NuMicro™ NUC200 Series
NUC200/220
DataSheet

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™ NUC200 Series 32-bit microcontrollers is embedded with the newest ARM® Cortex™-M0 core with a cost equivalent to traditional 8-bit MCU for industrial control and applications requiring rich communication interfaces. The NuMicro™ NUC200 Series includes NUC200 and NUC220 product lines.

The NuMicro™ NUC200 Advanced Line is embedded with the Cortex™-M0 core running up to 50 MHz and features 32K/64K/128K bytes flash, 8K/16K bytes embedded SRAM, and 4 Kbytes loader ROM for the ISP. It is also equipped with plenty of peripheral devices, such as Timers, Watchdog Timer, Window Watchdog Timer, RTC, PDMA with CRC calculation unit, UART, SPI, I^2C, I^2S, PWM Timer, GPIO, PS/2, Smart Card Host, 12-bit ADC, Analog Comparator, Low Voltage Reset Controller and Brown-out Detector.

The NuMicro™ NUC220 USB Line with USB 2.0 full-speed function is embedded with the Cortex™-M0 core running up to 50 MHz and features 32K/64K/128K bytes flash, 8K/16K bytes embedded SRAM, and 4 Kbytes loader ROM for the ISP. It is also equipped with plenty of peripheral devices, such as Timers, Watchdog Timer, Window Watchdog Timer, RTC, PDMA with CRC calculation unit, UART, SPI, I^2C, I^2S, PWM Timer, GPIO, PS/2, USB 2.0 FS Device, Smart Card Host, 12-bit ADC, Analog Comparator, Low Voltage Reset Controller and Brown-out Detector.

Product Line	UART	SPI	I^2C	USB	LIN	CAN	PS/2	I^2S	SC
NUC200	•	•	•				•	•	•
NUC220	•	•	•	•			•	•	•

Table 1-1 Connectivity Support Table

2 FEATURES

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

2.1 NuMicro™ NUC200 Features – Advanced Line

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

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

- Watchdog Timer
  - Multiple clock sources
  - 8 selectable time-out period from 1.6 ms ~ 26.0 sec (depending on clock source)
  - Wake-up from Power-down or Idle mode
  - Interrupt or reset selectable on watchdog time-out
- Window Watchdog Timer
  - 6-bit down counter with 11-bit prescale for wide range window selected
- RTC
  - Supports software compensation by setting frequency compensate register (FCR)
  - Supports RTC counter (second, minute, hour) and calendar counter (day, month, year)
  - Supports Alarm registers (second, minute, hour, day, month, year)
  - Selectable 12-hour or 24-hour mode
  - Automatic leap year recognition
  - Supports periodic time tick interrupt with 8 period options 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second
  - Supports battery power pin (V_{BAT})
  - Supports wake-up function
- PWM/Capture
  - Up to four built-in 16-bit PWM generators providing eight PWM outputs or four complementary paired PWM outputs
  - Each PWM generator equipped with one clock source selector, one clock divider, one 8-bit prescaler and one Dead-Zone generator for complementary paired PWM
  - Up to eight 16-bit digital capture timers (shared with PWM timers) providing eight rising/falling capture inputs
  - Supports Capture interrupt
- UART
  - Up to three UART controllers
  - UART ports with flow control (TXD, RXD, nCTS and nRTS)
  - UART0 with 64-byte FIFO is for high speed
  - UART1/2(optional) with 16-byte FIFO for standard device
  - Supports IrDA (SIR) and LIN function
  - Supports RS-485 9-bit mode and direction control
  - Programmable baud-rate generator up to 1/16 system clock
  - Supports PDMA mode
- SPI
  - Up to four sets of SPI controllers
  - The maximum SPI clock rate of Master can up to 36 MHz (chip working at 5V)
  - The maximum SPI clock rate of Slave can up to 18 MHz (chip working at 5V)
  - 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
  - Two slave/device select lines in Master mode, and one slave/device select line in Slave mode
  - Supports Byte Suspend mode in 32-bit transmission

- Supports PDMA mode
- Supports three wire, no slave select signal, bi-direction interface

- I^2C
  - Up to two sets of I^2C device
  - Master/Slave mode
  - Bidirectional data transfer between masters and slaves
  - Multi-master bus (no central master)
  - Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
  - Serial clock synchronization allowing devices with different bit rates to communicate via one serial bus
  - Serial clock synchronization used as a handshake mechanism to suspend and resume serial transfer
  - Programmable clocks allowing for versatile rate control
  - Supports multiple address recognition (four slave address with mask option)
  - Supports wake-up function
- I^2S
  - Interface with external audio CODEC
  - Operate as either Master or Slave mode
  - Capable of handling 8-, 16-, 24- and 32-bit word sizes
  - Supports mono and stereo audio data
  - Supports I^2S and MSB justified data format
  - Provides two 8 word FIFO data buffers, one for transmitting and the other for receiving
  - Generates interrupt requests when buffer levels cross a programmable boundary
  - Supports two DMA requests, one for transmitting and the other for receiving
- PS/2 Device
  - Host communication inhibit and request to send detection
  - Reception frame error detection
  - Programmable 1 to 16 bytes transmit buffer to reduce CPU intervention
  - Double buffer for data reception
  - Software override bus
- ADC
  - 12-bit SAR ADC with 760 kSPS
  - Up to 8-ch single-end input or 4-ch differential input
  - Single scan/single cycle scan/continuous scan
  - Each channel with individual result register
  - Scan on enabled channels
  - Threshold voltage detection
  - Conversion started by software programming or external input
  - Supports PDMA mode
- Analog Comparator
  - Up to two analog comparators
  - External input or internal Band-gap voltage selectable at negative node
  - Interrupt when compare result change
  - Supports Power-down wake-up
- Smart Card Host (SC)
  - Compliant to ISO-7816-3 T=0, T=1
  - Supports up to three ISO-7816-3 ports
  - Separate receive / transmit 4 bytes entry FIFO for data payloads
  - Programmable transmission clock frequency
  - Programmable receiver buffer trigger level

  - Programmable guard time selection (11 ETU ~ 266 ETU)
  - One 24-bit and two 8-bit time-out counters for Answer to Request (ATR) and waiting times processing
  - Supports auto inverse convention function
  - Supports transmitter and receiver error retry and error limit function
  - Supports hardware activation sequence process
  - Supports hardware warm reset sequence process
  - Supports hardware deactivation sequence process
  - Supports hardware auto deactivation sequence when detecting the card removal
- 96-bit unique ID (UID)
- One built-in temperature sensor with 1℃ resolution
- Brown-out Detector
  - With 4 levels: 4.4 V/3.7 V/2.7 V/2.2 V
  - Supports Brown-out Interrupt and Reset option
- Low Voltage Reset
  - Threshold voltage level: 2.0 V
- Operating Temperature: -40℃ ~ 85℃
- Packages:
  - All Green package (RoHS)
  - LQFP 100-pin / 64-pin / 48-pin

2.2 NuMicro™ NUC220 Features – USB Line

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

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

- Watchdog Timer
  - Multiple clock sources
  - 8 selectable time-out period from 1.6 ms ~ 26.0 sec (depending on clock source)
  - Wake-up from Power-down or Idle mode
  - Interrupt or reset selectable on watchdog time-out
- Window Watchdog Timer
  - 6-bit down counter with 11-bit prescale for wide range window selected
- RTC
  - Supports software compensation by setting frequency compensate register (FCR)
  - Supports RTC counter (second, minute, hour) and calendar counter (day, month, year)
  - Supports Alarm registers (second, minute, hour, day, month, year)
  - Selectable 12-hour or 24-hour mode
  - Automatic leap year recognition
  - Supports periodic time tick interrupt with 8 period options 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second
  - Supports battery power pin (V_{BAT})
  - Supports wake-up function
- PWM/Capture
  - Up to four built-in 16-bit PWM generators providing eight PWM outputs or four complementary paired PWM outputs
  - Each PWM generator equipped with one clock source selector, one clock divider, one 8-bit prescaler and one Dead-Zone generator for complementary paired PWM
  - Up to eight 16-bit digital capture timers (shared with PWM timers) providing eight rising/falling capture inputs
  - Supports Capture interrupt
- UART
  - Up to three UART controllers
  - UART ports with flow control (TXD, RXD, nCTS and nRTS)
  - UART0 with 64-byte FIFO is for high speed
  - UART1/2(optional) with 16-byte FIFO for standard device
  - Supports IrDA (SIR) and LIN function
  - Supports RS-485 9-bit mode and direction control
  - Programmable baud-rate generator up to 1/16 system clock
  - Supports PDMA mode
- SPI
  - Up to four sets of SPI controllers
  - The maximum SPI clock rate of Master can up to 36 MHz (chip working at 5V)
  - The maximum SPI clock rate of Slave can up to 18 MHz (chip working at 5V)
  - 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
  - Two slave/device select lines in Master mode, and one slave/device select line in Slave mode
  - Supports Byte Suspend mode in 32-bit transmission
  - Supports PDMA mode
  - Supports three wire, no slave select signal, bi-direction interface

- I^2C
  - Up to two sets of I^2C device
  - Master/Slave mode
  - Bidirectional data transfer between masters and slaves
  - Multi-master bus (no central master)
  - Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
  - Serial clock synchronization allowing devices with different bit rates to communicate via one serial bus
  - Serial clock synchronization used as a handshake mechanism to suspend and resume serial transfer
  - Programmable clocks allowing for versatile rate control
  - Supports multiple address recognition (four slave address with mask option)
  - Supports wake-up function
- I^2S
  - Interface with external audio CODEC
  - Operate as either Master or Slave mode
  - Capable of handling 8-, 16-, 24- and 32-bit word sizes
  - Supports mono and stereo audio data
  - Supports I^2S and MSB justified data format
  - Provides two 8 word FIFO data buffers, one for transmitting and the other for receiving
  - Generates interrupt requests when buffer levels cross a programmable boundary
  - Supports two DMA requests, one for transmitting and the other for receiving
- PS/2 Device
  - Host communication inhibit and request to send detection
  - Reception frame error detection
  - Programmable 1 to 16 bytes transmit buffer to reduce CPU intervention
  - Double buffer for data reception
  - Software override bus
- USB 2.0 Full-Speed Device
  - One set of USB 2.0 FS Device 12 Mbps
  - On-chip USB Transceiver
  - Provides 1 interrupt source with 4 interrupt events
  - Supports Control, Bulk In/Out, Interrupt and Isochronous transfers
  - Auto suspend function when no bus signaling for 3 ms
  - Provides 6 programmable endpoints
  - Includes 512 Bytes internal SRAM as USB buffer
  - Provides remote wake-up capability
- ADC
  - 12-bit SAR ADC with 760 kSPS
  - Up to 8-ch single-end input or 4-ch differential input
  - Single scan/single cycle scan/continuous scan
  - Each channel with individual result register
  - Scan on enabled channels
  - Threshold voltage detection
  - Conversion started by software programming or external input
  - Supports PDMA mode
- Analog Comparator
  - Up to two analog comparators
  - External input or internal Band-gap voltage selectable at negative node

  - Interrupt when compare result change
  - Supports Power-down wake-up
- Smart Card Host (SC)
  - Compliant to ISO-7816-3 T=0, T=1
  - Supports up to three ISO-7816-3 ports
  - Separate receive / transmit 4 bytes entry FIFO for data payloads
  - Programmable transmission clock frequency
  - Programmable receiver buffer trigger level
  - Programmable guard time selection (11 ETU ~ 266 ETU)
  - One 24-bit and two 8-bit time-out counters for Answer to Request (ATR) and waiting times processing
  - Supports auto inverse convention function
  - Supports transmitter and receiver error retry and error limit function
  - Supports hardware activation sequence process
  - Supports hardware warm reset sequence process
  - Supports hardware deactivation sequence process
  - Supports hardware auto deactivation sequence when detecting the card removal
- 96-bit unique ID (UID)
- One built-in temperature sensor with 1℃ resolution
- Brown-out Detector
  - With 4 levels: 4.4 V/3.7 V/2.7 V/2.2 V
  - Supports Brown-out Interrupt and Reset option
- Low Voltage Reset
  - Threshold voltage level: 2.0 V
- Operating Temperature: -40℃ ~ 85℃
- Packages:
  - All Green package (RoHS)
  - LQFP 100-pin / 64-pin / 48-pin

3 PARTS INFORMATION LIST AND PIN CONFIGURATION

3.1 NuMicro™ NUC200/220xxxAN Selection Guide

3.1.1 NuMicro™ NUC200 Advance Line Selection Guide

Part number	APROM	RAM	Data Flash	ISP Loader ROM	I/O	Timer	Connectivity						I^2S	SC	Comp.	PWM	ADC	RTC	ISP ICP IAP	Package
							UART	SPI	I^2C	USB	LIN	CAN								
NUC200LC2AN	32 KB	8 KB	4 KB	4 KB	up to 35	4x32-bit	2	1	2	-	-	-	1	2	1	6	7x12-bit	v	v	LQFP48
NUC200LD2AN	64 KB	8 KB	4KB	4 KB	up to 35	4x32-bit	2	1	2	-	-	-	1	2	1	6	7x12-bit	v	v	LQFP48
NUC200LE3AN	128 KB	16 KB	Definable	4 KB	up to 35	4x32-bit	2	1	2	-	-	-	1	2	1	6	7x12-bit	v	v	LQFP48
NUC200SC2AN	32 KB	8 KB	4 KB	4 KB	up to 49	4x32-bit	3	2	2	-	-	-	1	2	2	6	7x12-bit	v	v	LQFP64
NUC200SD2AN	64 KB	8 KB	4KB	4 KB	up to 49	4x32-bit	3	2	2	-	-	-	1	2	2	6	7x12-bit	v	v	LQFP64
NUC200SE3AN	128 KB	16 KB	Definable	4 KB	up to 49	4x32-bit	3	2	2	-	-	-	1	2	2	6	7x12-bit	v	v	LQFP64
NUC200VE3AN	128 KB	16 KB	Definable	4 KB	up to 83	4x32-bit	3	4	2	-	-	-	1	3	2	8	8x12-bit	v	v	LQFP100

