pair of channels are AINA[1], AINB[3], they will do the ADC conversion at the same time.
[2]	**SIMUSEL2**	**A/D SAMPLEA2, SAMPLEB2 Simultaneous Sampling Mode Selection** 0 = SAMPLEA2, SAMPLEB2 are in single sampling mode, both SAMPLEA2 and SAMPLEB2's 3 bits of CHSEL define the ADC channels to be converted. 1 = SAMPLEA2, SAMPLEB2 are in simultaneous sampling mode, Only SAMPLEA2 can trigger both the ADC conversions of SAMPLEA2 and SAMPLEB2, SAMPLEB2 trigger select TRGSEL is ignored. If SAMPLEA2's CHSEL = 1, and SAMPLEB2's CHSEL = 3, the pair of channels are AINA[1], AINB[3], they will do the ADC conversion at the same time.
[1]	**SIMUSEL1**	**A/D SAMPLEA1, SAMPLEB1 Simultaneous Sampling Mode Selection** 0 = SAMPLEA1, SAMPLEB1 are in single sampling mode, both SAMPLEA1 and SAMPLEB1's 3 bits of CHSEL define the ADC channels to be converted. 1 = SAMPLEA1, SAMPLEB1 are in simultaneous sampling mode, Only SAMPLEA1 can trigger both the ADC conversions of SAMPLEA1 and SAMPLEB1, SAMPLEB1 trigger select TRGSEL is ignored. If SAMPLEA1's CHSEL = 1, and SAMPLEB1's CHSEL = 3, the pair of channels are AINA[1], AINB[3], they will do the ADC conversion at the same time.
[0]	**SIMUSEL0**	**A/D SAMPLEA0, SAMPLEB0 Simultaneous Sampling Mode Selection** 0 = SAMPLEA0, SAMPLEB0 are in single sampling mode, both SAMPLEA0 and SAMPLEB0's 3 bits of CHSEL define the ADC channels to be converted. 1 = SAMPLEA0, SAMPLEB0 are in simultaneous sampling mode, Only SAMPLEA0 can trigger both the ADC conversions of SAMPLEA0 and SAMPLEB0, SAMPLEB0 trigger select TRGSEL is ignored. If SAMPLEA0's CHSEL = 1, and SAMPLEB0's CHSEL = 3, the pair of channels are AINA[1], AINB[3], they will do the ADC conversion at the same time.

A/D Result Compare Register 0/1 (ADCMPR0/1)

Register	Offset	R/W	Description	Reset Value
ADCMPR0	EADC_BA+0xA8	R/W	A/D Result Compare Register 0	0x0000_0000
ADCMPR1	EADC_BA+0xAC	R/W	A/D Result Compare Register 1	0x0000_0000

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

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27:16]	**CMPD**	**Comparison Data** The 12 bits data is used to compare with the conversion result of specified SAMPLE. Software can use it to monitor the external analog input pin voltage transition without imposing a load on software.
[15:12]	**Reserved**	Reserved.
[11:8]	**CMPMATCNT**	**Compare Match Count** When the specified A/D SAMPLE analog conversion result matches the compare condition defined by CMPCOND, the internal match counter will increase 1. When the internal counter reaches the value to (CMPMATCNT +1), the ADCMPFn(ADSR1[7:6], n=0, 1) bit will be set.
[7:6]	**Reserved**	Reserved.
[5:3]	**CMPSMPL**	**Compare SAMPLE Selection** 000 = SAMPLEA0 conversion result ADDRA0 is selected to be compared. 001 = SAMPLEA1 conversion result ADDRA1 is selected to be compared. 010 = SAMPLEA2 conversion result ADDRA2 is selected to be compared. 011 = SAMPLEA3 conversion result ADDRA3 is selected to be compared. 100 = SAMPLEB0 conversion result ADDRB0 is selected to be compared. 101 = SAMPLEB1 conversion result ADDRB1 is selected to be compared. 110 = SAMPLEB2 conversion result ADDRB2 is selected to be compared. 111 = SAMPLEB3 conversion result ADDRB3 is selected to be compared.

Bits	Description	
[2]	**CMPCOND**	**Compare Condition** 0= Set the compare condition as that when a 12-bit A/D conversion result is less than the 12-bit CMPD(ADCMPRn[27:16], n=0, 1), the internal match counter will increase one. 1= Set the compare condition as that when a 12-bit A/D conversion result is greater or equal to the 12-bit CMPD(ADCMPRn[27:16] , n=0, 1), the internal match counter will increase one. **Note:** When the internal counter reaches the value to (CMPMATCNT +1), the ADCMPFn(ADSR1[7:6], n=0, 1) bit will be set.
[1]	**ADCMPIE**	**A/D Result Compare Interrupt Enable Bit** 0 = Compare function interrupt Disabled. 1 = Compare function interrupt Enabled. If the compare function is enabled and the compare condition matches the setting of CMPCOND and CMPMATCNT, ADCMPFn(ADSR1[7:6], n=0, 1) bit will be asserted, in the meanwhile, if ADCMPIE is set to 1, a compare interrupt request is generated.
[0]	**ADCMP_EN**	**A/D Result Compare Enable Bit** 0 = Compare Disabled. 1 = Compare Enabled. Set this bit to 1 to enable compare CMPD with specified SAMPLE conversion result when converted data is loaded into ADDRAn and ADDRBn(n=0~3) register.

A/D Status Register 0 (ADSR0)

Register	Offset	R/W	Description	Reset Value
ADSR0	EADC_BA+0xB0	R	A/D Status Register 0	0x0000_0000

31	30	29	28	27	26	25	24
OVERRUN							
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
VALID							

Bits	Description	
[31:24]	OVERRUN	**ADDRB7~0 over Run Flag** It is a mirror to OVERRUN bit in SAMPLEB A/D result data register ADDRBn, n=0~7.
[23:16]	OVERRUN	**ADDRA7~0 over Run Flag** It is a mirror to OVERRUN bit in SAMPLEA A/D result data register ADDRAn, n=0~7.
[15:8]	VALID	**ADDRB7~0 Data Valid Flag** It is a mirror of VALID bit in SAMPLEB A/D result data register ADDRBn, n=0~7.
[7:0]	VALID	**ADDRA7~0 Data Valid Flag** It is a mirror of VALID bit in SAMPLEA A/D result data register ADDRAn, n=0~7.

A/D Status Register 1 (ADSR1)

Register	Offset	R/W	Description	Reset Value
ADSR1	EADC_BA+0xB4	R/W	A/D Status Register 1	0x0000_0000

31	30	29	28	27	26	25	24
Reserved				AOVERRUN	AVALID	ASPOVF	AADFOV
23	22	21	20	19	18	17	16
Reserved	CHANNELB			Reserved			BUSYB
15	14	13	12	11	10	9	8
Reserved	CHANNELA			Reserved			BUSYA
7	6	5	4	3	2	1	0
ADCMPF1	ADCMPF0	ADCMPO1	ADCMPO0	ADF3	ADF2	ADF1	ADF0

Bits	Description	
[31:28]	**Reserved**	Reserved.
[27]	**AOVERRUN**	**All SAMPLE A/D Result Data Register over Run Flags Check** 0 = None of SAMPLE data register over run flag OVERRUNn is set to 1. 1 = Any one of SAMPLE data register over run flag OVERRUNn is set to 1. This bit will keep 1 when any OVERRUNn Flag is equal to 1. OVERRUNn = OVERRUN(ADDRAn[16], ADDRBn[16], n=0~7).
[26]	**AVALID**	**All SAMPLE A/D Result Data Register ADDR Data Valid Flag Check** 0 = None of SAMPLE data register valid flag VALIDn is set to 1. 1 = Any one of SAMPLE data register valid flag VALIDn is set to 1. This bit will keep 1 when any VALIDn Flag is equal to 1. VALIDn = VALID(ADDRAn[17], ADDRBn[17], n=0~7).
[25]	**ASPOVF**	**All A/D SAMPLE Start Conversion over Run Flags Check** 0 = None of SAMPLE event over run flag SPOVFn is set to 1. 1 = Any one of SAMPLE event over run flag SPOVFn is set to 1. This bit will keep 1 when any SPOVFn Flag is equal to 1. SPOVFn = SPOVF(ADSPOVFR[n], n=0~15).
[24]	**AADFOV**	**All A/D Interrupt Flag over Run Bits Check** 0 = None of ADINT interrupt flag ADFOVn is overwritten to 1. 1 = Any one of ADINT interrupt flag ADFOVn is overwritten to 1. This bit will keep 1 when any ADFOVn Flag is equal to 1. ADFOVn = ADFOV(ADIFOVR[n], n=0~3).
[23]	**Reserved**	Reserved.

Bits	Description	
[22:20]	**CHANNELB**	**Current Conversion Channel (Read Only)** This filed reflects ADC1 current conversion channel when BUSYB=1. When BUSYB=0, it shows the last converted channel. 000 = AINB[0]. 001 = AINB[1]. 010 = AINB[2]. 011 = AINB[3]. 100 = AINB[4]. 101 = AINB[5]. 110 = AINB[6]. 111 = AINB[7].
[19:17]	**Reserved**	Reserved.
[16]	**BUSYB**	**BUSY/IDLE (Read Only)** 0 = A/D converter 1 (ADC1) is in idle state. 1 = A/D converter 1 (ADC1) is busy at conversion.
[14:12]	**CHANNELA**	**Current Conversion Channel (Read Only)** This filed reflects EADC0 current conversion channel when BUSYA=1. When BUSYA=0, it shows the last converted channel. 000 = AINA[0]. 001 = AINA[1]. 010 = AINA[2]. 011 = AINA[3]. 100 = AINA[4]. 101 = AINA[5]. 110 = AINA[6]. 111 = AINA[7].
[11:9]	**Reserved**	Reserved.
[8]	**BUSYA**	**BUSY/IDLE (Read Only)** 0 = A/D converter 0 (EADC0) is in idle state. 1 = A/D converter 0 (EADC0) is busy at conversion.
[7]	**ADCMPF1**	**ADC Compare 1 Flag** When the specific SAMPLE A/D conversion result meets setting condition in ADCMPR1 register then this bit is set to 1. And it is cleared by write 1. 0 = Conversion result in ADDR does not meet ADCMPR1 register setting. 1 = Conversion result in ADDR meets ADCMPR1 register setting.
[6]	**ADCMPF0**	**ADC Compare 0 Flag** When the specific SAMPLE A/D conversion result meets setting condition in ADCMPR0 then this bit is set to 1. And it is cleared by write 1. 0 = Conversion result in ADDR does not meet ADCMPR0 register setting. 1 = Conversion result in ADDR meets ADCMPR0 register setting.

Bits	Description	
[5]	**ADCMPO1**	**ADC Compare 1 Output Status Bit** The 12 bits compare1 data CMPD(ADCMPR1[27:16]) is used to compare with conversion result of specified SAMPLE. Software can use it to monitor the external analog input pin voltage status. 0 = Conversion result in ADDR less than CMPD(ADCMPR1[27:16]) setting. 1 = Conversion result in ADDR great than or equal CMPD(ADCMPR1[27:16]) setting.
[4]	**ADCMPO0**	**ADC Compare 0 Output Status Bit** The 12 bits compare0 data CMPD(ADCMPR0[27:16]) is used to compare with conversion result of specified SAMPLE. Software can use it to monitor the external analog input pin voltage status. 0 = Conversion result in ADDR is less than CMPD(ADCMPR0[27:16]) setting. 1 = Conversion result in ADDR is great than or equal CMPD(ADCMPR0[27:16]) setting.
[3]	**ADF3**	**A/D ADINT3 Interrupt Flag** 0 = No ADINT3 interrupt pulse received. 1 = ADINT3 interrupt pulse received. **Note1:** It is cleared by writing 1. **Note2:** This bit indicates whether an A/D conversion of specific SAMPLE has been completed
[2]	**ADF2**	**A/D ADINT2 Interrupt Flag** 0 = No ADINT2 interrupt pulse received. 1 = ADINT2 interrupt pulse received. **Note1:** It is cleared by writing 1. **Note2:** This bit indicates whether an A/D conversion of specific SAMPLE has been completed.
[1]	**ADF1**	**A/D ADINT1 Interrupt Flag** 0 = no ADINT1 interrupt pulse received. 1 = ADINT1 interrupt pulse has been received. **Note1:** It is cleared by writing 1. **Note2:** This bit indicates whether an A/D conversion of specific SAMPLE has been completed.
[0]	**ADF0**	**A/D ADINT0 Interrupt Flag** 0 = No ADINT0 interrupt pulse received. 1 = ADINT0 interrupt pulse received. **Note1:** It is cleared by writing 1. **Note2:** This bit indicates whether an A/D conversion of specific SAMPLE has been completed.