3.1.2 NuMicro™ NUC220 USB Line Selection Guide

Part number	APROM	RAM	Data Flash	ISP Loader ROM	I/O	Timer	Connectivity						I^2S	SC	Comp.	PWM	ADC	RTC	ISP ICP IAP	Package
							UART	SPI	I^2C	USB	LIN	CAN								
NUC220LC2AN	32 KB	8 KB	4 KB	4 KB	up to 31	4x32-bit	2	1	2	1	-	-	1	2	1	4	7x12-bit	v	v	LQFP48
NUC220LD2AN	64 KB	8 KB	4 KB	4 KB	up to 31	4x32-bit	2	1	2	1	-	-	1	2	1	4	7x12-bit	v	v	LQFP48
NUC220LE3AN	128 KB	16 KB	Definable	4 KB	up to 31	4x32-bit	2	1	2	1	-	-	1	2	1	4	7x12-bit	v	v	LQFP48
NUC220SC2AN	32 KB	8 KB	4 KB	4 KB	up to 45	4x32-bit	2	2	2	1	-	-	1	2	2	6	7x12-bit	v	v	LQFP64
NUC220SD2AN	64 KB	8 KB	8 KB	4 KB	up to 45	4x32-bit	2	2	2	1	-	-	1	2	2	6	7x12-bit	v	v	LQFP64
NUC220SE3AN	128 KB	16 KB	Definable	4 KB	up to 45	4x32-bit	2	2	2	1	-	-	1	2	2	6	7x12-bit	v	v	LQFP64
NUC220VE3AN	128 KB	16 KB	Definable	4 KB	up to 79	4x32-bit	3	4	2	1	-	-	1	3	2	8	8x12-bit	v	v	LQFP100