A/D Timing Control Register (ADTCR)

Register	Offset	R/W	Description	Reset Value
ADTCR	EADC_BA+0xB8	R/W	A/D Timing Control Register	0x0000_0000

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

Bits	Description	
[31:24]	**Reserved**	Reserved.
[23:16]	**ADBEST**	**ADC1 Extend Sampling Time** When A/D converting at high conversion rate, the sampling time of analog input voltage may not enough if input channel loading is heavy, User can extend A/D sampling time after trigger source is coming to get enough sampling time. The range of start delay time is from 0~255 ADC clock.
[15:8]	**Reserved**	Reserved.
[7:0]	**ADAEST**	**ADC0 Extend Sampling Time** When A/D converting at high conversion rate, the sampling time of analog input voltage may not enough if input channel loading is heavy, User can extend A/D sampling time after trigger source is coming to get enough sampling time. The range of start delay time is from 0~255 ADC clock.

A/D Data Registers double buffer for A/D Data Registers (ADDRDBA0~3, ADDRDBB0~3)

Register	Offset	R/W	Description	Reset Value
ADDRDBA0	EADC_BA+0x100	R	A/D Data Register double buffer for SAMPLEA0	0x0000_0000
ADDRDBA1	EADC_BA+0x104	R	A/D Data Register double buffer for SAMPLEA1	0x0000_0000
ADDRDBA2	EADC_BA+0x108	R	A/D Data Register double buffer for SAMPLEA2	0x0000_0000
ADDRDBA3	EADC_BA+0x10C	R	A/D Data Register double buffer for SAMPLEA3	0x0000_0000
ADDRDBB0	EADC_BA+0x120	R	A/D Data Register double buffer for SAMPLEB0	0x0000_0000
ADDRDBB1	EADC_BA+0x124	R	A/D Data Register double buffer for SAMPLEB1	0x0000_0000
ADDRDBB2	EADC_BA+0x128	R	A/D Data Register double buffer for SAMPLEB2	0x0000_0000
ADDRDBB3	EADC_BA+0x12C	R	A/D Data Register double buffer for SAMPLEB3	0x0000_0000

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

Bits	Description	
[31:17]	**Reserved**	Reserved.
[16]	**VALID**	**Valid Flag** 0 = Double data in RSLTDB[11:0] bits is not valid. 1 = Double data in RSLTDB[11:0] bits is valid. This bit is set to 1 when corresponding SAMPLE channel analog input conversion is completed and cleared by hardware after ADDRDBAn and ADDRDBBn register is read.(n=0~3).
[15:12]	**Reserved**	Reserved.
[11:0]	**RSLTDB**	**A/D Conversion Result** This field contains 12 bits conversion result if the corresponding DBMAn (n=0~3, ADDBM[3:0]) or DBMBn(n=0~3, ADDBM[11:8]) is set.

A/D Double Buffer Mode select (ADDBM)

Register	Offset	R/W	Description	Reset Value
ADDBM	EADC_BA+0x130	R/W	A/D Double Buffer Mode select	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved				**DBMB3**	**DBMB2**	**DBMB1**	**DBMB0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved				**DBMA3**	**DBMA2**	**DBMA1**	**DBMA0**

Bits	Description	
[31:12]	**Reserved**	Reserved.
[11]	**DBMB3**	**Double Buffer Mode for SAMPLE B3** 0 = SampleB3 has one sample result register. (default) 1 =SampleB3 has two sample result registers.
[10]	**DBMB2**	**Double Buffer Mode for SAMPLE B2** 0 = SampleB2 has one sample result register. (default). 1 =SampleB2 has two sample result registers.
[9]	**DBMB1**	**Double Buffer Mode for SAMPLE B1** 0 = SampleB1 has one sample result register. (default). 1 =SampleB1 has two sample result registers.
[8]	**DBMB0**	**Double Buffer Mode for SAMPLE B0** 0 = SampleB0 has one sample result register. (default) 1 =SampleB0 has two sample result registers.
[7:4]	**Reserved**	Reserved.
[3]	**DBMA3**	**Double Buffer Mode for SAMPLE A3** 0 = SampleA3 has one sample result register. (default). 1 =SampleA3 has two sample result registers.
[2]	**DBMA2**	**Double Buffer Mode for SAMPLE A2** 0 = SampleA2 has one sample result register. (default). 1 =SampleA2 has two sample result registers.
[1]	**DBMA1**	**Double Buffer Mode for SAMPLE A1** 0 = SampleA1 has one sample result register. (default). 1 =SampleA1 has two sample result registers.

Bits	Description	
[0]	**DBMA0**	**Double Buffer Mode for SAMPLE A0** 0 = SampleA0 has one sample result register. (default). 1 =SampleA0 has two sample result registers.

A/D interrupt Source Enable Control Register. (ADINT0SRCTL ~ ADINT3SRCTL)

Register	Offset	R/W	Description	Reset Value
ADINT0SRCTL	EADC_BA+0x134	R/W	A/D interrupt 0 Source Enable Control Register.	0x0000_0000
ADINT1SRCTL	EADC_BA+0x138	R/W	A/D interrupt 1 Source Enable Control Register.	0x0000_0000
ADINT2SRCTL	EADC_BA+0x13C	R/W	A/D interrupt 2 Source Enable Control Register.	0x0000_0000
ADINT3SRCTL	EADC_BA+0x140	R/W	A/D interrupt 3 Source Enable Control Register.	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
IESPLB7	IESPLB6	IESPLB5	IESPLB4	IESPLB3	IESPLB2	IESPLB1	IESPLB0
7	6	5	4	3	2	1	0
IESPLA7	IESPLA6	IESPLA5	IESPLA4	IESPLA3	IESPLA2	IESPLA1	IESPLA0

Bits	Description	
[31:16]	**Reserved**	Reserved.
[15]	**IESPLB7**	**SAMPLE B7 Interrupt Mask Enable Bit** 0 = SAMPLE B7 interrupt mask Disabled. 1 = SAMPLE B7 interrupt mask Enabled.
[14]	**IESPLB6**	**SAMPLE B6 Interrupt Mask Enable Bit** 0 = SAMPLE B6 interrupt mask Disabled. 1 = SAMPLE B6 interrupt mask Enabled.
[13]	**IESPLB5**	**SAMPLE B5 Interrupt Mask Enable Bit** 0 = SAMPLE B5 interrupt mask Disabled. 1 = SAMPLE B5 interrupt mask Enabled.
[12]	**IESPLB4**	**SAMPLE B4 Interrupt Mask Enable Bit** 0 = SAMPLE B4 interrupt mask Disabled. 1 = SAMPLE B4 interrupt mask Enabled.
[11]	**IESPLB3**	**SAMPLE B3 Interrupt Mask Enable Bit** 0 = SAMPLE B3 interrupt mask Disabled. 1 = SAMPLE B3 interrupt mask Enabled.
[10]	**IESPLB2**	**SAMPLE B2 Interrupt Mask Enable Bit** 0 = SAMPLE B2 interrupt mask Disabled. 1 = SAMPLE B2 interrupt mask Enabled.

Bits	Description	
[9]	**IESPLB1**	**SAMPLE B1 Interrupt Mask Enable Bit** 0 = SAMPLE B1 interrupt mask Disabled. 1 = SAMPLE B1 interrupt mask Enabled.
[8]	**IESPLB0**	**SAMPLE B0 Interrupt Mask Enable Bit** 0 = SAMPLE B0 interrupt mask Disabled. 1 = SAMPLE B0 interrupt mask Enabled.
[7]	**IESPLA7**	**SAMPLE A7 Interrupt Mask Enable Bit** 0 = SAMPLE A7 interrupt mask Disabled. 1 = SAMPLE A7 interrupt mask Enabled.
[6]	**IESPLA6**	**SAMPLE A6 Interrupt Mask Enable Bit** 0 = SAMPLE A6 interrupt mask Disabled. 1 = SAMPLE A6 interrupt mask Enabled.
[5]	**IESPLA5**	**SAMPLE A5 Interrupt Mask Enable Bit** 0 = SAMPLE A5 interrupt mask Disabled. 1 = SAMPLE A5 interrupt mask Enabled.
[4]	**IESPLA4**	**SAMPLE A4 Interrupt Mask Enable Bit** 0 = SAMPLE A4 interrupt mask Disabled. 1 = SAMPLE A4 interrupt mask Enabled.
[3]	**IESPLA3**	**SAMPLE A3 Interrupt Mask Enable Bit** 0 = SAMPLE A3 interrupt mask Disabled. 1 = SAMPLE A3 interrupt mask Enabled.
[2]	**IESPLA2**	**SAMPLE A2 Interrupt Mask Enable Bit** 0 = SAMPLE A2 interrupt mask Disabled. 1 = SAMPLE A2 interrupt mask Enabled.
[1]	**IESPLA1**	**SAMPLE A1 Interrupt Mask Enable Bit** 0 = SAMPLE A1 interrupt mask Disabled. 1 = SAMPLE A1 interrupt mask Enabled.
[0]	**IESPLA0**	**SAMPLE A0 Interrupt Mask Enable Bit** 0 = SAMPLE A0 interrupt mask Disabled. 1 = SAMPLE A0 interrupt mask Enabled.

A/D SAMPLE Trigger Condition Enable Control Register (SMPTRGA0~ SMPTRGA3, SMPTRGB0~ SMPTRGB3)

Register	Offset	R/W	Description	Reset Value
SMPTRGA0	EADC_BA+0x144	R/W	A/D trigger condition for SAMPLEA0	0x0000_0000
SMPTRGA1	EADC_BA+0x148	R/W	A/D trigger condition for SAMPLEA1	0x0000_0000
SMPTRGA2	EADC_BA+0x14C	R/W	A/D trigger condition for SAMPLEA2	0x0000_0000
SMPTRGA3	EADC_BA+0x150	R/W	A/D trigger condition for SAMPLEA3	0x0000_0000
SMPTRGB0	EADC_BA+0x154	R/W	A/D trigger condition for SAMPLEB0	0x0000_0000
SMPTRGB1	EADC_BA+0x158	R/W	A/D trigger condition for SAMPLEB1	0x0000_0000
SMPTRGB2	EADC_BA+0x15C	R/W	A/D trigger condition for SAMPLEB2	0x0000_0000
SMPTRGB3	EADC_BA+0x160	R/W	A/D trigger condition for SAMPLEB3	0x0000_0000

31	30	29	28	27	26	25	24
PWM21CEN	PWM21PEN	PWM21FEN	PWM21REN	PWM20CEN	PWM20PEN	PWM20FEN	PWM20REN
23	22	21	20	19	18	17	16
PWM14CEN	PWM14PEN	PWM14FEN	PWM14REN	PWM12CEN	PWM12PEN	PWM12FEN	PWM12REN
15	14	13	12	11	10	9	8
PWM10CEN	PWM10PEN	PWM10FEN	PWM10REN	PWM04CEN	PWM04PEN	PWM04FEN	PWM04REN
7	6	5	4	3	2	1	0
PWM02CEN	PWM02PEN	PWM02FEN	PWM02REN	PWM00CEN	PWM00PEN	PWM00FEN	PWM00REN

Bits	Description	
[31]	**PWM21CEN**	**BPWM0_CH1 Center Trigger Enable Bit** 0 = BPWM0_CH1 center trigger Disabled. 1 = BPWM0_CH1 center trigger Enabled.
[30]	**PWM21PEN**	**BPWM0_CH1 Period Trigger Enable Bit** 0 = BPWM0_CH1 period trigger Disabled. 1 = BPWM0_CH1 period trigger Enabled.
[29]	**PWM21FEN**	**BPWM0_CH1 Falling Edge Trigger Enable Bit** 0 = BPWM0_CH1 falling edge trigger Disabled. 1 = BPWM0_CH1 falling edge trigger Enabled.
[28]	**PWM21REN**	**BPWM0_CH1 Rising Edge Trigger Enable Bit** 0 = BPWM0_CH1 rising edge trigger Disabled. 1 = BPWM0_CH1 rising edge trigger Enabled.