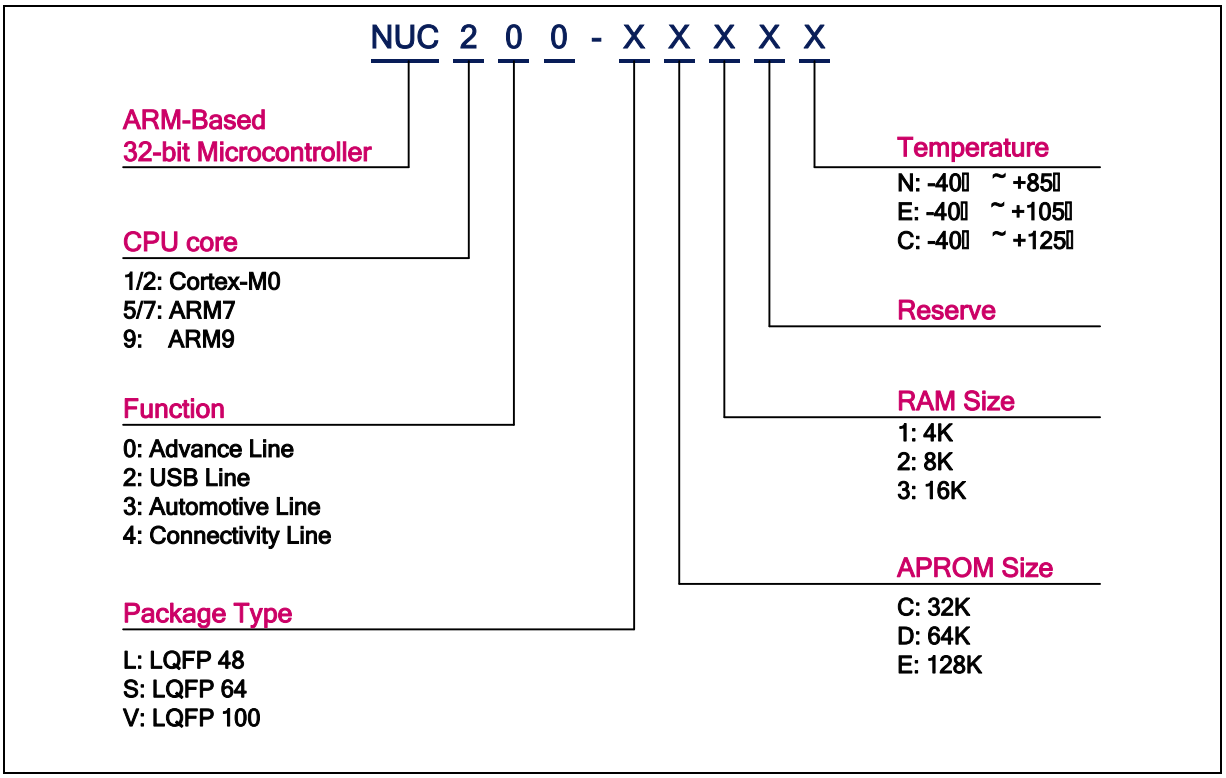


Figure 3-1 NuMicro™ NUC200 Series Selection Code

3.2 Pin Configuration

3.2.1 NuMicro™ NUC200 Pin Diagram

3.2.1.1 *NuMicro™ NUC200VxxAN LQFP 100-pin*

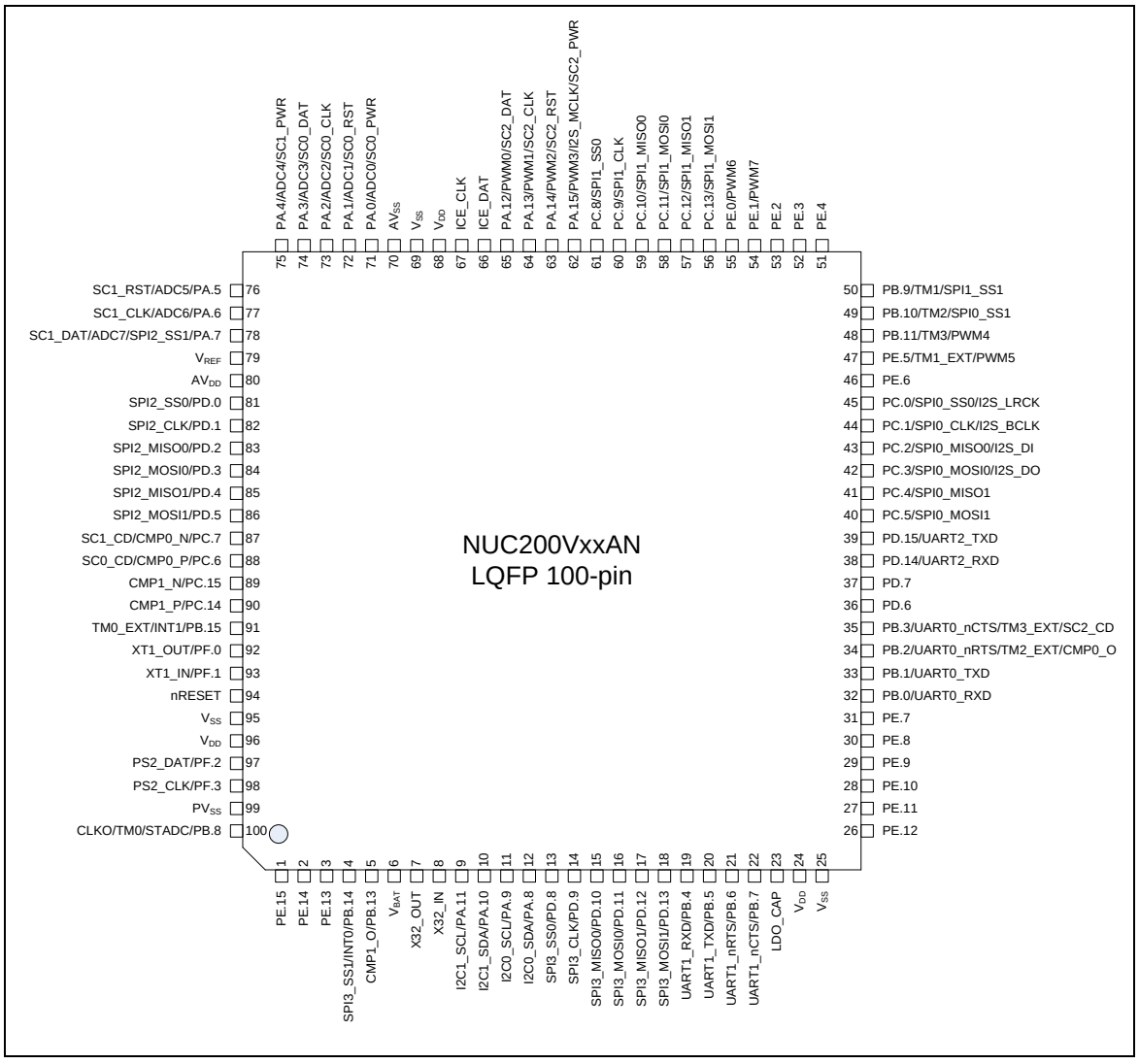


Figure 3-2 NuMicro™ NUC200VxxAN LQFP 100-pin Diagram

3.2.1.2 *NuMicro™ NUC200RxxAN LQFP 64-pin*

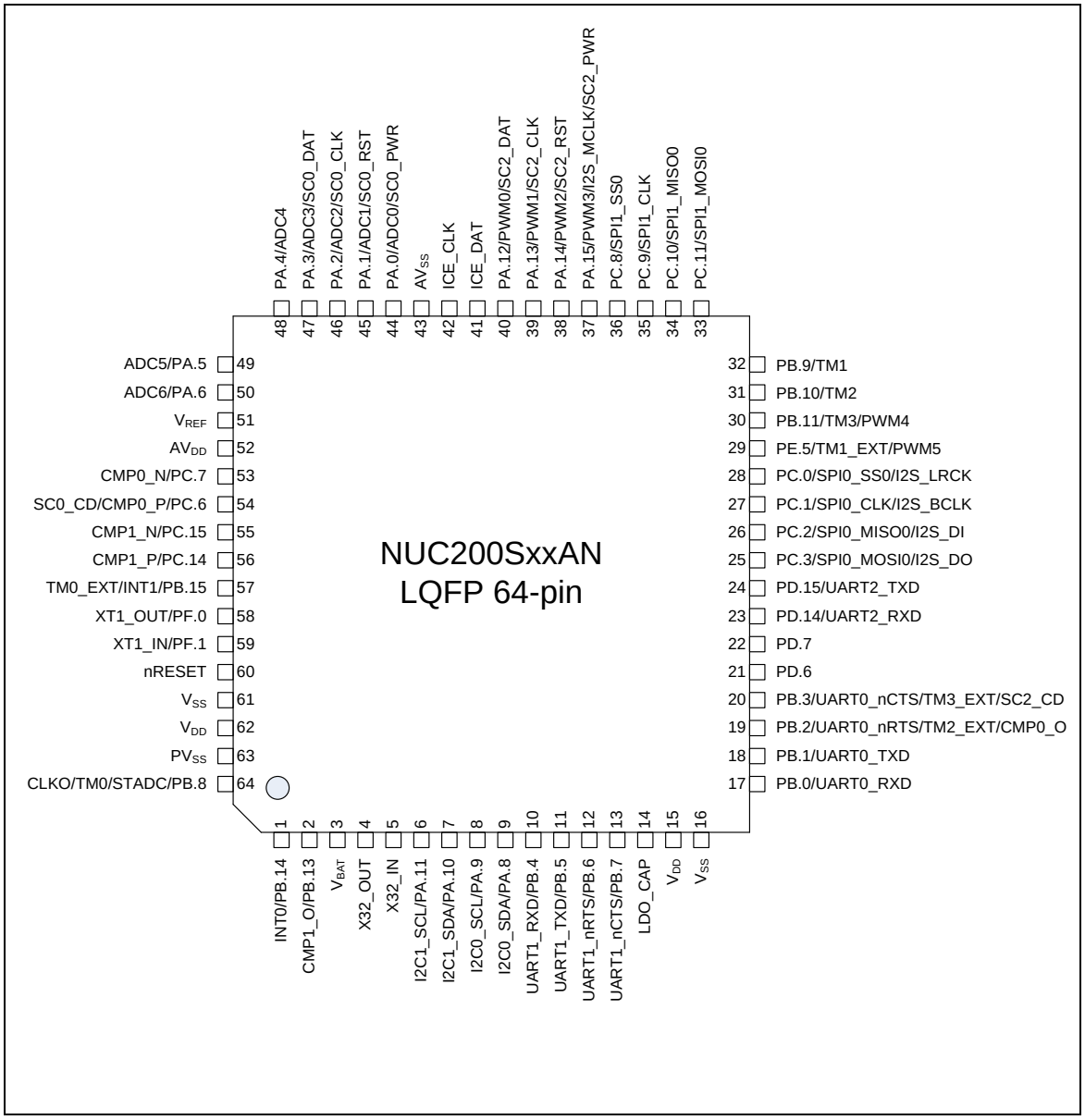


Figure 3-3 NuMicro™ NUC200SxxAN LQFP 64-pin Diagram

3.2.1.3 *NuMicro™ NUC200LxxAN LQFP 48-pin*

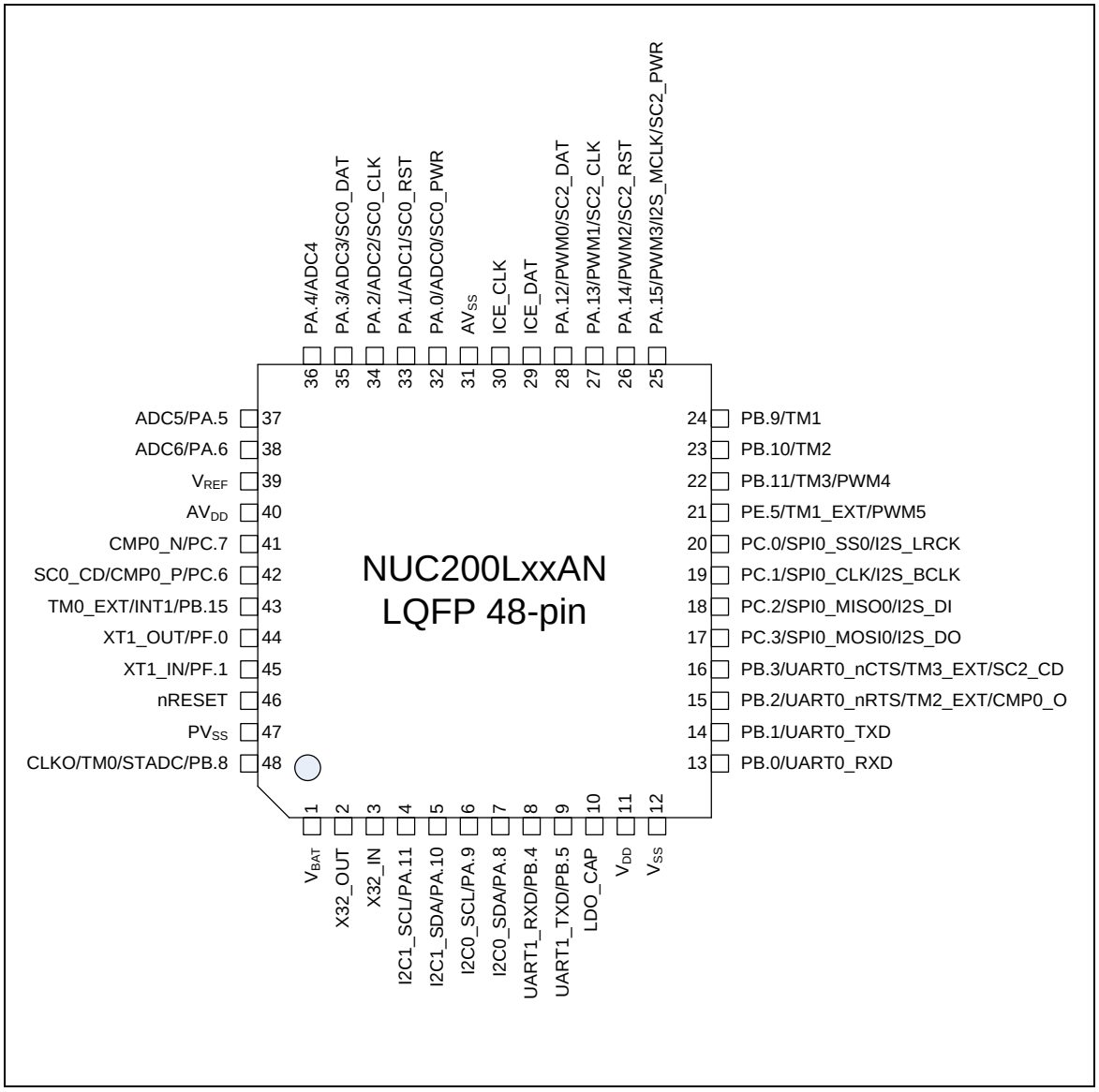


Figure 3-4 NuMicro™ NUC200LxxAN LQFP 48-pin Diagram

3.2.2 NuMicro™ NUC220 Pin Diagram

3.2.2.1 *NuMicro™ NUC220VxxAN LQFP 100-pin*

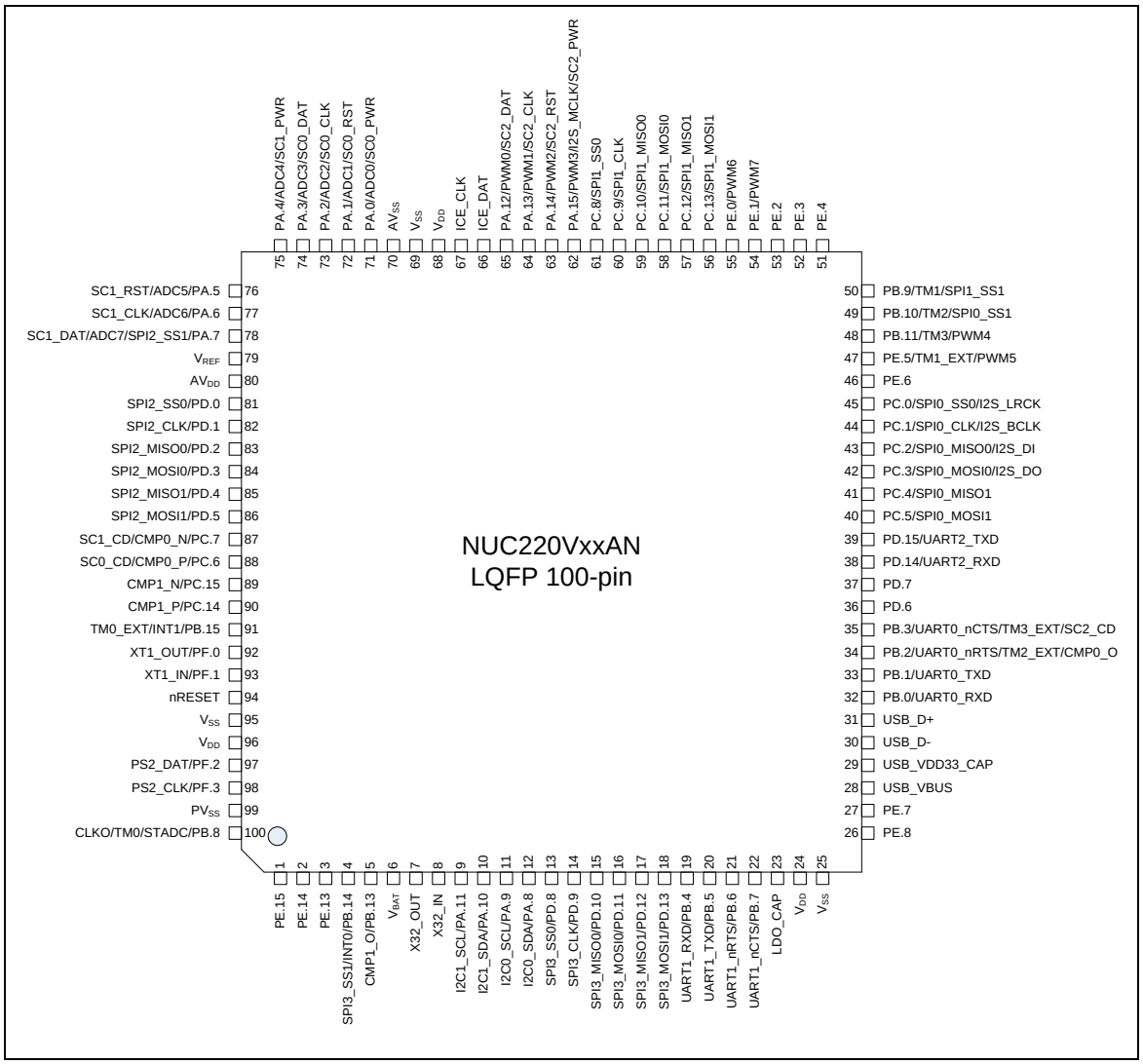


Figure 3-5 NuMicro™ NUC220VxxAN LQFP 100-pin Diagram

3.2.2.2 *NuMicro™ NUC220RxxAN LQFP 64-pin*

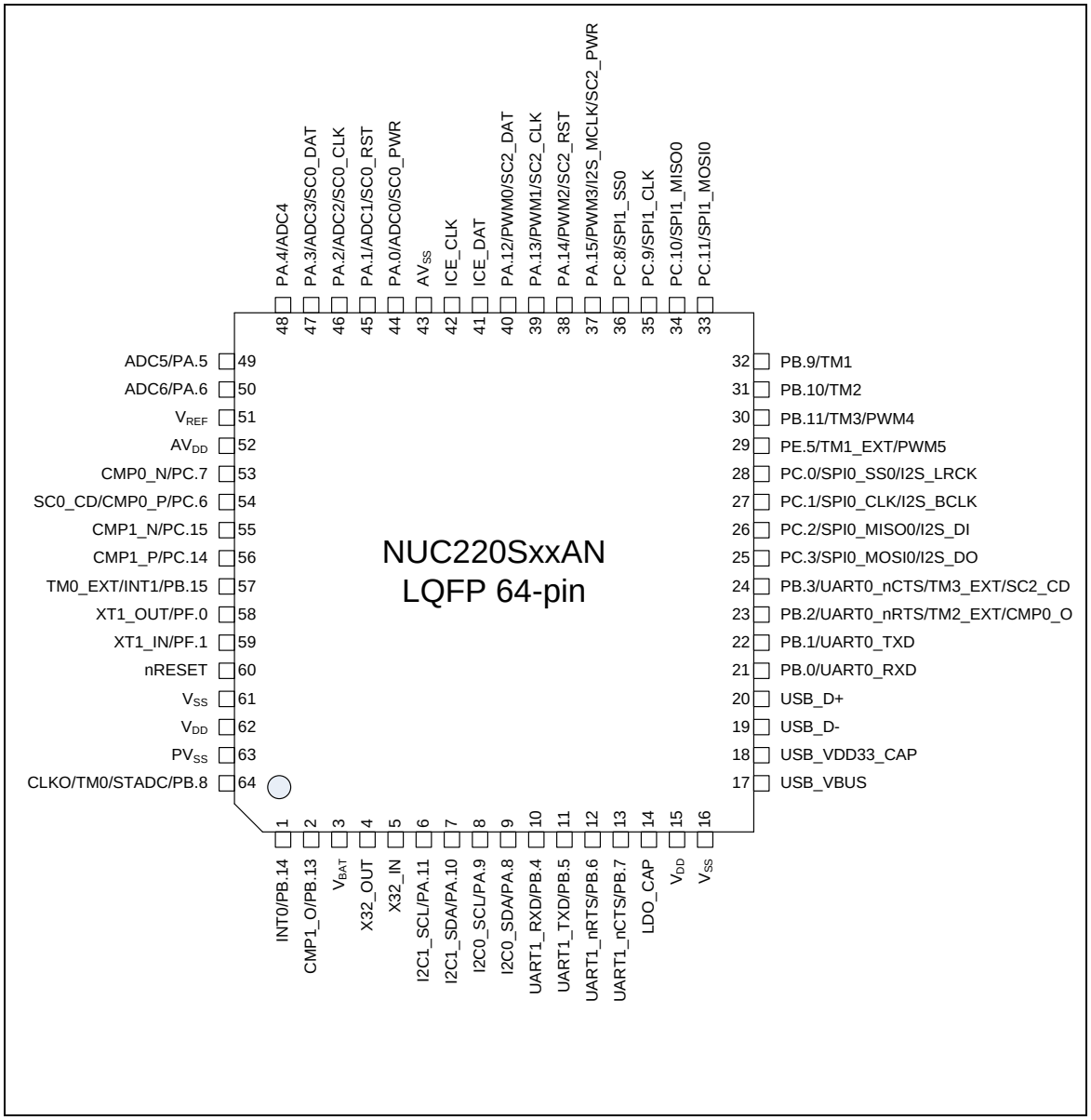


Figure 3-6 NuMicro™ NUC220SxxAN LQFP 64-pin Diagram

3.2.2.3 *NuMicro™ NUC220LxxAN LQFP 48-pin*

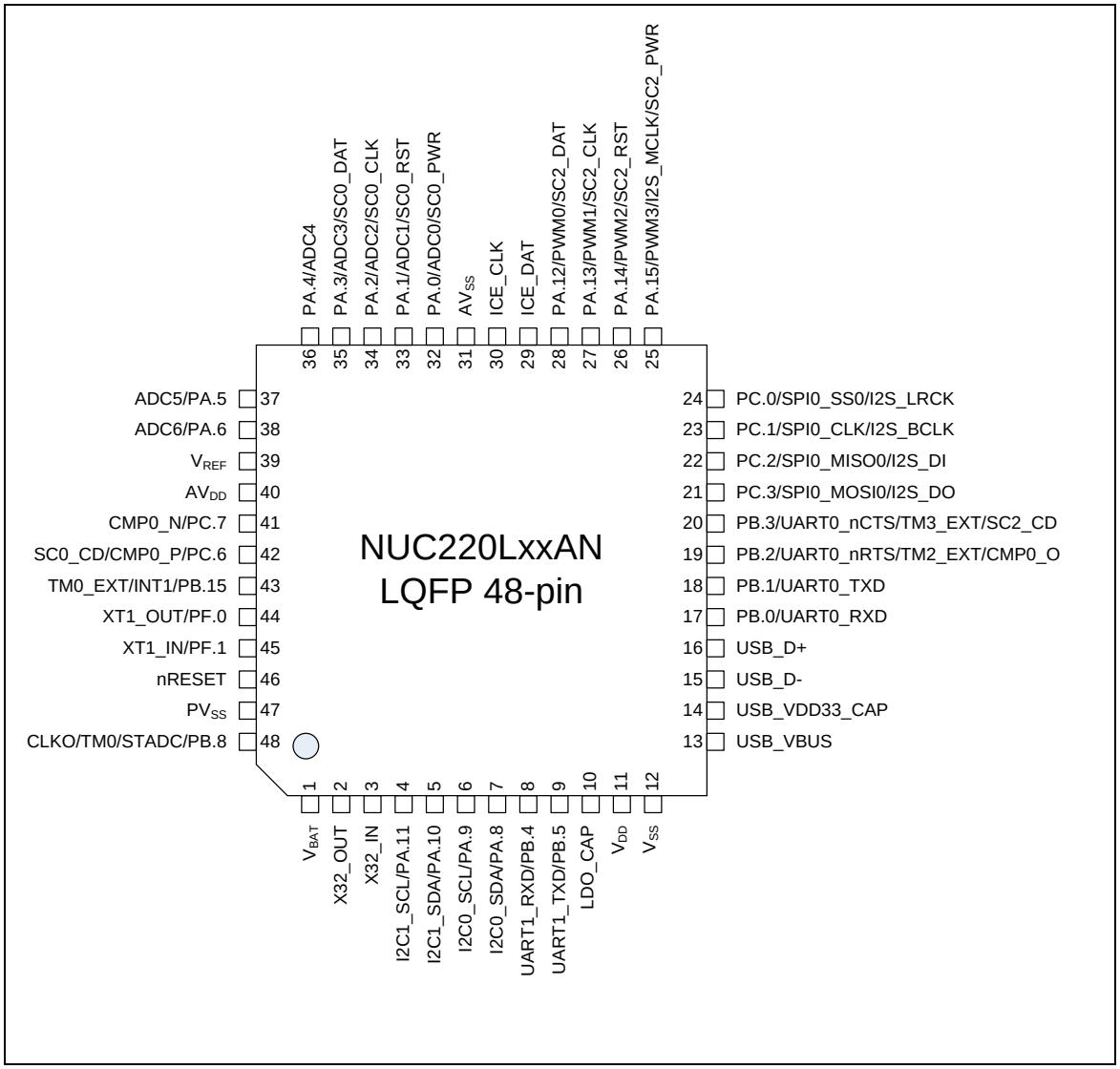


Figure 3-7 NuMicro™ NUC220LxxAN LQFP 48-pin Diagram

3.3 Pin Description

3.3.1 NuMicro™ NUC200 Pin Description

Pin No.			Pin Name	Pin Type	Description
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
1			PE.15	**I/O**	General purpose digital I/O pin.
2			PE.14	**I/O**	General purpose digital I/O pin.
3			PE.13	**I/O**	General purpose digital I/O pin.
4	1		PB.14	**I/O**	General purpose digital I/O pin.
			INT0	**I**	External interrupt0 input pin.
			SPI3_SS1	**I/O**	2nd SPI3 slave select pin.
5	2		PB.13	**I/O**	General purpose digital I/O pin.
			CMP1_O	**O**	Comparator1 output pin.
6	3	1	V_{BAT}	**P**	Power supply by batteries for RTC.
7	4	2	X32_OUT	**O**	External 32.768 kHz (low speed) crystal output pin.
8	5	3	X32_IN	**I**	External 32.768 kHz (low speed) crystal input pin.
9	6	4	PA.11	**I/O**	General purpose digital I/O pin.
			I2C1_SCL	**I/O**	I^2C1 clock pin.
10	7	5	PA.10	**I/O**	General purpose digital I/O pin.
			I2C1_SDA	**I/O**	I^2C1 data input/output pin.
11	8	6	PA.9	**I/O**	General purpose digital I/O pin.
			I2C0_SCL	**I/O**	I^2C0 clock pin.
12	9	7	PA.8	**I/O**	General purpose digital I/O pin.
			I2C0_SDA	**I/O**	I^2C0 data input/output pin.
13			PD.8	**I/O**	General purpose digital I/O pin.
			SPI3_SS0	**I/O**	1st SPI3 slave select pin.
14			PD.9	**I/O**	General purpose digital I/O pin.
			SPI3_CLK	**I/O**	SPI3 serial clock pin.
15			PD.10	**I/O**	General purpose digital I/O pin.
			SPI3_MISO0	**I/O**	1st SPI3 MISO (Master In, Slave Out) pin.
16			PD.11	**I/O**	General purpose digital I/O pin.
			SPI3_MOSI0	**I/O**	1st SPI3 MOSI (Master Out, Slave In) pin.

Pin No.			Pin Name	Pin Type	Description
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
17			PD.12	**I/O**	General purpose digital I/O pin.
			SPI3_MISO1	**I/O**	2nd SPI3 MISO (Master In, Slave Out) pin.
18			PD.13	**I/O**	General purpose digital I/O pin.
			SPI3_MOSI1	**I/O**	2nd SPI3 MOSI (Master Out, Slave In) pin.
19	10	8	PB.4	**I/O**	General purpose digital I/O pin.
			UART1_RXD	**I**	Data receiver input pin for UART1.
20	11	9	PB.5	**I/O**	General purpose digital I/O pin.
			UART1_TXD	**O**	Data transmitter output pin for UART1.
21	12		PB.6	**I/O**	General purpose digital I/O pin.
			UART1_nRTS	**O**	Request to Send output pin for UART1.
22	13		PB.7	**I/O**	General purpose digital I/O pin.
			UART1_nCTS	**I**	Clear to Send input pin for UART1.
23	14	10	LDO_CAP	**P**	LDO output pin.
24	15	11	V_{DD}	**P**	Power supply for I/O ports and LDO source for internal PLL and digital circuit.
25	16	12	V_{SS}	**P**	Ground pin for digital circuit.
26			PE.12	**I/O**	General purpose digital I/O pin.
27			PE.11	**I/O**	General purpose digital I/O pin.
28			PE.10	**I/O**	General purpose digital I/O pin.
29			PE.9	**I/O**	General purpose digital I/O pin.
30			PE.8	**I/O**	General purpose digital I/O pin.
31			PE.7	**I/O**	General purpose digital I/O pin.
32	17	13	PB.0	**I/O**	General purpose digital I/O pin.
			UART0_RXD	**I**	Data receiver input pin for UART0.
33	18	14	PB.1	**I/O**	General purpose digital I/O pin.
			UART0_TXD	**O**	Data transmitter output pin for UART0.
34	19	15	PB.2	**I/O**	General purpose digital I/O pin.
			UART0_nRTS	**O**	Request to Send output pin for UART0.
			TM2_EXT	**I**	Timer2 external capture input pin.
			CMP0_O	**O**	Comparator0 output pin.
35	20	16	PB.3	**I/O**	General purpose digital I/O pin.
			UART0_nCTS	**I**	Clear to Send input pin for UART0.

Pin No.			Pin Name	Pin Type	Description
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
			TM3_EXT	I	Timer3 external capture input pin.
			SC2_CD	I	SmartCard2 card detect pin.
36	21		PD.6	I/O	General purpose digital I/O pin.
37	22		PD.7	I/O	General purpose digital I/O pin.
38	23		PD.14	I/O	General purpose digital I/O pin.
			UART2_RXD	I	Data receiver input pin for UART2.
39	24		PD.15	I/O	General purpose digital I/O pin.
			UART2_TXD	O	Data transmitter output pin for UART2.
40			PC.5	I/O	General purpose digital I/O pin.
			SPI0_MOSI1	I/O	2nd SPI0 MOSI (Master Out, Slave In) pin.
41			PC.4	I/O	General purpose digital I/O pin.
			SPI0_MISO1	I/O	2nd SPI0 MISO (Master In, Slave Out) pin.
42	25	17	PC.3	I/O	General purpose digital I/O pin.
			SPI0_MOSI0	I/O	1st SPI0 MOSI (Master Out, Slave In) pin.
			I2S_DO	O	I2S data output.
43	26	18	PC.2	I/O	General purpose digital I/O pin.
			SPI0_MISO0	I/O	1st SPI0 MISO (Master In, Slave Out) pin.
			I2S_DI	I	I2S data input.
44	27	19	PC.1	I/O	General purpose digital I/O pin.
			SPI0_CLK	I/O	SPI0 serial clock pin.
			I2S_BCLK	I/O	I2S bit clock pin.
45	28	20	PC.0	I/O	General purpose digital I/O pin.
			SPI0_SS0	I/O	1st SPI0 slave select pin.
			I2S_LRCK	I/O	I2S left right channel clock.
46			PE.6	I/O	General purpose digital I/O pin.
47	29	21	PE.5	I/O	General purpose digital I/O pin.
			PWM5	I/O	PWM5 output/Capture input.
			TM1_EXT	I	Timer1 external capture input pin.
48	30	22	PB.11	I/O	General purpose digital I/O pin.
			TM3	I/O	Timer3 event counter input / toggle output.
			PWM4	I/O	PWM4 output/Capture input.

Pin No.			Pin Name	Pin Type	Description
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
49	31	23	PB.10	I/O	General purpose digital I/O pin.
			TM2	I/O	Timer2 event counter input / toggle output.
			SPI0_SS1	I/O	2nd SPI0 slave select pin.
50	32	24	PB.9	I/O	General purpose digital I/O pin.
			TM1	I/O	Timer1 event counter input / toggle output.
			SPI1_SS1	I/O	2nd SPI1 slave select pin.
51			PE.4	I/O	General purpose digital I/O pin.
52			PE.3	I/O	General purpose digital I/O pin.
53			PE.2	I/O	General purpose digital I/O pin.
54			PE.1	I/O	General purpose digital I/O pin.
			PWM7	I/O	PWM7 output/Capture input.
55			PE.0	I/O	General purpose digital I/O pin.
			PWM6	I/O	PWM6 output/Capture input.
56			PC.13	I/O	General purpose digital I/O pin.
			SPI1_MOSI1	I/O	2nd SPI1 MOSI (Master Out, Slave In) pin.
57			PC.12	I/O	General purpose digital I/O pin.
			SPI1_MISO1	I/O	2nd SPI1 MISO (Master In, Slave Out) pin.
58	33		PC.11	I/O	General purpose digital I/O pin.
			SPI1_MOSI0	I/O	1st SPI1 MOSI (Master Out, Slave In) pin.
59	34		PC.10	I/O	General purpose digital I/O pin.
			SPI1_MISO0	I/O	1st SPI1 MISO (Master In, Slave Out) pin.
60	35		PC.9	I/O	General purpose digital I/O pin.
			SPI1_CLK	I/O	SPI1 serial clock pin.
61	36		PC.8	I/O	General purpose digital I/O pin.
			SPI1_SS0	I/O	1st SPI1 slave select pin.
62	37	25	PA.15	I/O	General purpose digital I/O pin.
			PWM3	I/O	PWM output/Capture input.
			I2S_MCLK	O	I2S master clock output pin.
			SC2_PWR	O	SmartCard2 power pin.
63	38	26	PA.14	I/O	General purpose digital I/O pin.
			PWM2	I/O	PWM2 output/Capture input.

Pin No.			Pin Name	Pin Type	Description
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
			SC2_RST	**O**	SmartCard2 reset pin.
64	39	27	PA.13	**I/O**	General purpose digital I/O pin.
			PWM1	**I/O**	PWM1 output/Capture input.
			SC2_CLK	**O**	SmartCard2 clock pin.
65	40	28	PA.12	**I/O**	General purpose digital I/O pin.
			PWM0	**I/O**	PWM0 output/Capture input.
			SC2_DAT	**O**	SmartCard2 data pin.
66	41	29	ICE_DAT	**I/O**	Serial wire debugger data pin.
67	42	30	ICE_CLK	**I**	Serial wire debugger clock pin.
68			V_{DD}	**P**	Power supply for I/O ports and LDO source for internal PLL and digital circuit.
69			V_{SS}	**P**	Ground pin for digital circuit.
70	43	31	AV_{SS}	**AP**	Ground pin for analog circuit.
71	44	32	PA.0	**I/O**	General purpose digital I/O pin.
			ADC0	**AI**	ADC0 analog input.
			SC0_PWR	**O**	SmartCard0 power pin.
72	45	33	PA.1	**I/O**	General purpose digital I/O pin.
			ADC1	**AI**	ADC1 analog input.
			SC0_RST	**O**	SmartCard0 reset pin.
73	46	34	PA.2	**I/O**	General purpose digital I/O pin.
			ADC2	**AI**	ADC2 analog input.
			SC0_CLK	**O**	SmartCard0 clock pin.
74	47	35	PA.3	**I/O**	General purpose digital I/O pin.
			ADC3	**AI**	ADC3 analog input.
			SC0_DAT	**O**	SmartCard0 data pin.
75	48	36	PA.4	**I/O**	General purpose digital I/O pin.
			ADC4	**AI**	ADC4 analog input.
			SC1_PWR	**O**	SmartCard1 power pin.
76	49	37	PA.5	**I/O**	General purpose digital I/O pin.
			ADC5	**AI**	ADC5 analog input.
			SC1_RST	**O**	SmartCard1 reset pin.
77	50	38	PA.6	**I/O**	General purpose digital I/O pin.

Pin No.			Pin Name	Pin Type	Description
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
			ADC6	**AI**	ADC6 analog input.
			SC1_CLK	**I/O**	SmartCard1 clock pin.
78			PA.7	**I/O**	General purpose digital I/O pin.
			ADC7	**AI**	ADC7 analog input.
			SC1_DAT	**O**	SmartCard1 data pin.
			SPI2_SS1	**I/O**	2nd SPI2 slave select pin.
79	51	39	V_{REF}	**AP**	Voltage reference input for ADC.
80	52	40	AV_{DD}	**AP**	Power supply for internal analog circuit.
81			PD.0	**I/O**	General purpose digital I/O pin.
			SPI2_SS0	**I/O**	1st SPI2 slave select pin.
82			PD.1	**I/O**	General purpose digital I/O pin.
			SPI2_CLK	**I/O**	SPI2 serial clock pin.
83			PD.2	**I/O**	General purpose digital I/O pin.
			SPI2_MISO0	**I/O**	1st SPI2 MISO (Master In, Slave Out) pin.
84			PD.3	**I/O**	General purpose digital I/O pin.
			SPI2_MOSI0	**I/O**	1st SPI2 MOSI (Master Out, Slave In) pin.
85			PD.4	**I/O**	General purpose digital I/O pin.
			SPI2_MISO1	**I/O**	2nd SPI2 MISO (Master In, Slave Out) pin.
86			PD.5	**I/O**	General purpose digital I/O pin.
			SPI2_MOSI1	**I/O**	2nd SPI2 MOSI (Master Out, Slave In) pin.
87	53	41	PC.7	**I/O**	General purpose digital I/O pin.
			CMP0_N	**AI**	Comparator0 negative input pin.
			SC1_CD	**I**	SmartCard1 card detect pin.
88	54	42	PC.6	**I/O**	General purpose digital I/O pin.
			CMP0_P	**AI**	Comparator0 positive input pin.
			SC0_CD	**I**	SmartCard0 card detect pin.
89	55		PC.15	**I/O**	General purpose digital I/O pin.
			CMP1_N	**AI**	Comparator1 negative input pin.
90	56		PC.14	**I/O**	General purpose digital I/O pin.
			CMP1_P	**AI**	Comparator1 positive input pin.
91	57	43	PB.15	**I/O**	General purpose digital I/O pin.