Bits	Description	
[27]	**PWM20CEN**	**BPWM0_CH0 Center Trigger Enable Bit** 0 = BPWM0_CH0 center trigger Disabled. 1 = BPWM0_CH0 center trigger Enabled.
[26]	**PWM20PEN**	**BPWM0_CH0 Period Trigger Enable Bit** 0 = BPWM0_CH0 period trigger Disabled. 1 = BPWM0_CH0 period trigger Enabled.
[25]	**PWM20FEN**	**BPWM0_CH0 Falling Edge Trigger Enable Bit** 0 = BPWM0_CH0 falling edge trigger Disabled. 1 = BPWM0_CH0 falling edge trigger Enabled.
[24]	**PWM20REN**	**BPWM0_CH0 Rising Edge Trigger Enable Bit** 0 = BPWM0_CH0 rising edge trigger Disabled. 1 = BPWM0_CH0 rising edge trigger Enabled.
[23]	**PWM14CEN**	**PWM1_CH4 Center Trigger Enable Bit** 0 = PWM1_CH4 center trigger Disabled. 1 = PWM1_CH4 center trigger Enabled.
[22]	**PWM14PEN**	**PWM1_CH4 Period Trigger Enable Bit** 0 = PWM1_CH4 period trigger Disabled. 1 = PWM1_CH4 period trigger Enabled.
[21]	**PWM14FEN**	**PWM1_CH4 Falling Edge Trigger Enable Bit** 0 = PWM1_CH4 falling edge trigger Disabled. 1 = PWM1_CH4 falling edge trigger Enabled.
[20]	**PWM14REN**	**PWM1_CH4 Rising Edge Trigger Enable Bit** 0 = PWM1_CH4 rising edge trigger Disabled. 1 = PWM1_CH4 rising edge trigger Enabled.
[19]	**PWM12CEN**	**PWM1_CH2 Center Trigger Enable Bit** 0 = PWM1_CH2 center trigger Disabled. 1 = PWM1_CH2 center trigger Enabled.
[18]	**PWM12PEN**	**PWM1_CH2 Period Trigger Enable Bit** 0 = PWM1_CH2 period trigger Disabled. 1 = PWM1_CH2 period trigger Enabled.
[17]	**PWM12FEN**	**PWM1_CH2 Falling Edge Trigger Enable Bit** 0 = PWM1_CH2 falling edge trigger Disabled. 1 = PWM1_CH2 falling edge trigger Enabled.
[16]	**PWM12REN**	**PWM1_CH2 Rising Edge Trigger Enable Bit** 0 = PWM1_CH2 rising edge trigger Disabled. 1 = PWM1_CH2 rising edge trigger Enabled.
[15]	**PWM10CEN**	**PWM1_CH0 Center Trigger Enable Bit** 0 = PWM1_CH0 center trigger Disabled. 1 = PWM1_CH0 center trigger Enabled.
[14]	**PWM10PEN**	**PWM1_CH0 Period Trigger Enable Bit** 0 = PWM1_CH0 period trigger Disabled. 1 = PWM1_CH0 period trigger Enabled.

Bits	Description	
[13]	**PWM10FEN**	**PWM1_CH0 Falling Edge Trigger Enable Bit** 0 = PWM1_CH0 falling edge trigger Disabled. 1 = PWM1_CH0 falling edge trigger Enabled.
[12]	**PWM10REN**	**PWM1_CH0 Rising Edge Trigger Enable Bit** 0 = PWM1_CH0 rising edge trigger Disabled. 1 = PWM1_CH0 rising edge trigger Enabled.
[11]	**PWM04CEN**	**PWM0_CH4 Center Trigger Enable Bit** 0 = PWM0_CH4 center trigger Disabled. 1 = PWM0_CH4 center trigger Enabled.
[10]	**PWM04PEN**	**PWM0_CH4 Period Trigger Enable Bit** 0 = PWM0_CH4 period trigger Disabled. 1 = PWM0_CH4 period trigger Enabled.
[9]	**PWM04FEN**	**PWM0_CH4 Falling Rdge Trigger Enable Bit** 0 = PWM0_CH4 falling edge trigger Disabled. 1 = PWM0_CH4 falling edge trigger Enabled.
[8]	**PWM04REN**	**PWM0_CH4 Rising Edge Trigger Enable Bit** 0 = PWM0_CH4 rising edge trigger Disabled. 1 = PWM0_CH4 rising edge trigger Enabled.
[7]	**PWM02CEN**	**PWM0_CH2 Center Trigger Enable Bit** 0 = PWM0_CH2 center trigger Disabled. 1 = PWM0_CH2 center trigger Enabled.
[6]	**PWM02PEN**	**PWM0_CH2 Period Trigger Enable Bit** 0 = PWM0_CH2 period trigger Disabled. 1 = PWM0_CH2 period trigger Enabled.
[5]	**PWM02FEN**	**PWM0_CH2 Falling Edge Trigger Enable Bit** 0 = PWM0_CH2 falling edge trigger Disabled. 1 = PWM0_CH2 falling edge trigger Enabled.
[4]	**PWM02REN**	**PWM0_CH2 Rising Edge Trigger Enable Bit** 0 = PWM0_CH2 rising edge trigger Disabled. 1 = PWM0_CH2 rising edge trigger Enabled.
[3]	**PWM00CEN**	**PWM0_CH0 Center Trigger Enable Bit** 0 = PWM0_CH0 center trigger Disabled. 1 = PWM0_CH0 center trigger Enabled.
[2]	**PWM00PEN**	**PWM0_CH0 Period Trigger Enable Bit** 0 = PWM0_CH0 period trigger Disabled. 1 = PWM0_CH0 period trigger Enabled.
[1]	**PWM00FEN**	**PWM0_CH0 Falling Edge Trigger Enable Bit** 0 = PWM0_CH0 falling edge trigger Disabled. 1 = PWM0_CH0 falling edge trigger Enabled.
[0]	**PWM00REN**	**PWM0_CH0 Rising Edge Trigger Enable Bit** 0 = PWM0_CH0 rising edge trigger Disabled. 1 = PWM0_CH0 rising edge trigger Enabled.

6.20 Analog Comparator (ACMP)

6.20.1 Overview

The NuMicro®NM1530 Series contains three comparators. The comparator output is logic 1 when positive input voltage is greater than negative input voltage; otherwise the output is logic 0. Each comparator can be configured to cause an interrupt when the comparator output value changes. The block diagram is shown in Figure 6–175.

6.20.2 Features

- Analog input voltage range: 0~AV_{DD}
- Supports hysteresis function
- Supports wake-up function
- Supports comparator output inverse function
- Supports the comparator output can be the brake source for EPWM function
- ACMP0 supports
  - 2 positive sources:ACMP0_P and OP0_O
  - 2 negative sources: ACMP0_N and Internal band-gap voltage (V_{BG})
- ACMP1 supports
  - 2 positive sources:ACMP1_P and OP1_O
  - 2 negative sources: ACMP1_N and Internal band-gap voltage (V_{BG})
- ACMP2 supports
  - 1 positive sources:ACMP2_P
  - 2 negative sources: ACMP2_N and Internal band-gap voltage (V_{BG})
- Shares one ACMP interrupt vector for all comparators

6.20.3 Block Diagram

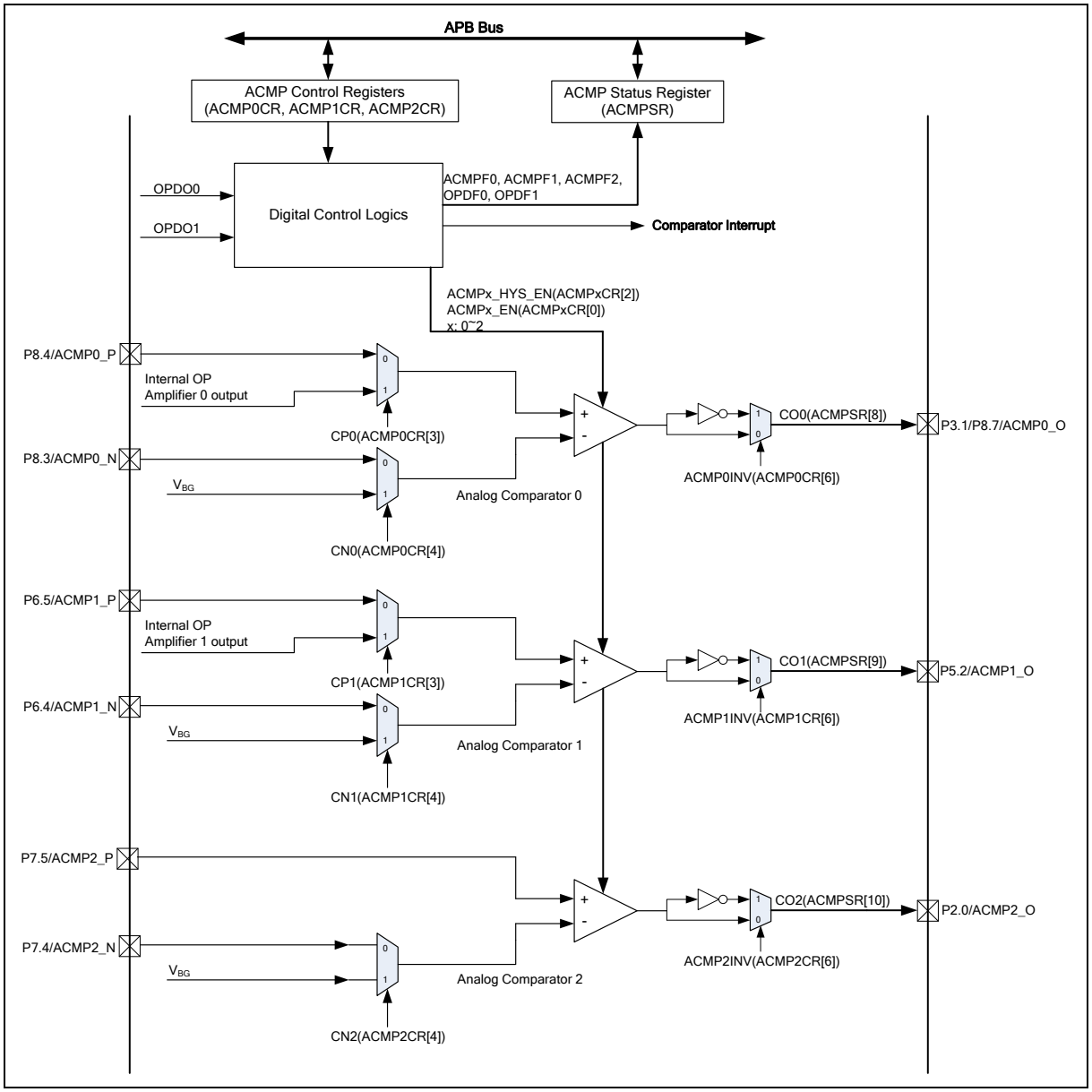


Figure 6–175 Analog Comparator Block Diagram

6.20.4 Basic Configuration

ACMP clock source is PCLK and can be enabled by setting ACMP_EN (APBCLK[22]). The ACMP pin functions are configured in P2_MFP, P3_MFP, P5_MFP, P6_MFP, P7_MFP and P8_MFP registers. It is recommended to disable the digital input path of the analog input pins to avoid the leakage current. The digital input path can be disabled by configuring P6_OFFD, P7_OFFD and P8_OFFD registers.

6.20.5 Functional Description

6.20.5.1 Interrupt Sources

The output of comparators are sampled by PCLK and reflected at CO0(ACMPSR[8]),CO1(ACMPSR[9]) and CO2(ACMPSR[10]). If ACMPIEx(ACMPxCR[1]) is set to 1, the comparator interrupt will be enabled.As the output state of comparator is changed, the comparator interrupt will be asserted and the corresponding flag, ACMPFx(ACMPSR[x]), will be set. Software can clear the flag to 0 by writing 1 to it.

OPDF0(ACMPSR[4]), OPDF1(ACMPSR[5]) interrupt flags are set respectively by hardware whenever the OP0 and OP1 amplifier Schmitt trigger non-inverting buffer output change states. Software can clear the flag to 0 by writing 1 to it.

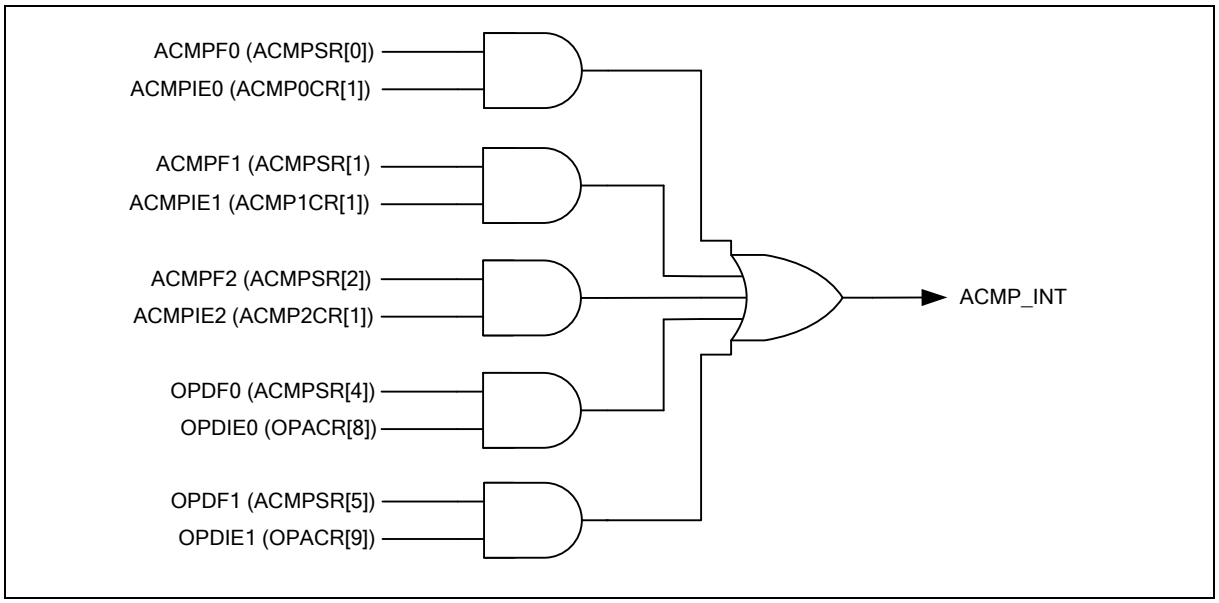


Figure 6–176 Analog Comparator Controller Interrupt

6.20.5.2 Hysteresis Function

The analog comparator provides hysteresis function to make the comparator output transition more stable. If comparator output is 0, it will not change to 1 until the positive input voltage exceeds the negative input voltage by a positive hysteresis voltage. Similarly, if comparator output is 1, it will not change to 0 until the positive input voltage drops below the negative input voltage by a negative hysteresis voltage.

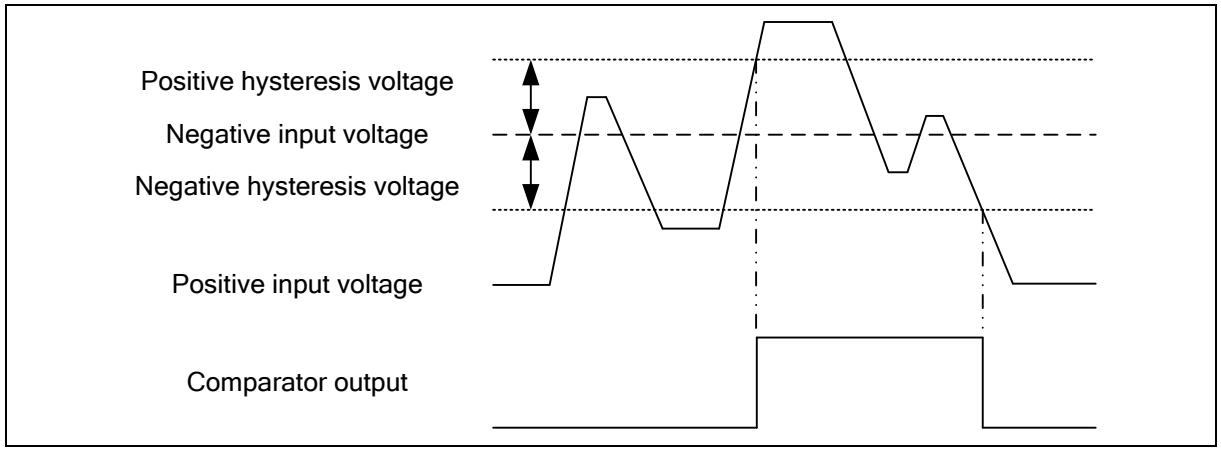


Figure 6-177 Comparator Hysteresis function

6.20.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 0 can be written

Register	Offset	R/W	Description	Reset Value
ACMP Base Address: **ACMP_BA = 0x400D_0000**				
ACMP0CR	ACMP_BA+0x00	R/W	Analog Comparator 0 Control Register	0x0000_0000
ACMP1CR	ACMP_BA+0x04	R/W	Analog Comparator 1 Control Register	0x0000_0000
ACMP2CR	ACMP_BA+0x08	R/W	Analog Comparator 2 Control Register	0x0000_0000
ACMPSR	ACMP_BA+0x0C	R/W	Analog Comparator Status Register	0x0000_0000

6.20.7 Register Description

Analog CMP 0 Control Register (ACMP0CR)

Register	Offset	R/W	Description	Reset Value
ACMP0CR	ACMP_BA+0x00	R/W	Analog Comparator 0 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	**22**	**21**	**20**	**19**	**18**	**17**	**16**
Reserved							
15	**14**	**13**	**12**	**11**	**10**	**9**	**8**
Reserved							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved	**ACMP0INV**	**Reserved**	**CN0**	**CP0**	**ACMP0_HYS_EN**	**ACMPIE0**	**ACMP0_EN**

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**ACMP0INV**	**Analog Comparator0Output Inverse Select** 0 = The comparator output inverse function Disabled. 1 = The comparator output inverse function Enabled.
[5]	**Reserved**	Reserved.
[4]	**CN0**	**Analog Comparator0 Negative Input Select** 0 = The comparator reference pin P8.3/ACMP0_N is selected as the negative comparator input. 1 = The internal band-gapvoltage (V_{BG}) is selected as the negative comparator input.
[3]	**CP0**	**Analog Comparator0Positive Input Select** 0 = The comparator reference pin P8.4/ACMP0_P is selected as the positivecomparator input. 1 = The internal OP amplifier 0 output is selected as the positive comparator input.
[2]	**ACMP0_HYS_EN**	**CMP Hysteresis Enable Bit** 0 = Hysteresis function Disabled. 1 = Hysteresis function Enabled.
[1]	**ACMPIE0**	**Analog Comparator 0 Interrupt Enable Bit** 0 = Interrupt function Disabled. 1 = Interrupt function Enabled.
[0]	**ACMP0_EN**	**Analog Comparator0 Enable Bit** 0 = Analog comparator Disabled. 1 = Analog comparator Enabled.

Analog CMP 1 Control Register (ACMP1CR)

Register	Offset	R/W	Description	Reset Value
ACMP1CR	ACMP_BA+0x04	R/W	Analog Comparator 1 Control Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved							
7	6	5	4	3	2	1	0
Reserved	ACMP1INV	Reserved	CN1	CP1	ACMP1_HYS_EN	ACMPIE1	ACMP1_EN

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**ACMP1INV**	**Analog Comparator1Output Inverse Select** 0 = The comparator output inverse function Disabled. 1 = The comparator output inverse function Enabled.
[5]	**Reserved**	Reserved.
[4]	**CN1**	**Analog Comparator1 Negative Input Select** 0 = The comparator reference pin P6.4/ACMP1_N is selected as the negative comparator input. 1 = The internal band-gapvoltage (V_{BG}) is selected as the negative comparator input.
[3]	**CP1**	**Analog Comparator1Positive Input Select** 0 = The comparator reference pin P6.5/ACMP1_P is selected as the positivecomparator input. 1 = The internal OP amplifier 1 output is selected as the positive comparator input.
[2]	**ACMP1_HYS_EN**	**CMP Hysteresis Enable Bit** 0 = Hysteresis function Disabled. 1 = Hysteresis function Enabled.
[1]	**ACMPIE1**	**Analog Comparator 1 Interrupt Enable Bit** 0 = Interrupt function Disabled. 1 = Interrupt function Enabled.
[0]	**ACMP1_EN**	**Analog Comparator1 Enable Bit** 0 = Analog comparator Disabled. 1 = Analog comparator Enabled.

Analog CMP 2 Control Register (ACMP2CR)

Register	Offset	R/W	Description	Reset Value
ACMP2CR	ACMP_BA+0x08	R/W	Analog Comparator 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							
7	6	5	4	3	2	1	0
Reserved	ACMP2INV	Reserved	CN2	Reserved	ACMP2_HYS_EN	ACMPIE2	ACMP2_EN

Bits	Description	
[31:7]	**Reserved**	Reserved.
[6]	**ACMP2INV**	**Analog Comparator2Output Inverse Select** 0 = The comparator output inverse function Disabled. 1 = The comparator output inverse function Enabled.
[5]	**Reserved**	Reserved.
[4]	**CN2**	**Analog Comparator2 Negative Input Select** 0 = The comparator reference pin P7.4/ACMP2_N is selected as the negative comparator input. 1 = The internal band-gap voltage (V_{BG}) is selected as the negative comparator input.
[3]	**Reserved**	Reserved.
[2]	**ACMP2_HYS_EN**	**CMP Hysteresis Enable Bit** 0 = Hysteresis function Disabled. 1 = Hysteresis function Enabled.
[1]	**ACMPIE2**	**Analog Comparator 2 Interrupt Enable Bit** 0 = Interrupt function Disabled. 1 = Interrupt function Enabled.
[0]	**ACMP2_EN**	**Analog Comparator2 Enable Bit** 0 = Analog comparator Disabled. 1 = Analog comparator Enabled.