Pin No.			Pin Name	Pin Type	Description
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
			INT1	**I**	External interrupt1 input pin.
			TM0_EXT	**I**	Timer0 external capture input pin.
92	58	44	PF.0	**I/O**	General purpose digital I/O pin.
			XT1_OUT	**O**	External 4~24 MHz (high speed) crystal output pin.
93	59	45	PF.1	**I/O**	General purpose digital I/O pin.
			XT1_IN	**I**	External 4~24 MHz (high speed) crystal input pin.
94	60	46	nRESET	**I**	External reset input: active LOW, with an internal pull-up. Set this pin low reset chip to initial state.
95	61		V_{SS}	**P**	Ground pin for digital circuit.
96	62		V_{DD}	**P**	Power supply for I/O ports and LDO source for internal PLL and digital circuit.
97			PF.2	**I/O**	General purpose digital I/O pin.
			PS2_DAT	**I/O**	PS2 data pin.
98			PF.3	**I/O**	General purpose digital I/O pin.
			PS2_CLK	**I/O**	PS2 clock pin.
99	63	47	PV_{SS}	**P**	PLL ground.
100	64	48	PB.8	**I/O**	General purpose digital I/O pin.
			STADC	**I**	ADC external trigger input.
			TM0	**I/O**	Timer0 event counter input / toggle output.
			CLKO	**O**	Frequency divider clock output pin.

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

3.3.2 NuMicro™ NUC220 Pin Description

Pin No.			Pin Name	Pin Type	Description
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
1			PE.15	**I/O**	General purpose digital I/O pin.
2			PE.14	**I/O**	General purpose digital I/O pin.
3			PE.13	**I/O**	General purpose digital I/O pin.
4	1		PB.14	**I/O**	General purpose digital I/O pin.
			INT0	**I**	External interrupt0 input pin.
			SPI3_SS1	**I/O**	2nd SPI3 slave select pin.
5	2		PB.13	**I/O**	General purpose digital I/O pin.
			CMP1_O	**O**	Comparator1 output pin.
6	3	1	V_{BAT}	**P**	Power supply by batteries for RTC.
7	4	2	X32_OUT	**O**	External 32.768 kHz (low speed) crystal output pin.
8	5	3	X32_IN	**I**	External 32.768 kHz (low speed) crystal input pin.
9	6	4	PA.11	**I/O**	General purpose digital I/O pin.
			I2C1_SCL	**I/O**	I^2C1 clock pin.
10	7	5	PA.10	**I/O**	General purpose digital I/O pin.
			I2C1_SDA	**I/O**	I^2C1 data input/output pin.
11	8	6	PA.9	**I/O**	General purpose digital I/O pin.
			I2C0_SCL	**I/O**	I^2C0 clock pin.
12	9	7	PA.8	**I/O**	General purpose digital I/O pin.
			I2C0_SDA	**I/O**	I^2C0 data input/output pin.
13			PD.8	**I/O**	General purpose digital I/O pin.
			SPI3_SS0	**I/O**	1st SPI3 slave select pin.
14			PD.9	**I/O**	General purpose digital I/O pin.
			SPI3_CLK	**I/O**	SPI3 serial clock pin.
15			PD.10	**I/O**	General purpose digital I/O pin.
			SPI3_MISO0	**I/O**	1st SPI3 MISO (Master In, Slave Out) pin.
16			PD.11	**I/O**	General purpose digital I/O pin.
			SPI3_MOSI0	**I/O**	1st SPI3 MOSI (Master Out, Slave In) pin.
17			PD.12	**I/O**	General purpose digital I/O pin.
			SPI3_MISO1	**I/O**	2nd SPI3 MISO (Master In, Slave Out) pin.
18			PD.13	**I/O**	General purpose digital I/O pin.

Pin No.			Pin Name	Pin Type	Description
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
			SPI3_MOSI1	**I/O**	2nd SPI3 MOSI (Master Out, Slave In) pin.
19	10	8	PB.4	**I/O**	General purpose digital I/O pin.
			UART1_RXD	**I**	Data receiver input pin for UART1.
20	11	9	PB.5	**I/O**	General purpose digital I/O pin.
			UART1_TXD	**O**	Data transmitter output pin for UART1.
21	12		PB.6	**I/O**	General purpose digital I/O pin.
			UART1_nRTS	**O**	Request to Send output pin for UART1.
22	13		PB.7	**I/O**	General purpose digital I/O pin.
			UART1_nCTS	**I**	Clear to Send input pin for UART1.
23	14	10	LDO_CAP	**P**	LDO output pin.
24	15	11	V_{DD}	**P**	Power supply for I/O ports and LDO source for internal PLL and digital circuit.
25	16	12	V_{SS}	**P**	Ground pin for digital circuit.
26			PE.8	**I/O**	General purpose digital I/O pin.
27			PE.7	**I/O**	General purpose digital I/O pin.
28	17	13	USB_VBUS	**USB**	Power supply from USB host or HUB.
29	18	14	USB_VDD33_CAP	**USB**	Internal power regulator output 3.3V decoupling pin.
30	19	15	USB_D-	**USB**	USB differential signal D-.
31	20	16	USB_D+	**USB**	USB differential signal D+.
32	21	17	PB.0	**I/O**	General purpose digital I/O pin.
			UART0_RXD	**I**	Data receiver input pin for UART0.
33	22	18	PB.1	**I/O**	General purpose digital I/O pin.
			UART0_TXD	**O**	Data transmitter output pin for UART0.
34	23	19	PB.2	**I/O**	General purpose digital I/O pin.
			UART0_nRTS	**O**	Request to Send output pin for UART0.
			TM2_EXT	**I**	Timer2 external capture input pin.
			CMP0_O	**O**	Comparator0 output pin.
35	24	20	PB.3	**I/O**	General purpose digital I/O pin.
			UART0_nCTS	**I**	Clear to Send input pin for UART0.
			TM3_EXT	**I**	Timer3 external capture input pin.
			SC2_CD	**I**	SmartCard2 card detect pin.

Pin No.			Pin Name	Pin Type	Description
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
36			PD.6	**I/O**	General purpose digital I/O pin.
37			PD.7	**I/O**	General purpose digital I/O pin.
38			PD.14	**I/O**	General purpose digital I/O pin.
			UART2_RXD	**I**	Data receiver input pin for UART2.
39			PD.15	**I/O**	General purpose digital I/O pin.
			UART2_TXD	**O**	Data transmitter output pin for UART2.
40			PC.5	**I/O**	General purpose digital I/O pin.
			SPI0_MOSI1	**I/O**	2nd SPI0 MOSI (Master Out, Slave In) pin.
41			PC.4	**I/O**	General purpose digital I/O pin.
			SPI0_MISO1	**I/O**	2nd SPI0 MISO (Master In, Slave Out) pin.
42	25	21	PC.3	**I/O**	General purpose digital I/O pin.
			SPI0_MOSI0	**I/O**	1st SPI0 MOSI (Master Out, Slave In) pin.
			I2S_DO	**O**	I2S data output.
43	26	22	PC.2	**I/O**	General purpose digital I/O pin.
			SPI0_MISO0	**I/O**	1st SPI0 MISO (Master In, Slave Out) pin.
			I2S_DI	**I**	I2S data input.
44	27	23	PC.1	**I/O**	General purpose digital I/O pin.
			SPI0_CLK	**I/O**	SPI0 serial clock pin.
			I2S_BCLK	**I/O**	I2S bit clock pin.
45	28	24	PC.0	**I/O**	General purpose digital I/O pin.
			SPI0_SS0	**I/O**	1st SPI0 slave select pin.
			I2S_LRCK	**I/O**	I2S left right channel clock.
46			PE.6	**I/O**	General purpose digital I/O pin.
47	29		PE.5	**I/O**	General purpose digital I/O pin.
			PWM5	**I/O**	PWM5 output/Capture input.
			TM1_EXT	**I**	Timer1 external capture input pin.
48	30		PB.11	**I/O**	General purpose digital I/O pin.
			TM3	**I/O**	Timer3 event counter input / toggle output.
			PWM4	**I/O**	PWM4 output/Capture input.
49	31		PB.10	**I/O**	General purpose digital I/O pin.

Pin No.			Pin Name	Pin Type	Description
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
			TM2	I/O	Timer2 event counter input / toggle output.
			SPI0_SS1	I/O	2nd SPI0 slave select pin.
50	32		PB.9	I/O	General purpose digital I/O pin.
			TM1	I/O	Timer1 event counter input / toggle output.
			SPI1_SS1	I/O	2nd SPI1 slave select pin.
51			PE.4	I/O	General purpose digital I/O pin.
52			PE.3	I/O	General purpose digital I/O pin.
53			PE.2	I/O	General purpose digital I/O pin.
54			PE.1	I/O	General purpose digital I/O pin.
			PWM7	I/O	PWM7 output/Capture input.
55			PE.0	I/O	General purpose digital I/O pin.
			PWM6	I/O	PWM6 output/Capture input.
56			PC.13	I/O	General purpose digital I/O pin.
			SPI1_MOSI1	I/O	2nd SPI1MOSI (Master Out, Slave In) pin.
57			PC.12	I/O	General purpose digital I/O pin.
			SPI1_MISO1	I/O	2nd SPI1 MISO (Master In, Slave Out) pin.
58	33		PC.11	I/O	General purpose digital I/O pin.
			SPI1_MOSI0	I/O	1st SPI1 MOSI (Master Out, Slave In) pin.
59	34		PC.10	I/O	General purpose digital I/O pin.
			SPI1_MISO0	I/O	1st SPI1 MISO (Master In, Slave Out) pin.
60	35		PC.9	I/O	General purpose digital I/O pin.
			SPI1_CLK	I/O	SPI1 serial clock pin.
61	36		PC.8	I/O	General purpose digital I/O pin.
			SPI1_SS0	I/O	1st SPI1 slave select pin.
62	37	25	PA.15	I/O	General purpose digital I/O pin.
			PWM3	I/O	PWM3 output/Capture input.
			I2S_MCLK	O	I^2S master clock output pin.
			SC2_PWR	O	SmartCard2 power pin.
63	38	26	PA.14	I/O	General purpose digital I/O pin.
			PWM2	I/O	PWM2 output/Capture input.
			SC2_RST	O	SmartCard2 reset pin.

Pin No.			Pin Name	Pin Type	Description
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
64	39	27	PA.13	I/O	General purpose digital I/O pin.
			PWM1	I/O	PWM1 output/Capture input.
			SC2_CLK	O	SmartCard2 clock pin.
65	40	28	PA.12	I/O	General purpose digital I/O pin.
			PWM0	I/O	PWM0 output/Capture input.
			SC2_DAT	O	SmartCard2 data pin.
66	41	29	ICE_DAT	I/O	Serial wire debugger data pin.
67	42	30	ICE_CLK	I	Serial wire debugger clock pin.
68			V_{DD}	P	Power supply for I/O ports and LDO source for internal PLL and digital circuit.
69			V_{SS}	P	Ground pin for digital circuit.
70	43	31	AV_{SS}	AP	Ground pin for analog circuit.
71	44	32	PA.0	I/O	General purpose digital I/O pin.
			ADC0	AI	ADC0 analog input.
			SC0_PWR	O	SmartCard0 power pin.
72	45	33	PA.1	I/O	General purpose digital I/O pin.
			ADC1	AI	ADC1 analog input.
			SC0_RST	O	SmartCard0 reset pin.
73	46	34	PA.2	I/O	General purpose digital I/O pin.
			ADC2	AI	ADC2 analog input.
			SC0_CLK	O	SmartCard0 clock pin.
74	47	35	PA.3	I/O	General purpose digital I/O pin.
			ADC3	AI	ADC3 analog input.
			SC0_DAT	O	SmartCard0 data pin.
75	48	36	PA.4	I/O	General purpose digital I/O pin.
			ADC4	AI	ADC4 analog input.
			SC1_PWR	O	SmartCard1 power pin.
76	49	37	PA.5	I/O	General purpose digital I/O pin.
			ADC5	AI	ADC5 analog input.
			SC1_RST	O	SmartCard1 reset pin.
77	50	38	PA.6	I/O	General purpose digital I/O pin.
			ADC6	AI	ADC6 analog input.

Pin No.			Pin Name	Pin Type	Description
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
			SC1_CLK	**I/O**	SmartCard1 clock pin.
78			PA.7	**I/O**	General purpose digital I/O pin.
			ADC7	**AI**	ADC7 analog input.
			SC1_CLK	**O**	SmartCard1 clock pin.
			SPI2_SS1	**I/O**	2nd SPI2 slave select pin.
79	51	39	V_{REF}	**AP**	Voltage reference input for ADC.
80	52	40	AV_{DD}	**AP**	Power supply for internal analog circuit.
81			PD.0	**I/O**	General purpose digital I/O pin.
			SPI2_SS0	**I/O**	1st SPI2 slave select pin.
82			PD.1	**I/O**	General purpose digital I/O pin.
			SPI2_CLK	**I/O**	SPI2 serial clock pin.
83			PD.2	**I/O**	General purpose digital I/O pin.
			SPI2_MISO0	**I/O**	1st SPI2 MISO (Master In, Slave Out) pin.
84			PD.3	**I/O**	General purpose digital I/O pin.
			SPI2_MOSI0	**I/O**	1st SPI2 MOSI (Master Out, Slave In) pin.
85			PD.4	**I/O**	General purpose digital I/O pin.
			SPI2_MISO1	**I/O**	2nd SPI2 MISO (Master In, Slave Out) pin.
86			PD.5	**I/O**	General purpose digital I/O pin.
			SPI2_MOSI1	**I/O**	2nd SPI2 MOSI (Master Out, Slave In) pin.
87	53	41	PC.7	**I/O**	General purpose digital I/O pin.
			CMP0_N	**AI**	Comparator0 negative input pin.
			SC1_CD	**I**	SmartCard1 card detect pin.
88	54	42	PC.6	**I/O**	General purpose digital I/O pin.
			CMP0_P	**AI**	Comparator0 positive input pin.
			SC0_CD	**I**	SmartCard0 card detect pin.
89	55		PC.15	**I/O**	General purpose digital I/O pin.
			CMP1_N	**AI**	Comparator1 negative input pin.
90	56		PC.14	**I/O**	General purpose digital I/O pin.
			CMP1_P	**AI**	Comparator1 positive input pin.
91	57	43	PB.15	**I/O**	General purpose digital I/O pin.
			INT1	**I**	External interrupt1 input pin.