Analog CMP Status Register (ACMPSR)

Register	Offset	R/W	Description	Reset Value
ACMPSR	ACMP_BA+0x0C	R/W	Analog Comparator Status Register	0x0000_0000

31	30	29	28	27	26	25	24
Reserved							
23	22	21	20	19	18	17	16
Reserved							
15	14	13	12	11	10	9	8
Reserved					CO2	CO1	CO0
7	6	5	4	3	2	1	0
Reserved		OPDF1	OPDF0	Reserved	ACMPF2	ACMPF1	ACMPF0

Bits	Description	
[31:11]	**Reserved**	Reserved.
[10]	**CO2**	**Compare 2 Output** Synchronized to the PCLK clock to allow reading by software. Cleared when the comparator is disabled (ACMP2_EN = 0).
[9]	**CO1**	**Compare 1 Output** Synchronized to the PCLK clock to allow reading by software. Cleared when the comparator is disabled (ACMP1_EN = 0).
[8]	**CO0**	**Compare 0 Output** Synchronized to the PCLK clock to allow reading by software. Cleared when the comparator is disabled (ACMP0_EN = 0).
[7:6]	**Reserved**	Reserved.
[5]	**OPDF1**	**OP Amplifier 1 Schmitt Trigger DigitalOutput Interrupt Flag** OPDF1 interrupt flag is set by hardware whenever the OP amplifier 1 Schmitt trigger non-inverting buffer digital output changes state. **Note:** This bit is remapping from OPASR[5] and writing 1 to ACMPSR[5] or OPASR[5] can clear this bit.
[4]	**OPDF0**	**OP Amplifier 0 Schmitt Trigger DigitalOutput Interrupt Flag** OPDF0 interrupt flag is set by hardware whenever the OP amplifier 0 Schmitt trigger non-inverting buffer digital output changes state. **Note:** This bit is remapping from OPASR[4] and writing 1 to ACMPSR[4] or OPASR[4] can clear this bit.
[3]	**Reserved**	Reserved.
[2]	**ACMPF2**	**Compare 2 Flag** This bit is set by hardware whenever the comparator 2 output changes state. This will cause an interrupt if ACMP2CR[1]is set to 1. Write 1 to clear this bit to 0.

[1]	**ACMPF1**	**Compare 1 Flag** This bit is set by hardware whenever the comparator 1 output changes state. This will cause an interrupt if ACMP1CR[1]is set to 1. Write 1 to clear this bit to 0.
[0]	**ACMPF0**	**Compare 0 Flag** This bit is set by hardware whenever the comparator 0 output changes state. This will cause an interrupt if ACMP0CR[1]is set to 1. Write 1 to clear this bit to 0.

6.21 OP Amplifier (OPA)

6.21.1 Overview

This device integrated two operational amplifiers. It can be enabled through OP0_EN (OPACR[0])and OP1_EN (OPACR[1]) bit. User can measure the output of the OP amplifier throughthe integrated A/D converter.

6.21.2 Features

- Analog input voltage range: 0~AV_{DD}
- Supports two analog OP amplifiers
- Supports OP output voltage measurement by A/D converter
- Supports Schmitt trigger buffer outputs and generate interrupt
- OP amplifier 0 output can be an optional input source of integrated comparator 0 positive input
- OP amplifier 1 output can be an optional input source of integrated comparator 1 positive input

6.21.3 Block Diagram

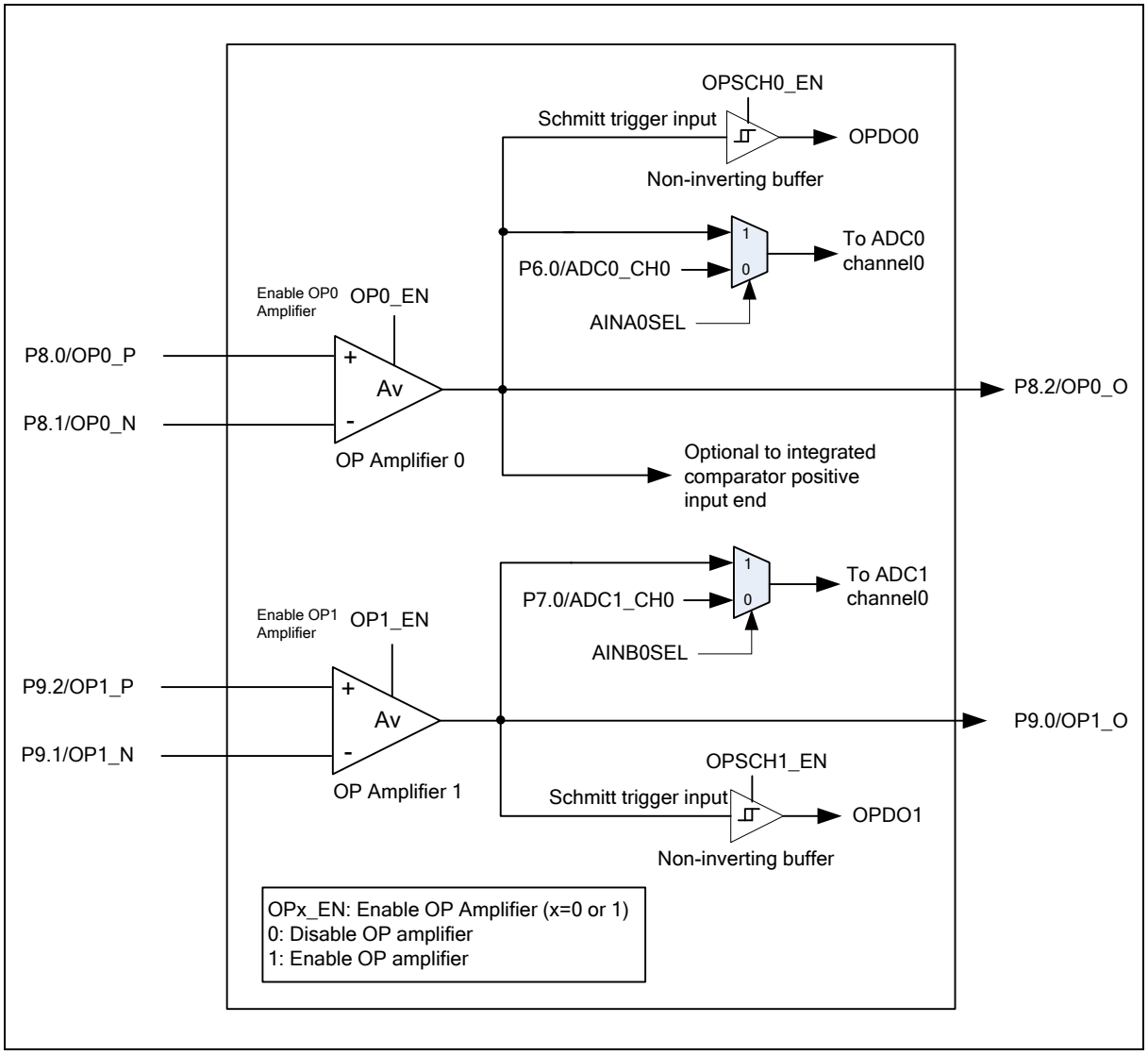


Figure 6–178 OP Amplifier Block Diagram

6.21.4 Basic Configuration

OPA clock source is PCLK and can be enabled by setting OPA_EN (APBCLK[29]). The OPA pin functions are configured in P8_MFP and P9_MFP registers. It is recommended to disable the digital input path of the analog input pins to avoid the leakage current. The digital input path can be disabled by configuring P8_OFFD and P9_OFFD registers.

6.21.5 Interrupt Sources

The OPDF0 (OPASR[4]), OPDF1 (OPASR[5]) interrupt flags are set respectively by hardware whenever the OP0, 1 amplifier Schmitt trigger non-inverting buffer output change states. The flag bit is cleared by writing 1 to itself. Schmitt trigger buffer outputs of the OP amplifier0, 1 can be one of the comparator interrupt sources.

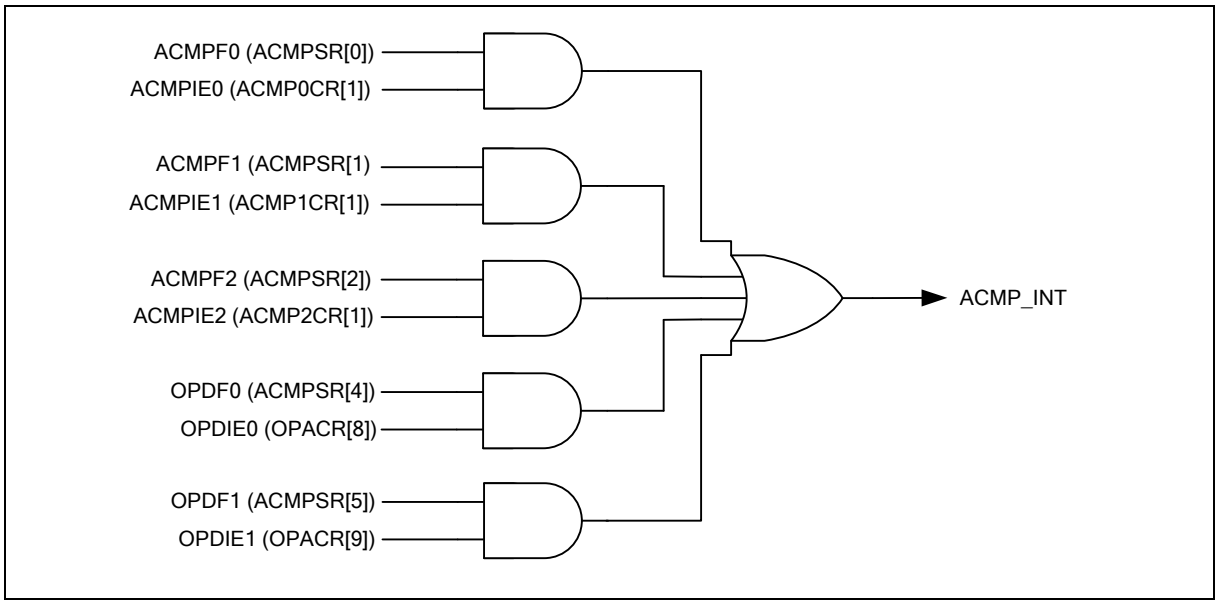


Figure 6–179 OP Amplifier Interrupt Flags for Analog Comparator Interrupt

6.21.6 Register Map

R: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 0 can be written

Register	Offset	R/W	Description	Reset Value
OPA Base Address: **OPA_BA = 0x400F_0000**				
OPACR	OPA_BA+0x00	R/W	OP Amplifier Control Register	0x0000_0000
OPASR	OPA_BA+0x04	R/W	OP Amplifier Status Register	0x0000_0000

6.21.7 Register Description

OPA Control Register (OPACR)

Register	Offset	R/W	Description	Reset Value
OPACR	OPA_BA+0x00	R/W	OP Amplifier 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						**OPDIE1**	**OPDIE0**
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Reserved		**OPSCH1_EN**	**OPSCH0_EN**	**Reserved**		**OP1_EN**	**OP0_EN**

Bits	Description	
[31:10]	**Reserved**	Reserved.
[9]	**OPDIE1**	**OP Amplifier 1 Schmitt Trigger Digital Output Interrupt Enable Bit** 0 = OP Amplifier 1 digital output interrupt function Disabled. 1 = OP Amplifier 1 digital output interrupt function Enabled. The OPDF1 interrupt flag is set by hardware whenever the OP amplifier 1 Schmitt trigger non-inverting buffer digital output changes state, in the meanwhile, if OPDIE1 is set to 1, a comparator interrupt request is generated.
[8]	**OPDIE0**	**OP Amplifier 0 Schmitt Trigger Digital Output Interrupt Enable Bit** 0 = OP Amplifier 0 digital output interrupt function Disabled. 1 = OP Amplifier 0 digital output interrupt function Enabled. The OPDF0 interrupt flag is set by hardware whenever the OP amplifier 0 Schmitt trigger non-inverting buffer digital output changes state, in the meanwhile, if OPDIE0 is set to 1, a comparator interrupt request is generated.
[7:6]	**Reserved**	Reserved.
[5]	**OPSCH1_EN**	**OP Amplifier 1 Schmitt Trigger Non-inverting Buffer Enable Bit** 0 = OP Amplifier 1 Schmitt trigger Disabled. 1 = OP Amplifier 1 Schmitt trigger Enabled.
[4]	**OPSCH0_EN**	**OP Amplifier 0 Schmitt Trigger Non-inverting Buffer Enable Bit** 0 = OP Amplifier 0 Schmitt trigger Disabled. 1 = OP Amplifier 0 Schmitt trigger Enabled.
[3:2]	**Reserved**	Reserved.
[1]	**OP1_EN**	**OP Amplifier 1 Enable Bit** 0 = OP Amplifier 1 Disabled. 1 = OP Amplifier 1 Enabled.