Pin No.			Pin Name	Pin Type	Description
LQFP 100-pin	LQFP 64-pin	LQFP 48-pin			
			TM0_EXT	I	Timer 0 external capture input pin.
92	58	44	PF.0	I/O	General purpose digital I/O pin.
			XT1_OUT	O	External 4~24 MHz (high speed) crystal output pin.
93	59	45	PF.1	I/O	General purpose digital I/O pin.
			XT1_IN	I	External 4~24 MHz (high speed) crystal input pin.
94	60	46	nRESET	I	External reset input: active LOW, with an internal pull-up. Set this pin low reset chip to initial state.
95	61		V_{SS}	P	Ground pin for digital circuit.
96	62		V_{DD}	P	Power supply for I/O ports and LDO source for internal PLL and digital circuit.
97			PF.2	I/O	General purpose digital I/O pin.
			PS2_DAT	I/O	PS2 data pin.
98			PF.3	I/O	General purpose digital I/O pin.
			PS2_CLK	I/O	PS2 clock pin.
99	63	47	PV_{SS}	P	PLL ground.
100	64	48	PB.8	I/O	General purpose digital I/O pin.
			STADC	I	ADC external trigger input.
			TM0	I/O	Timer0 event counter input / toggle output.
			CLKO	O	Frequency divider clock output pin.

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

4 BLOCK DIAGRAM

4.1 NuMicro™ NUC200 Block Diagram

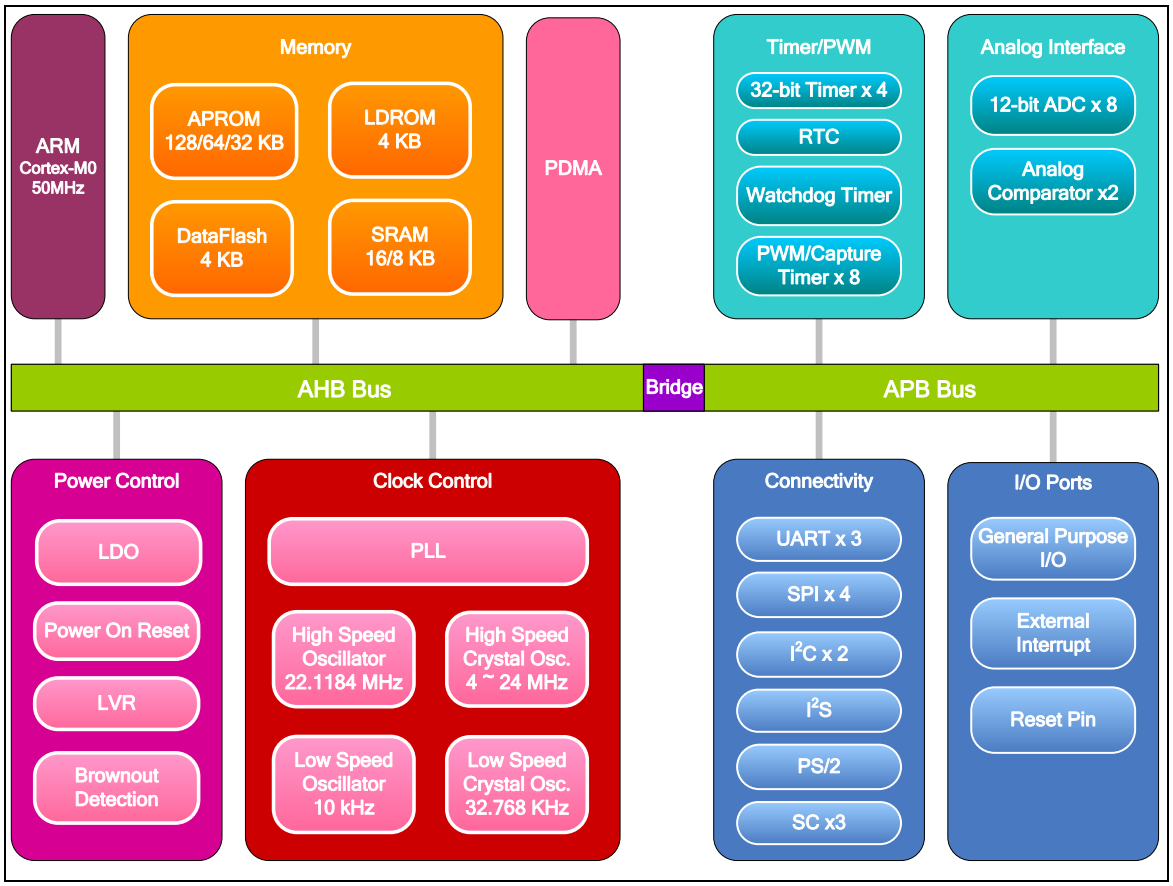


Figure 4-1 NuMicro™ NUC200 Block Diagram

4.2 NuMicro™ NUC220 Block Diagram

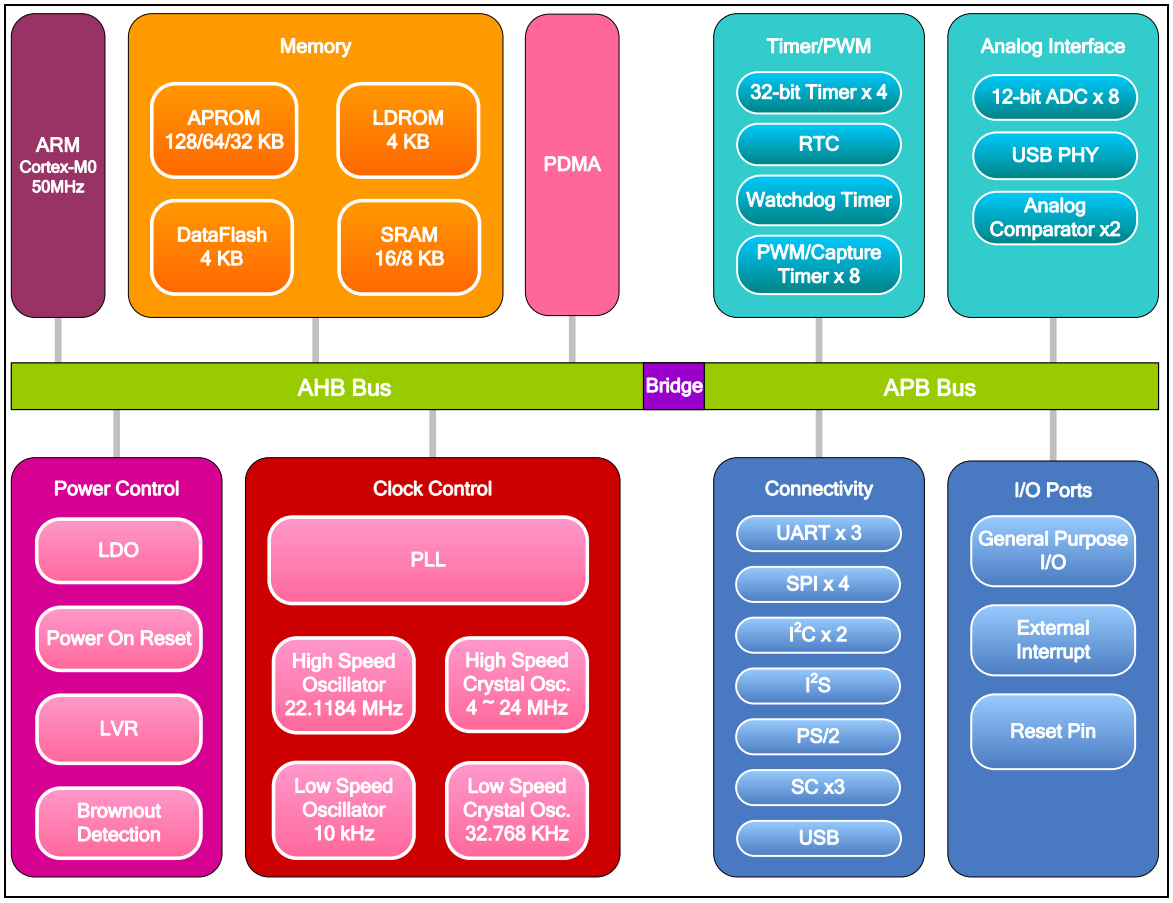


Figure 4-2 NuMicro™ NUC220 Block Diagram

5 FUNCTIONAL DESCRIPTION

5.1 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 5-1 shows the functional controller of processor.

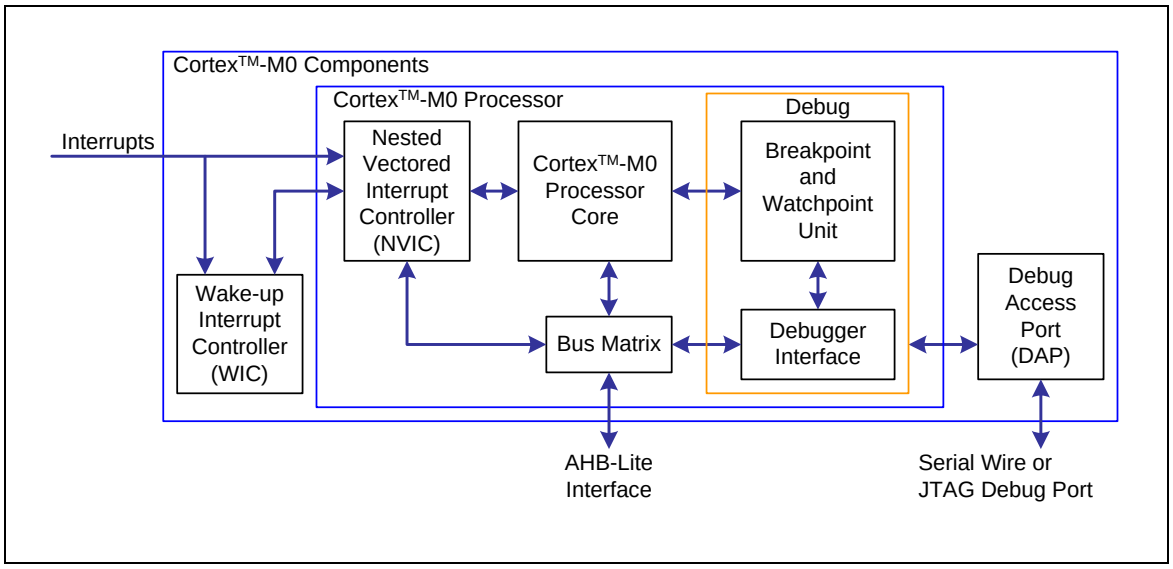


Figure 5-1 Functional Controller Diagram

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 supported with 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 Sleep mode entry using Wait For Interrupt (WFI), Wait For Event (WFE) instructions, or the return from interrupt sleep-on-exit feature
- NVIC:

- 32 external interrupt inputs, each with four levels of priority
- Dedicated Non-maskable Interrupt (NMI) input
- Supports 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).

5.2 System Manager

5.2.1 Overview

System management includes the following sections:

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

5.2.2 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.

- Power-on Reset
- Low level on the nRESET pin
- Watchdog Time-out Reset
- Low Voltage Reset
- Brown-out Detector Reset
- CPU Reset
- System Reset

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

5.2.3 System Power Distribution

In this chip, the power distribution is divided into four 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 internal regulator which provides a fixed 1.8 V power for digital operation and I/O pins.
- USB transceiver power from USB_VBUS offers the power for operating the USB transceiver.
- Battery power from V_{BAT} supplies the RTC and external 32.768 kHz crystal.

The outputs of internal voltage regulators, LDO_CAP and USB_VDD33_CAP, require an external capacitor which should be located close to the corresponding pin. Analog power (AV_{DD}) should be the same voltage level of the digital power (V_{DD}). Figure 5-2 shows the power distribution of NuMicro™ NUC200; Figure 5-3 shows the power distribution of NuMicro™ NUC220.