Bits	Description	
[0]	**OP0_EN**	**OP Amplifier 0 Enable Bit** 0 = OP Amplifier 0 Disabled. 1 = OP Amplifier 0 Enabled.

OPA Status Register (OPASR)

Register	Offset	R/W	Description	Reset Value
OPASR	OPA_BA+0x04	R/W	OP Amplifier 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		**OPDF1**	**OPDF0**	**Reserved**		**OPDO1**	**OPDO0**

Bits	Description	
[31:6]	**Reserved**	Reserved.
[5]	**OPDF1**	**OP Amplifier 1 Schmitt Trigger DigitalOutput Interrupt Flag** OPDF1 interrupt flag is set by hardware whenever the OP amplifier 1 Schmitt trigger non-inverting buffer digital output changes state. **Note:** This bit is cleared by writing 1 to it.
[4]	**OPDF0**	**OP Amplifier 0 Schmitt Trigger DigitalOutput Interrupt Flag** OPDF0 interrupt flag is set by hardware whenever the OP amplifier 0 Schmitt trigger non-inverting buffer digital output changes state. **Note:** This bit is cleared by writing 1 to it.
[3:2]	**Reserved**	Reserved.
[1]	**OPDO1**	**OP Amplifier 1 DigitalOutput** Synchronized to the APB clock to allow reading by software. Cleared when the Schmitt trigger buffer is disabled (OPSCH1_EN = 0).
[0]	**OPDO0**	**OP Amplifier 0 DigitalOutput** Synchronized to the APB clock to allow reading by software. Cleared when the Schmitt trigger buffer is disabled (OPSCH0_EN = 0).

7 ELECTRICAL CHARACTERISTICS

7.1 Absolute Maximum Ratings

SYMBOL	PARAMETER	MIN	MAX	UNIT
DC Power Supply	V_{DD}–V_{SS}	-0.3	+6.3	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	105	°C
Storage Temperature	T_{ST}	-55	+150	°C
Maximum Current into VDD		-	120	mA
Maximum Current out of VSS			120	mA
Maximum Current sunk by a I/O pin			35	mA
Maximum Current sourced by a I/O pin			35	mA
Maximum Current sunk by total I/O pins			100	mA
Maximum Current sourced by total I/O pins			100	mA

Note: Exposure to conditions beyond those listed under absolute maximum ratings may adversely affects the lift and reliability of the device.

7.2 DC Electrical Characteristics

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS				
		MIN.	TYP.	MAX.	UNIT					
Operation voltage	V_{DD}	2.5	-	5.5	V	V_{DD} =2.5V ~ 5.5V				
Power Ground	V_{SS}/ AV_{SS}	-0.3	-	-	V					
LDO Output Voltage	V_{LDO}	1.62	1.8	1.98	V	V_{DD}>= 2.5V				
Analog Operating Voltage	AV_{DD}	2.5	-	V_{DD}	V					
Analog Reference Voltage	V_{REF}	1.2	-	AV_{DD}	V					
Operating Current Normal Run Mode At 72MHz while(1){} executed from flash V_{LDO} =1.8V	I_{DD1}		38.7		mA	VDD	HXT	HIRC	PLL	All digital module
						5.5V	12MHz	X	✓	✓
	I_{DD2}		17.9		mA	5.5V	12MHz	X	✓	X
	I_{DD3}		37.2		mA	3.3V	12MHz	X	✓	✓
	I_{DD4}		16.4		mA	3.3V	12MHz	X	✓	X
Operating Current Normal Run Mode At 60MHz while(1){} executed from flash V_{LDO} =1.8V	I_{DD5}		32.9		mA	5.5V	12MHz	X	✓	✓
	I_{DD6}		15.5		mA	5.5V	12MHz	X	✓	X
	I_{DD7}		31.4		mA	3.3V	12MHz	X	✓	✓
	I_{DD8}		14.0		mA	3.3V	12MHz	X	✓	X
Operating Current Normal Run Mode At 50MHz while(1){} executed from flash V_{LDO} =1.8V	I_{DD9}		29.1		mA	5.5V	12MHz	X	✓	✓
	I_{DD10}		14.5		mA	5.5V	12MHz	X	✓	X
	I_{DD11}		27.6		mA	3.3V	12MHz	X	✓	✓
	I_{DD12}		13.0		mA	3.3V	12MHz	X	✓	X
Operating Current Normal Run Mode At 48MHz while(1){} executed from flash V_{LDO} =1.8V	I_{DD13}		28.1		mA	5.5V	12MHz	X	✓	✓
	I_{DD14}		14.0		mA	5.5V	12MHz	X	✓	X
	I_{DD15}		26.6		mA	3.3V	12MHz	X	✓	✓
	I_{DD16}		12.5		mA	3.3V	12MHz	X	✓	X
Operating Current Normal Run Mode At 32MHz while(1){} executed from flash V_{LDO} =1.8V	I_{DD17}		19.9		mA	5.5V	12MHz	X	✓	✓
	I_{DD18}		10.4		mA	5.5V	12MHz	X	✓	X
	I_{DD19}		18.4		mA	3.3V	12MHz	X	✓	✓
	I_{DD20}		8.9		mA	3.3V	12MHz	X	✓	X

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS				
		MIN.	TYP.	MAX.	UNIT					
Operating Current Normal Run Mode At 22.1184MHz while(1){} executed from flash V_{LDO} =1.8V	I_{DD21}		11.7		mA	5.5V	X	✓	X	✓
	I_{DD22}		5.3		mA	5.5V	X	✓	X	X
	I_{DD23}		11.6		mA	3.3V	X	✓	X	✓
	I_{DD24}		5.2		mA	3.3V	X	✓	X	X
Operating Current Normal Run Mode At 12MHz while(1){} executed from flash V_{LDO} =1.8V	I_{DD25}		8.6		mA	5.5V	12MHz	X	X	✓
	I_{DD26}		4.9		mA	5.5V	12MHz	X	X	X
	I_{DD27}		7.2		mA	3.3V	12MHz	X	X	✓
	I_{DD28}		3.4		mA	3.3V	12MHz	X	X	X
Operating Current Normal Run Mode At 10kHz while(1){} executed from flash V_{LDO} =1.8V	I_{DD29}		0.13		mA	VDD	HXT/ LXT	LIRC	PLL	All digital module
						5.5V	X	10KHz	X	✓
	I_{DD30}		0.12		mA	5.5V	X	10KHz	X	X
	I_{DD31}		0.11		mA	3.3V	X	10KHz	X	✓
	I_{DD32}		0.11		mA	3.3V	X	10KHz	X	X
Operating Current Idle Mode At 72MHz while(1){} executed from flash V_{LDO} =1.8V	I_{IDLE1}		30.1		mA	VDD	HXT	HIRC	PLL	All digital module
						5.5V	12MHz	X	✓	✓
	I_{IDLE2}		9.2		mA	5.5V	12MHz	X	✓	X
	I_{IDLE3}		28.6		mA	3.3V	12MHz	X	✓	✓
	I_{IDLE4}		7.7		mA	3.3V	12MHz	X	✓	X
Operating Current Idle Mode At 60MHz while(1){} executed from flash V_{LDO} =1.8V	I_{IDLE5}		25.7		mA	5.5V	12MHz	X	✓	✓
	I_{IDLE6}		8.2		mA	5.5V	12MHz	X	✓	X
	I_{IDLE7}		24.2		mA	3.3V	12MHz	X	✓	✓
	I_{IDLE8}		6.7		mA	3.3V	12MHz	X	✓	X
Operating Current Idle Mode At 50MHz while(1){} executed from flash V_{LDO} =1.8V	I_{IDLE9}		23.0		mA	5.5V	12MHz	X	✓	✓
	I_{IDLE10}		8.4		mA	5.5V	12MHz	X	✓	X
	I_{IDLE11}		21.5		mA	3.3V	12MHz	X	✓	✓
	I_{IDLE12}		6.9		mA	3.3V	12MHz	X	✓	X

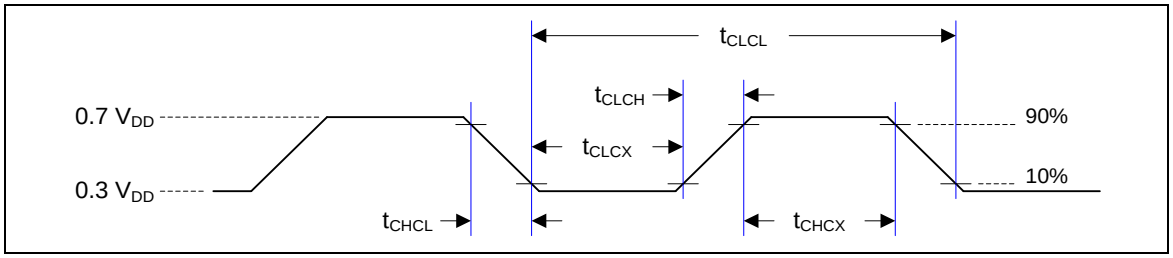
PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS				
		MIN.	TYP.	MAX.	UNIT					
Operating Current Idle Mode At 48MHz while(1){} executed from flash V_{LDO} =1.8V	I_{IDLE13}		22.3		mA	5.5V	12MHz	X	✓	✓
	I_{IDLE14}		8.1		mA	5.5V	12MHz	X	✓	X
	I_{IDLE15}		20.8		mA	3.3V	12MHz	X	✓	✓
	I_{IDLE16}		6.7		mA	3.3V	12MHz	X	✓	X
Operating Current Idle Mode At 32MHz while(1){} executed from flash V_{LDO} =1.8V	I_{IDLE17}		16.0		mA	5.5V	12MHz	X	✓	✓
	I_{IDLE18}		6.4		mA	5.5V	12MHz	X	✓	X
	I_{IDLE19}		14.5		mA	3.3V	12MHz	X	✓	✓
	I_{IDLE20}		4.9		mA	3.3V	12MHz	X	✓	X
Operating Current Idle Mode At 22.1184MHz while(1){} executed from flash V_{LDO} =1.8V	I_{IDLE21}		8.6		mA	5.5V	X	✓	X	✓
	I_{IDLE22}		2.2		mA	5.5V	X	✓	X	X
	I_{IDLE23}		8.6		mA	3.3V	X	✓	X	✓
	I_{IDLE24}		2.2		mA	3.3V	X	✓	X	X
Operating Current Idle Mode At 12MHz while(1){} executed from flash V_{LDO} =1.8V	I_{IDLE25}		7.2		mA	5.5V	12MHz	X	X	✓
	I_{IDLE26}		3.4		mA	5.5V	12MHz	X	X	X
	I_{IDLE27}		5.7		mA	3.3V	12MHz	X	X	✓
	I_{IDLE28}		1.9		mA	3.3V	12MHz	X	X	X
Operating Current Idle Mode At 10kHz while(1){} executed from flash V_{LDO} =1.8V	I_{IDLE29}		0.13		mA	VDD	HXT/ LXT	LIRC	PLL	All digital module
						5.5V	X	10KHz	X	✓
	I_{IDLE30}		0.12		mA	5.5V	X	10KHz	X	X
	I_{IDLE31}		0.11		mA	3.3V	X	10KHz	X	✓
	I_{IDLE32}		0.11		mA	3.3V	X	10KHz	X	X
Standby Current Power-down Mode	I_{PWD1}		-		μA	VDD	HXT	HIRC	LIRC	All digital module
						5.5V	✓	X	X	X
	I_{PWD2}		-		μA	5.5V	X	✓	X	X
	I_{PWD3}		-		μA	5.5V	X	X	✓	X
	I_{PWD4}		-		μA	3.3V	✓	X	X	X
	I_{PWD5}		-		μA	3.3V	X	✓	X	X

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS				
		MIN.	TYP.	MAX.	UNIT					
	I_{PWD6}		-		µA	3.3V	X	X	✓	X
Logic 0 Input Current (Quasi-bidirectional mode)	I_{IL}	-	-	-75	µA					
Input Leakage Current (input only)	I_{LK}	-	-	2	µA					
Logic 1 to 0 Transition Current (Quasi-bidirectional mode)	I_{TL}[3]	-	-	-660	µA	V_{DD} = 5.5V, V_{IN}<2.0V				
Internal Pull-High Resistor of /RESET[1]	R_{RST}	15	-	-	kΩ					
Input Low Voltage (TTL input)	V_{IL}	-0.3	-	0.2V_{DD}-0.1	V					
Input Low Voltage (Schmitt input)	V_{IL1}	-0.3		0.3V_{DD}	V					
Input Low Voltage (/RESET, XTAL in)	V_{IL2}	-0.3		0.15V_{DD}	V					
Input High Voltage (TTL input)	V_{IH}	0.2V_{DD}+0.9	-	V_{DD} +0.3	V					
Input High Voltage (Schmitt input, /RESET, XTAL in)	V_{IH1}	0.7V_{DD}	-	V_{DD} +0.3	V					
Hysteresis voltage of (Schmitt input)	V_{HY}	-	0.2V_{DD}	-	V					
Band-gap Voltage	V_{BG}	1.22	1.25	1.28	V	V_{DD} = 2.5 V ~ 5.5 V, T_A = 25°C				
Source Current (Quasi-bidirectional Mode)	I_{OH}	-360	-	-	µA	V_{DD} = 4.5V, V_S = 2.4V				
		-60	-	-	µA	V_{DD} = 2.7V, V_S = 2.2V				
		-50	-	-	µA	V_{DD} = 2.5V, V_S = 2.0V				
Source Current (Push-pull Mode)	I_{OH1}	-25	-	-	mA	V_{DD} = 4.5V, V_S = 2.4V				
		-4	-	-	mA	V_{DD} = 2.7V, V_S = 2.2V				
		-3	-	-	mA	V_{DD} = 2.5V, V_S = 2.0V				
Sink Current (Quasi-bidirectional and Push-pull Mode)	I_{OL}	16	-	-	mA	V_{DD} = 4.5V, V_S = 0.45V				
		10	-	-	mA	V_{DD} = 2.7V, V_S = 0.45V				
		9	-	-	mA	V_{DD} = 2.5V, V_S = 0.45V				

Note:

1. /RESET pin is a Schmitt trigger input.
2. Crystal Input is a CMOS input.
3. I/O pin can source a transition current when they are being externally driven from 1 to 0. In the condition of V_{DD}=5.5V, 5he transition current reaches its maximum value when V_{IN} approximates to 2V.

7.3 AC Electrical Characteristics



Note: Duty cycle is 50%.

SYMBOL	PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
t_{CHCX}	Clock High Time		10	-	-	nS
t_{CLCX}	Clock Low Time		10	-	-	nS
t_{CLCH}	Clock Rise Time		2	-	15	nS
t_{CHCL}	Clock Fall Time		2	-	15	nS

7.3.1 External 4~24MHz Crystal

PARAMETER	CONDITION	MIN.	TYP..	MAX.	UNIT
Operation Voltage V_{DD}	-	2.5	-	5.5	V
Temperature	-	-40	-	85	℃
Operating Current	12 MHz at V_{DD} = 5V	-	1	-	mA
Clock Frequency	External crystal	4		24	MHz

7.3.1.1 Typical Crystal Application Circuits

CRYSTAL	C1	C2	R
4MHz ~ 24 MHz	10~20 pF	10~20 pF	without

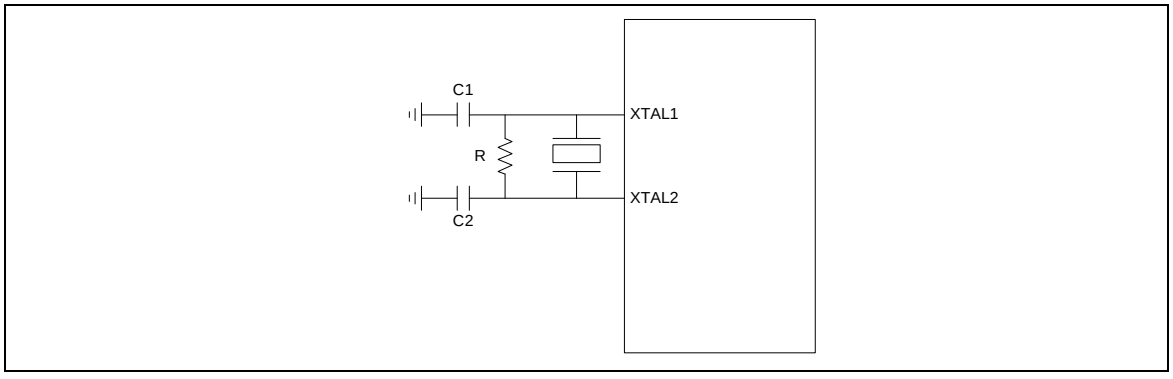


Figure 7–1 Typical Crystal Application Circuit

7.3.2 Internal 22.1184 MHz Oscillator

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Supply voltage	-	2.5	-	5.5	V
Frequency (After calibration)	-	-	22.1184	-	MHz
	+25°C; V_{DD} =5V	-1	-	+1	%
	-40 to +105°C; V_{DD}=2.5V~5.5V	-2	-	+2	%
Operation Current	V_{DD} =5V	-	500	-	uA

7.3.3 Internal 10 kHz Oscillator

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Supply voltage	-	2.5	-	5.5	V
Center Frequency	-	-	10	-	kHz
Calibrated Internal Oscillator Frequency	+25℃; V_{DD} =5V	-30	-	+30	%
	-40℃~+85℃; V_{DD}=2.5V~5.5V	-50	-	+50	%

7.4 Analog Characteristics

7.4.1 12-bit SARADC

PARAMETER	SYMBOL	CONDITON	MIN.	TYP.	MAX.	UNIT
Resolution	-		-	-	12	Bit
Differential nonlinearity error	DNL		-	-1~2	-1~4	LSB
Integral nonlinearity error	INL		-	±2	±4	LSB
Offset error	EO		-	±1	10	LSB
Full scale error	EG		-	1	1.005	LSB
Monotonic	-		Guaranteed			
ADC clock frequency	F_{ADC}	AVDD = 5V	-	-	16	MHz
		AVDD = 3V	-	-	8	
Sample rate	F_S	AVDD = 5V	-	-	800	ksps
		AVDD = 3V	-	-	400	
Sample time	T_S		-	8	-	Clock
Conversion time	T_{ADC}		-	12	-	Clock
Supply voltage	AV_{DD}		2.5	-	5.5	V
V_{REF} voltage	V_{REF}		2.0		AV_{DD}	
Supply current	I_{DDA}		-	1.5	-	mA
Reference current	I_{REF}		-	1	-	mA
Input voltage	VIN		0	-	V_{REF}	V
Capacitance	CIN		-	5	-	pF

7.4.2 LDO

PARAMETER	MIN.	TYP.	MAX.	UNIT	NOTE
Input Voltage V_{DD}	2.5		5.5	V	V_{DD} input voltage
Output Voltage	1.62	1.8	1.98	V	V_{DD}> 2.5 V
Operating Temperature	-40	25	105	℃	
Cbp	-	1	-	μF	R_{ESR} =1 Ω

Note:

1. It is recommended that a 10 uF or higher capacitor and a 100 nF bypass capacitor are connected between V_{DD} and the closest V_{SS} pin of the device.

2. To ensure power stability, a 1 μF or higher capacitor must be connected between LDO_CAP pin and the closest V_{SS} pin of the device.

7.4.3 Low Voltage Reset

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Operation Voltage	-	0	-	5.5	V
Quiescent Current	AV_{DD}=5.5V	-	1	5	μA
Operation Temperature	-	-40	25	105	℃
Threshold Voltage	-	1.6	2.0	2.4	V
Hysteresis	-	0	0	0	V

7.4.4 Brown-out Detector

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Operation Voltage	-	0	-	5.5	V
Temperature	-	-40	25	105	℃
Quiescent Current	AV_{DD}=5.5V	-	-	125	μA
Brown-out Voltage	BOD_VL[1:0]=11	4.2	4.4	4.6	V
	BOD_VL [1:0]=10	3.5	3.7	3.9	V
	BOD_VL [1:0]=01	2.6	2.7	2.8	V
	BOD_VL [1:0]=00	2.1	2.2	2.3	V
Hysteresis	-	30	-	150	mV

7.4.5 Power-On Reset (5V)

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Operation Temperature	-	-40	25	105	℃
Reset Voltage	V+	-	2	-	V
Quiescent Current	Vin>reset voltage	-	1	-	nA

7.4.6 Temperature Sensor

PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
Operation Voltage[1]		2.5	-	5.5	V
Operation Temperature		-40	-	105	℃
Current Consumption		6.4	-	10.5	μA
Gain			-1.76		mV/℃
Offset Voltage	Temp=0 ℃		720		mV

7.4.7 Comparator

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Operation Voltage AV_{DD}	-	2.5		5.5	V
Operation Current	V_{DD}=3.0 V	-	20	40	μA
Input Offset Voltage	-	-	5	15	mV
Output Swing	-	0.1	-	V_{DD}-0.1	V
Input Common Mode Range	-	0.1	-	V_{DD}-1.2	V
DC Gain	-	-	70	-	dB
Propagation Delay	VCM=1.2V and VDIFF=0.1V	-	200	-	ns
Comparison Voltage	20mV at VCM=1V 50mV at VCM=0.1V 50mV at VCM=V_{DD}-1.2 10mV for non-hysteresis	10	20	-	mV
Hysteresis	VCM=0.4V ~ V_{DD}-1.2V	-	±10	-	mV
Wake-up Time	CINP=1.3V CINN=1.2V	-	-	2	μs

7.4.8 OP Amplifier

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
AVDD	-	3.0	3.3	5.5	V
Input offset voltage	-	-	2	5	mV
Input offset average drift		-	-	1	uV/℃
Output swing	-	0.1	-	VDD-0.1	V
Input common mode range	-	0.1	-	VDD-1.2	V
DC gain	-	-	80	-	dB
Unity gain freq.	AVDD=5V	-	-	5	MHz
Phase margin		-	50°	-	°
PSRR+	AVDD=5V	-	90	-	dB
CMRR	AVDD=5V	-	90	-	dB
Slew rate	AVDD=5V, RLOAD=33K, CLOAD=50p	6.0	-	-	V/us
Wake up time		-	-	1	us
Quiescent current		-	-	2	mA