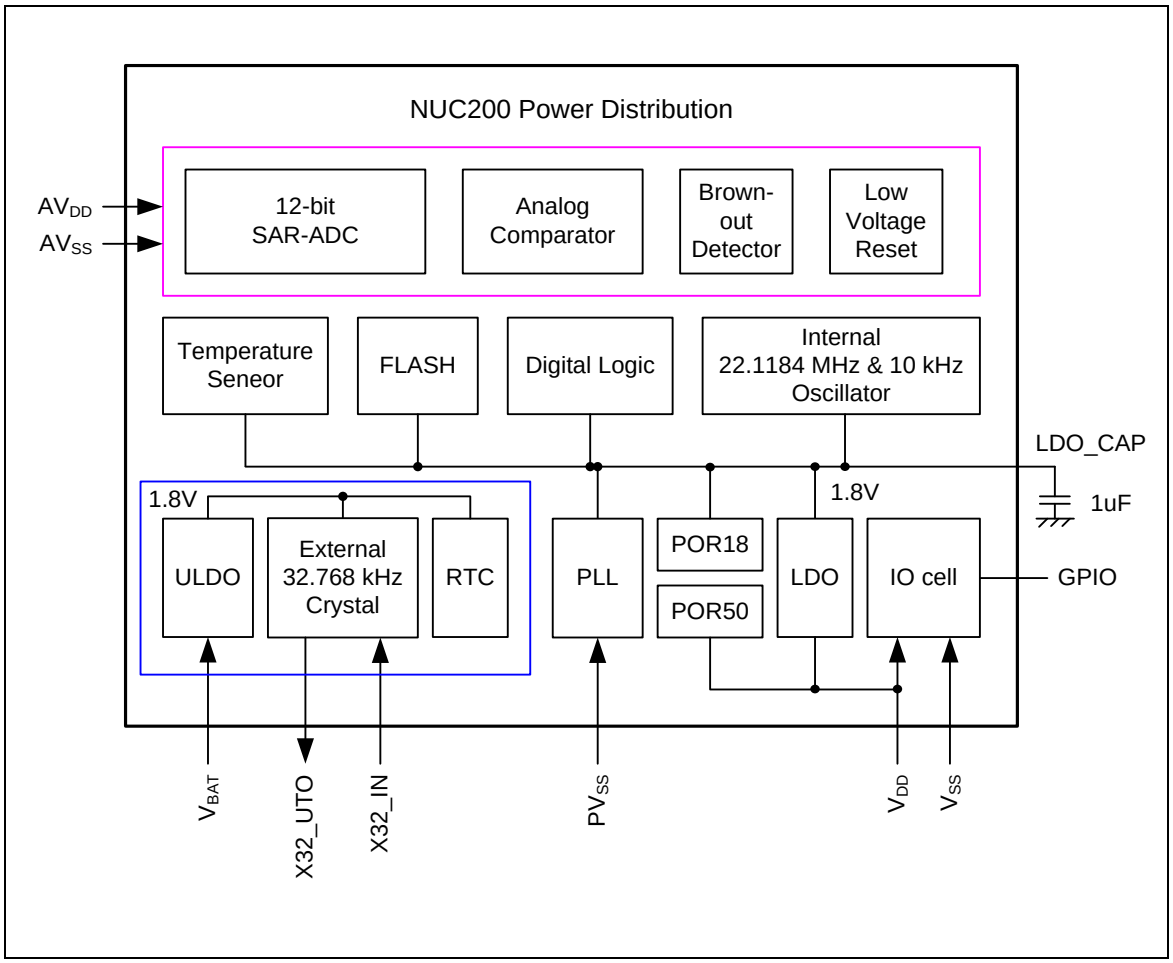


Figure 5-2 NuMicro™ NUC200 Power Distribution Diagram

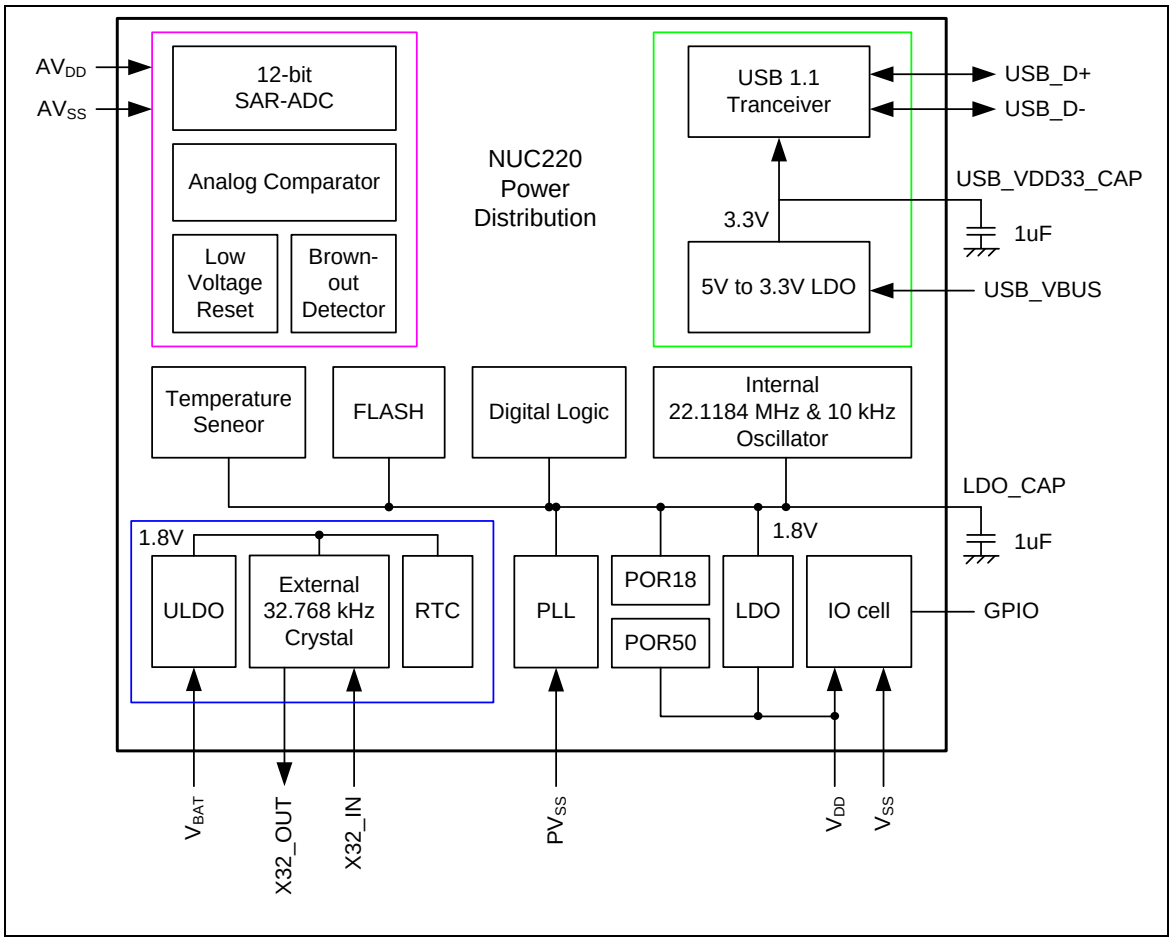


Figure 5-3 NuMicro™ NUC220 Power Distribution Diagram

5.2.4 System Memory Map

The NuMicro™ NUC200 Series provides 4G-byte addressing space. The memory locations assigned to each on-chip controllers are shown in the following table. The detailed register definition, memory space, and programming detailed will be described in the following sections for each on-chip peripheral. The NuMicro™ NUC200 Series only supports little-endian data format.

Address Space	Token	Controllers
Flash and SRAM Memory Space		
0x0000_0000 – 0x0001_FFFF	FLASH_BA	FLASH Memory Space (128 KB)
0x2000_0000 – 0x2000_3FFF	SRAM_BA	SRAM Memory Space (16 KB)
AHB Controllers Space (0x5000_0000 – 0x501F_FFFF)		
0x5000_0000 – 0x5000_01FF	GCR_BA	System Global Control Registers
0x5000_0200 – 0x5000_02FF	CLK_BA	Clock Control Registers
0x5000_0300 – 0x5000_03FF	INT_BA	Interrupt Multiplexer Control Registers
0x5000_4000 – 0x5000_7FFF	GPIO_BA	GPIO Control Registers
0x5000_8000 – 0x5000_BFFF	PDMA_BA	Peripheral DMA Control Registers
0x5000_C000 – 0x5000_FFFF	FMC_BA	Flash Memory Control Registers
APB1 Controllers Space (0x4000_0000 ~ 0x400F_FFFF)		
0x4000_4000 – 0x4000_7FFF	WDT_BA	Watchdog Timer Control Registers
0x4000_8000 – 0x4000_BFFF	RTC_BA	Real Time Clock (RTC) Control Register
0x4001_0000 – 0x4001_3FFF	TMR01_BA	Timer0/Timer1 Control Registers
0x4002_0000 – 0x4002_3FFF	I2C0_BA	I^2C0 Interface Control Registers
0x4003_0000 – 0x4003_3FFF	SPI0_BA	SPI0 with master/slave function Control Registers
0x4003_4000 – 0x4003_7FFF	SPI1_BA	SPI1 with master/slave function Control Registers
0x4004_0000 – 0x4004_3FFF	PWMA_BA	PWM0/1/2/3 Control Registers
0x4005_0000 – 0x4005_3FFF	UART0_BA	UART0 Control Registers
0x4006_0000 – 0x4006_3FFF	USBD_BA	USB 2.0 FS device Controller Registers
0x400D_0000 – 0x400D_3FFF	ACMP_BA	Analog Comparator Control Registers
0x400E_0000 – 0x400E_FFFF	ADC_BA	Analog-Digital-Converter (ADC) Control Registers
APB2 Controllers Space (0x4010_0000 ~ 0x401F_FFFF)		
0x4010_0000 – 0x4010_3FFF	PS2_BA	PS/2 Interface Control Registers

0x4011_0000 – 0x4011_3FFF	TMR23_BA	Timer2/Timer3 Control Registers
0x4012_0000 – 0x4012_3FFF	I2C1_BA	I^2C1 Interface Control Registers
0x4013_0000 – 0x4013_3FFF	SPI2_BA	SPI2 with master/slave function Control Registers
0x4013_4000 – 0x4013_7FFF	SPI3_BA	SPI3 with master/slave function Control Registers
0x4014_0000 – 0x4014_3FFF	PWMB_BA	PWM4/5/6/7 Control Registers
0x4015_0000 – 0x4015_3FFF	UART1_BA	UART1 Control Registers
0x4015_4000 – 0x4015_7FFF	UART2_BA	UART2 Control Registers
0x4019_0000 – 0x4019_3FFF	SC0_BA	SC0 Control Registers
0x4019_4000 – 0x4019_7FFF	SC1_BA	SC1 Control Registers
0x4019_8000 – 0x4019_BFFF	SC2_BA	SC2 Control Registers
0x401A_0000 – 0x401A_3FFF	I2S_BA	I^2S Interface Control Registers
System Controllers Space (0xE000_E000 ~ 0xE000_EFFF)		
0xE000_E010 – 0xE000_E0FF	SYST_BA	System Timer Control Registers
0xE000_E100 – 0xE000_ECFF	NVIC_BA	External Interrupt Controller Control Registers
0xE000_ED00 – 0xE000_ED8F	SCS_BA	System Control Registers

Table 5-1 Address Space Assignments for On-Chip Controllers

5.2.5 System 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".

5.2.6 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 kernel 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".

5.2.6.1 *Exception Model and System Interrupt Map*

The following table lists the exception model supported by NuMicro™ NUC200 Series. Software can set four levels of priority on some of these exceptions as well as on all interrupts. The highest user-configurable priority is denoted as “0” and the lowest priority is denoted as “3”. The default priority of all the user-configurable interrupts is “0”. Note that priority “0” is treated as the fourth priority on the system, after three system exceptions “Reset”, “NMI” and “Hard Fault”.

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

Table 5-2 Exception Model

Vector Number	Interrupt Number (Bit in Interrupt Registers)	Interrupt Name	Source IP	Interrupt Description
0 ~ 15	-	**-**	-	System exceptions
16	0	**BOD_INT**	Brown-out	Brown-out low voltage detected interrupt
17	1	**WDT_INT**	WDT	Watchdog Timer interrupt
18	2	**EINT0**	GPIO	External signal interrupt from PB.14 pin
19	3	**EINT1**	GPIO	External signal interrupt from PB.15 pin
20	4	**GPAB_INT**	GPIO	External signal interrupt from PA[15:0]/PB[13:0]
21	5	**GPCDEF_INT**	GPIO	External interrupt from PC[15:0]/PD[15:0]/PE[15:0]/ PF[3:0]
22	6	**PWMA_INT**	PWM0~3	PWM0, PWM1, PWM2 and PWM3 interrupt
23	7	**PWMB_INT**	PWM4~7	PWM4, PWM5, PWM6 and PWM7 interrupt
24	8	**TMR0_INT**	TMR0	Timer 0 interrupt
25	9	**TMR1_INT**	TMR1	Timer 1 interrupt
26	10	**TMR2_INT**	TMR2	Timer 2 interrupt
27	11	**TMR3_INT**	TMR3	Timer 3 interrupt

28	12	**UART02_INT**	UART0/2	UART0 and UART2 interrupt
29	13	**UART1_INT**	UART1	UART1 interrupt
30	14	**SPI0_INT**	SPI0	SPI0 interrupt
31	15	**SPI1_INT**	SPI1	SPI1 interrupt
32	16	**SPI2_INT**	SPI2	SPI2 interrupt
33	17	**SPI3_INT**	SPI3	SPI3 interrupt
34	18	**I2C0_INT**	I^2C0	I^2C0 interrupt
35	19	**I2C1_INT**	I^2C1	I^2C1 interrupt
36	20	**Reserved**	-	-
37	21	**Reserved**	-	-
38	22	**SC012_INT**	SC0/1/2	SC0, SC1 and SC2 interrupt
39	23	**USB_INT**	USBD	USB 2.0 FS Device interrupt
40	24	**PS2_INT**	PS/2	PS/2 interrupt
41	25	**ACMP_INT**	ACMP	Analog Comparator-0 or Comaprator-1 interrupt
42	26	**PDMA_INT**	PDMA	PDMA interrupt
43	27	**I2S_INT**	I^2S	I^2S interrupt
44	28	**PWRWU_INT**	CLKC	Clock controller interrupt for chip wake-up from power-down state
45	29	**ADC_INT**	ADC	ADC interrupt
46	30	**IRCT_INT**	IRC	IRC TRIM interrupt
47	31	**RTC_INT**	RTC	Real time clock interrupt

Table 5-3 System Interrupt Map

5.2.6.2 *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	Description
0	SP_main – The Main stack pointer
Vector Number	Exception Entry Pointer using that Vector Number

Table 5-4 Vector Table Format

5.2.6.3 *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.

5.2.6.4 *Interrupt Source Register Map*

Besides the interrupt control registers associated with the NVIC, the NuMicro™ NUC200 Series also implements some specific control registers to facilitate the interrupt functions, including “interrupt source identification”, ”NMI source selection” and “interrupt test mode”, which are described below.