7.5 Flash DC Electrical Characteristics

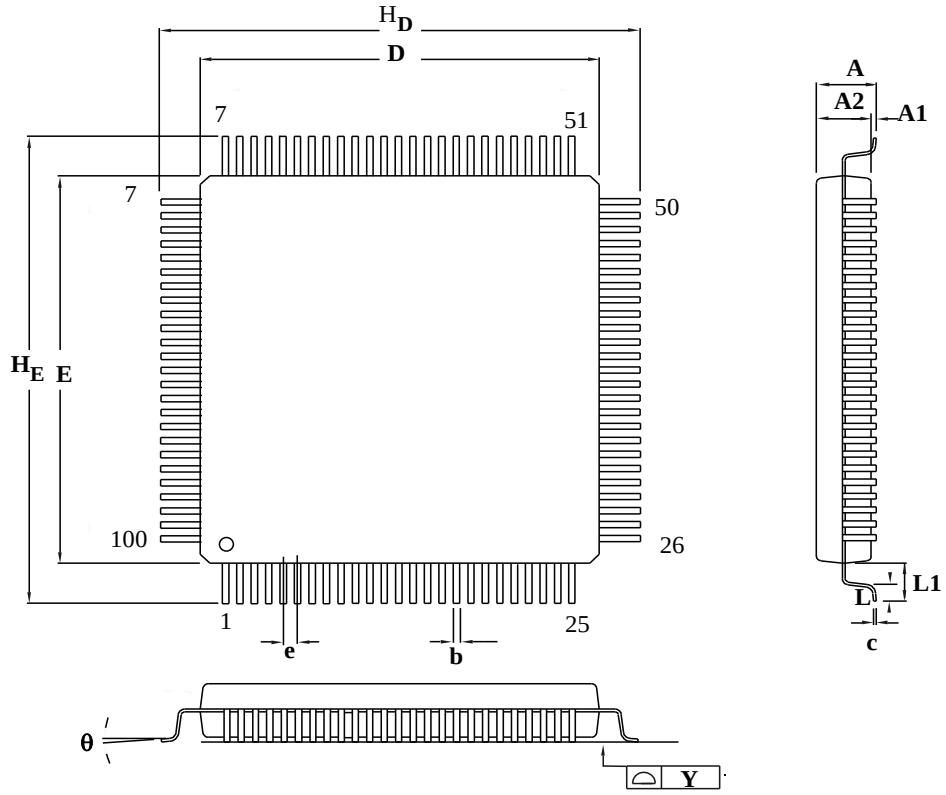
SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
V_{DD}	Supply Voltage		1.62	1.8	1.98	V[2]
N_{ENDUR}	Endurance		10000			cycles[1]
T_{RET}	Data Retention	At 25℃	100			year
T_{ERASE}	Page Erase Time			20		ms
T_{MER}	Mass Erase Time			40		ms
T_{PROG}	Program Time			40		μs
I_{DD1}	Read Current		-	0.15	0.5	mA/MHz
I_{DD2}	Program/Erase Current				7	mA

Note : This table is guaranteed by design, not test in production.

[1]Number of program/erase cycles.
[2] V_{DD} is source from chip LDO output voltage.

8 PACKAGE DIMENSIONS

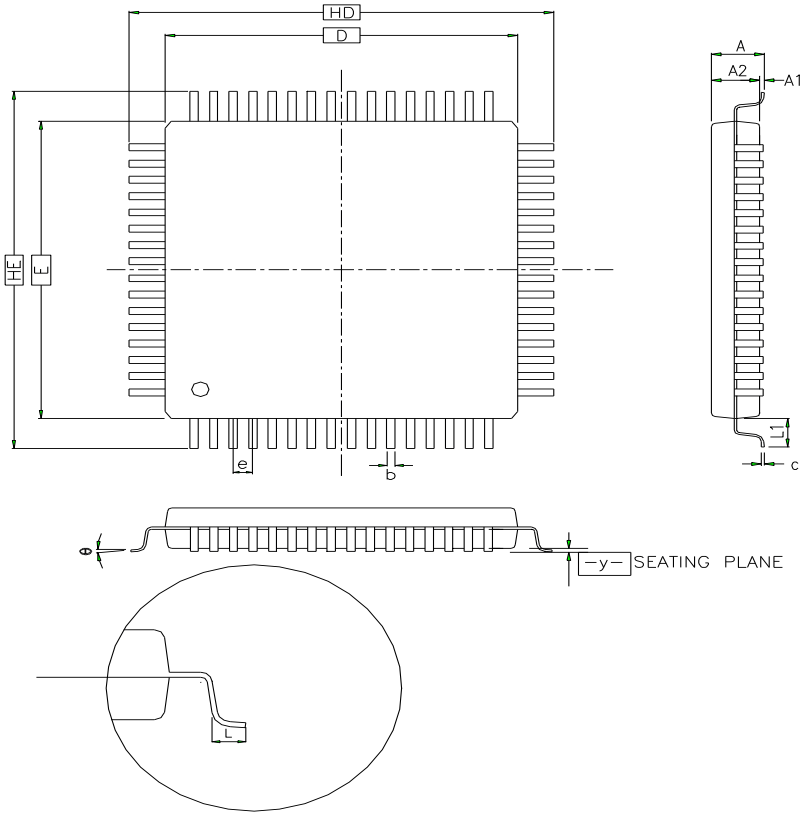
8.1 LQFP 100L (14x14x1.4 mm footprint 2.0mm)



Controlling Dimension : Millimeters

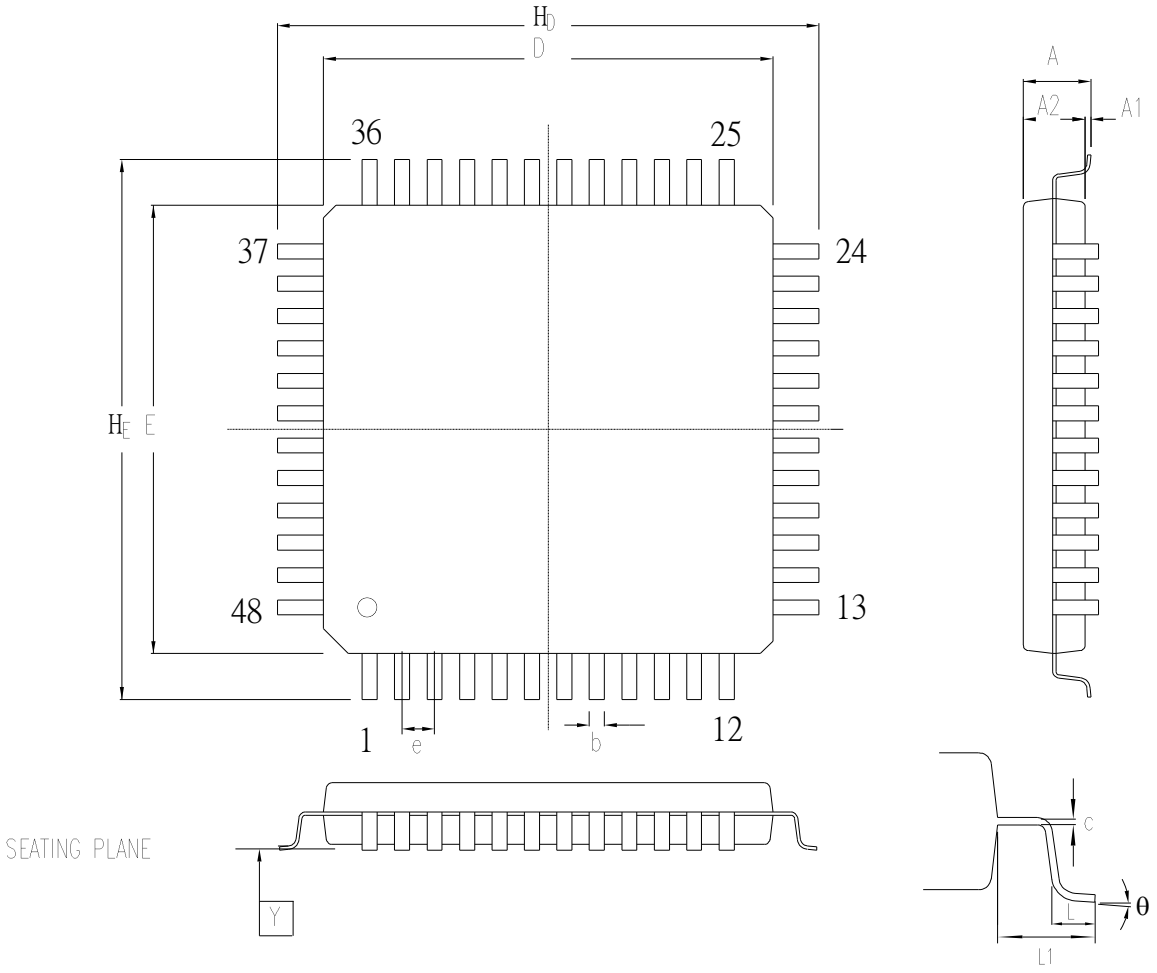
Symbol	Dimension in inch			Dimension in mm		
	Min	Nom	Max	Min	Nom	Max
A	—	—	0.063	—	—	1.60
A1	0.002	—	—	0.05	—	—
A2	0.053	0.055	0.057	1.35	1.40	1.45
b	0.007	0.009	0.011	0.17	0.22	0.27
c	0.004	0.006	0.008	0.10	0.15	0.20
D	0.547	0.551	0.556	13.90	14.00	14.10
E	0.547	0.551	0.556	13.90	14.00	14.10
e	—	0.020	—	—	0.50	—
H_D	0.622	0.630	0.638	15.80	16.00	16.20
H_E	0.622	0.630	0.638	15.80	16.00	16.20
L	0.018	0.024	0.030	0.45	0.60	0.75
L1	—	0.039	—	—	1.00	—
y	—	—	0.004	—	—	0.10
θ	0°	—	7°	0°	—	7°

8.2 LQFP 64L (10x10x1.4mm footprint 2.0 mm)



Symbol	Dimension in inch			Dimension in mm		
	Min	Nom	Max	Min	Nom	Max
A	—	—	0.063	—	—	1.60
A_1	0.002	—	0.006	0.05	—	0.15
A_2	0.053	0.055	0.057	1.35	1.40	1.45
b	0.007	0.008	0.011	0.17	0.20	0.27
c	0.004	—	0.008	0.09	—	0.20
D	—	0.393	—	—	10.00	—
E	—	0.393	—	—	10.00	—
e	—	0.020	—	—	0.50	—
H_D	—	0.472	—	—	12.00	—
H_E	—	0.472	—	—	12.00	—
L	0.018	0.024	0.030	0.45	0.60	0.75
L_1	—	0.039	—	—	1.00	—
y	—	0.004	—	—	0.10	—
θ	0°	3.5°	7°	0°	3.5°	7°

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



Controlling dimension : Millimeters

Symbol	Dimension in inch			Dimension in mm		
	Min	Nom	Max	Min	Nom	Max
A	—	—	—	—	—	—
A_1	0.002	0.004	0.006	0.05	0.10	0.15
A_2	0.053	0.055	0.057	1.35	1.40	1.45
b	0.006	0.008	0.010	0.15	0.20	0.25
c	0.004	0.006	0.008	0.10	0.15	0.20
D	0.272	0.276	0.280	6.90	7.00	7.10
E	0.272	0.276	0.280	6.90	7.00	7.10
e	0.014	0.020	0.026	0.35	0.50	0.65
H_D	0.350	0.354	0.358	8.90	9.00	9.10
H_E	0.350	0.354	0.358	8.90	9.00	9.10
L	0.018	0.024	0.030	0.45	0.60	0.75
L_1	—	0.039	—	—	1.00	—
Y	—	—	0.004	—	—	0.10
θ	0°	—	7°	0°	—	7°

9 REVISION HISTORY

Date	Revision	Description
2015.11.18	1.00	1. First version.
2016.03.09	1.10	1. Correct pin configuration and pin description 2. Modify the content of P0_MFP.
2017.09.26	1.20	1. Modify the pin description of P5.0 and P5.1: Add I2C functions 2. Modify the section of Hardware Divider (HDIV)
2018.04.11	1.30	1. Modify the section of Pin Description. 2. Modify the section of Pin Configuration.(64 pin/48 pin) 3. Modify the base address of MDUx as MDUx_BA = 0x401D_0000 + (0x0400 * x)
2018.12.24	1.40	1. In section of 4.3 Pin Description, add a note for pins ICE_CLK and ICE_DAT that the 2 pins should not be in floating state when MCU is in operation
2019.04.20	1.50	1. Add section 6.8.5.9 Asymmetric PWM Output 2. Modify the bit name of DIRF as DIRSTS in QEI section
2019.12.25	1.51	1. Remove the redundant data, operation temperature, from section 7.4.7 Comparator.
2022.04.18	1.52	1. Update Figure 6–4 Clock Generator Global View Diagram.
2022.11.08	1.53	1. Add compliance statement of International Environmental Regulations.

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.