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

Register	Offset	R/W	Description	Reset Value
INT 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 (GPA/GPB) interrupt source identity	0xXXXX_XXXX
IRQ5_SRC	INT_BA+0x14	R	IRQ5 (GPC/GPD/GPE/GPF) interrupt source identity	0xXXXX_XXXX
IRQ6_SRC	INT_BA+0x18	R	IRQ6 (PWMA) interrupt source identity	0xXXXX_XXXX
IRQ7_SRC	INT_BA+0x1C	R	IRQ7 (PWMB) interrupt source identity	0xXXXX_XXXX
IRQ8_SRC	INT_BA+0x20	R	IRQ8 (TMR0) interrupt source identity	0xXXXX_XXXX
IRQ9_SRC	INT_BA+0x24	R	IRQ9 (TMR1) interrupt source identity	0xXXXX_XXXX
IRQ10_SRC	INT_BA+0x28	R	IRQ10 (TMR2) interrupt source identity	0xXXXX_XXXX
IRQ11_SRC	INT_BA+0x2C	R	IRQ11 (TMR3) interrupt source identity	0xXXXX_XXXX
IRQ12_SRC	INT_BA+0x30	R	IRQ12 (UART0/UART2) 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 (SPI3) interrupt source identity	0xXXXX_XXXX
IRQ18_SRC	INT_BA+0x48	R	IRQ18 (I^2C0) interrupt source identity	0xXXXX_XXXX
IRQ19_SRC	INT_BA+0x4C	R	IRQ19 (I^2C1) interrupt source identity	0xXXXX_XXXX
IRQ20_SRC	INT_BA+0x50	R	Reserved	0xXXXX_XXXX
IRQ21_SRC	INT_BA+0x54	R	Reserved	0xXXXX_XXXX
IRQ22_SRC	INT_BA+0x58	R	IRQ22 (SC0/SC1/SC2) interrupt source identity	0xXXXX_XXXX
IRQ23_SRC	INT_BA+0x5C	R	IRQ23 (USB) interrupt source identity	0xXXXX_XXXX

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

5.2.7 System Control (SCS)

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

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

5.3 Clock Controller

5.3.1 Overview

The clock controller generates clocks for the whole chip, including system clocks and all peripheral clocks. The clock controller also implements the power control function with the individually clock ON/OFF control, clock source selection and clock divider. The chip enters Power-down mode when Cortex™-M0 core executes the WFI instruction only if the PWR_DOWN_EN (PWRCON[7]) bit and PD_WAIT_CPU (PWRCON[8]) bit are both set to 1. After that, chip enters Power-down mode and waits for wake-up interrupt source triggered to exit Power-down mode. In Power-down mode, the clock controller turns off the external 4~24 MHz high speed crystal and internal 22.1184 MHz high speed oscillator to reduce the overall system power consumption.

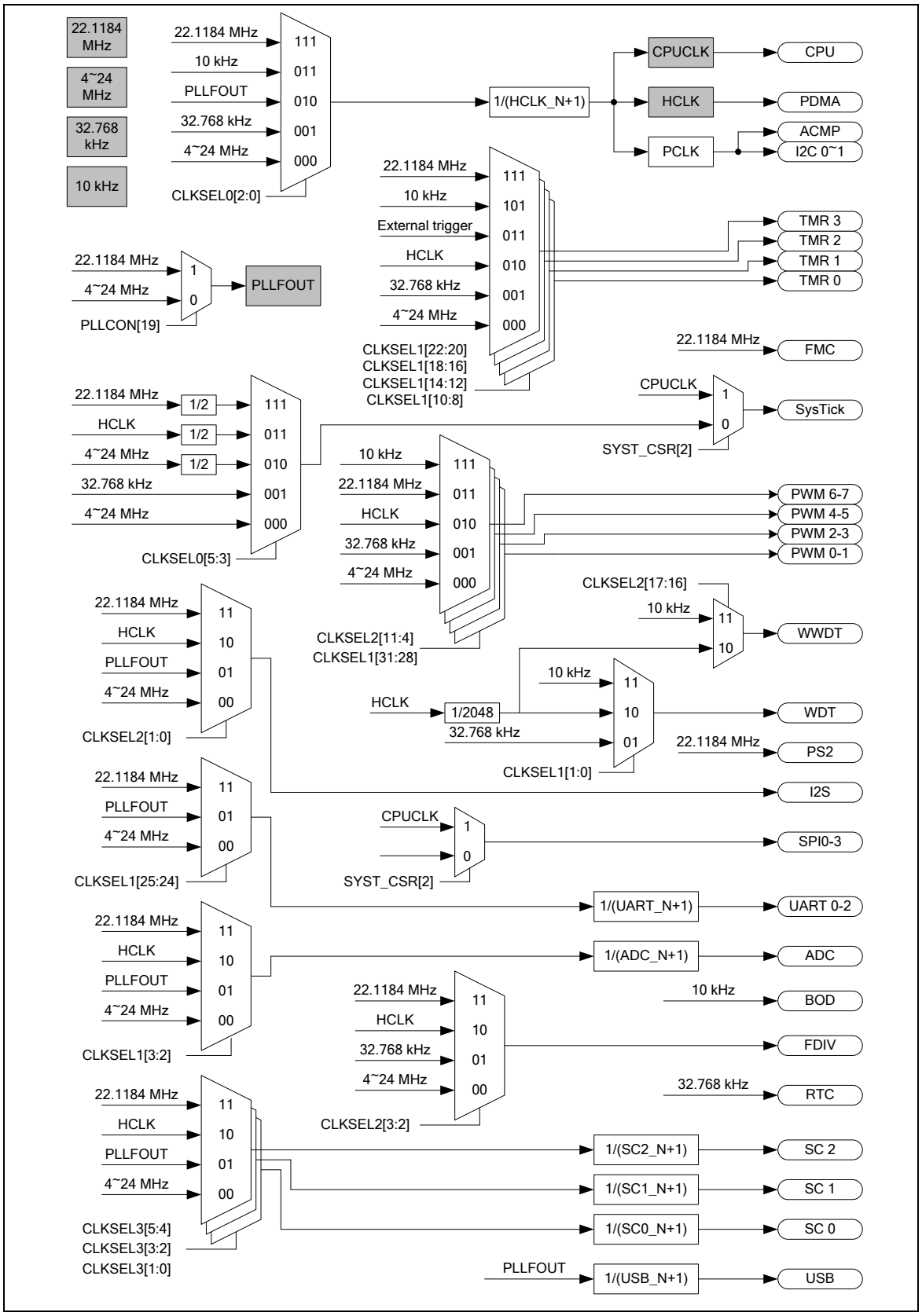


Figure 5-4 Clock Generator Global View Diagram

5.3.2 Clock Generator

The clock generator consists of 5 clock sources as listed below:

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

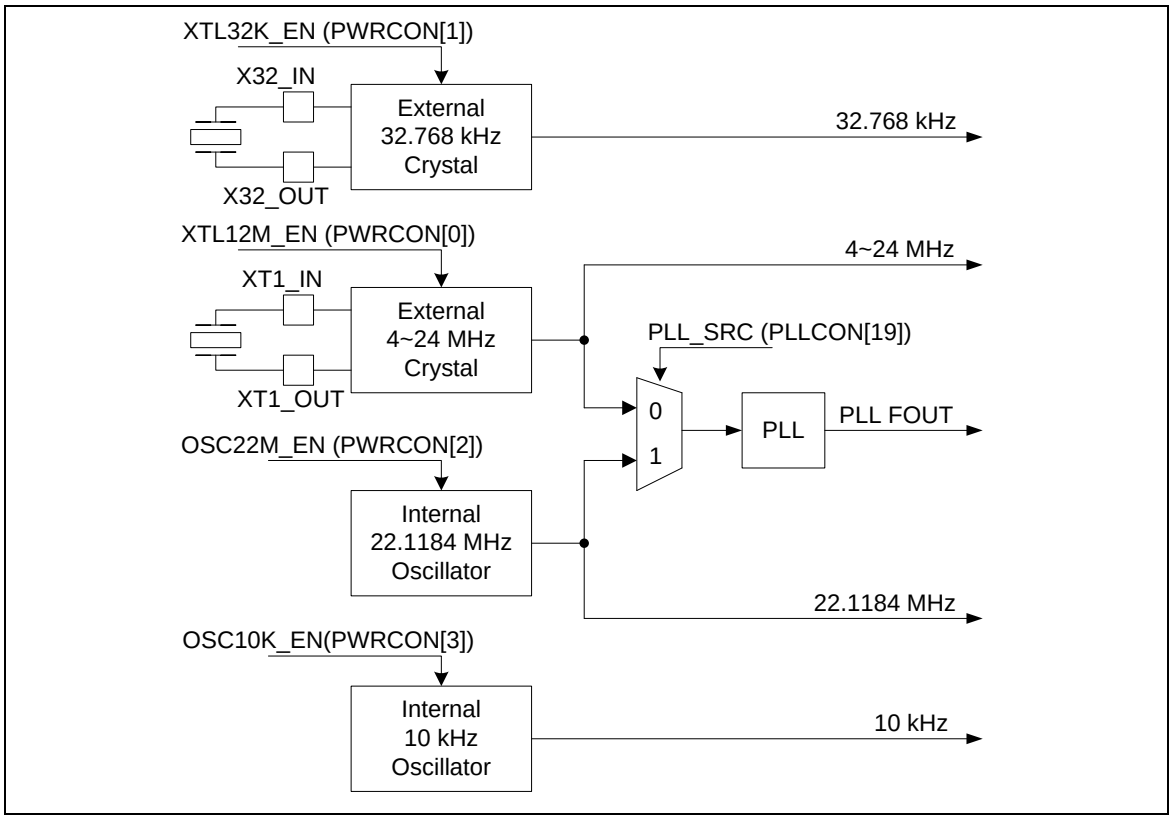


Figure 5-5 Clock Generator Block Diagram

5.3.3 System Clock and SysTick Clock

The system clock has 5 clock sources which were generated from clock generator block. The clock source switch depends on the register HCLK_S (CLKSEL0[2:0]). The block diagram is shown in Figure 5-6.

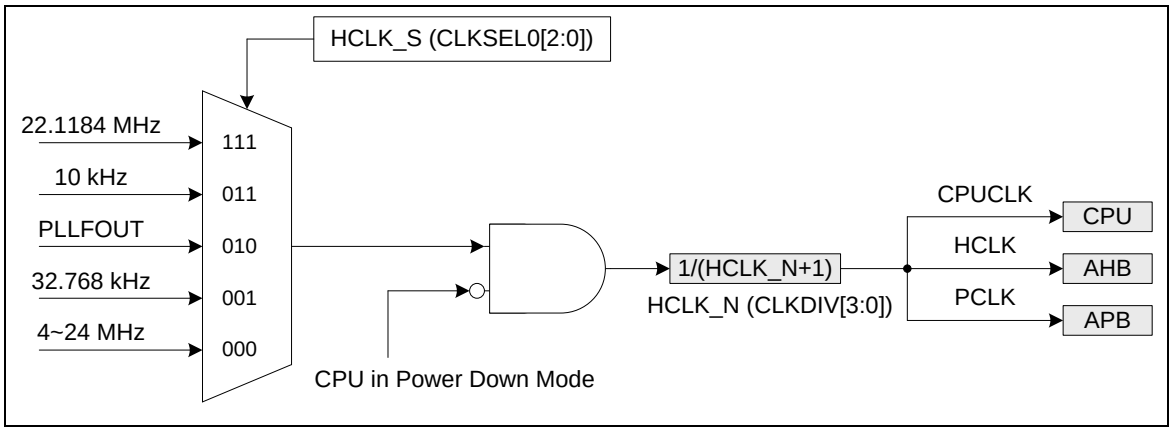


Figure 5-6 System Clock Block Diagram

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

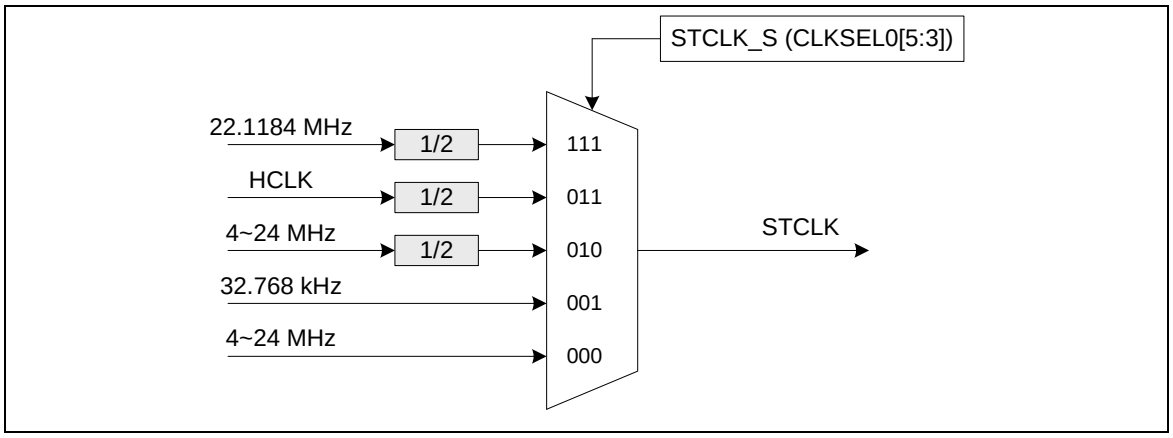


Figure 5-7 SysTick Clock Control Block Diagram

5.3.4 Peripherals Clock

The peripherals clock can be selected as different clock source depends on the clock source select control registers (CLKSEL1, CLKSEL2 and CLKSEL3).

5.3.5 Power-down 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 clocks are still active in Power-down mode.

The clocks still kept active are listed below:

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

5.3.6 Frequency Divider Output

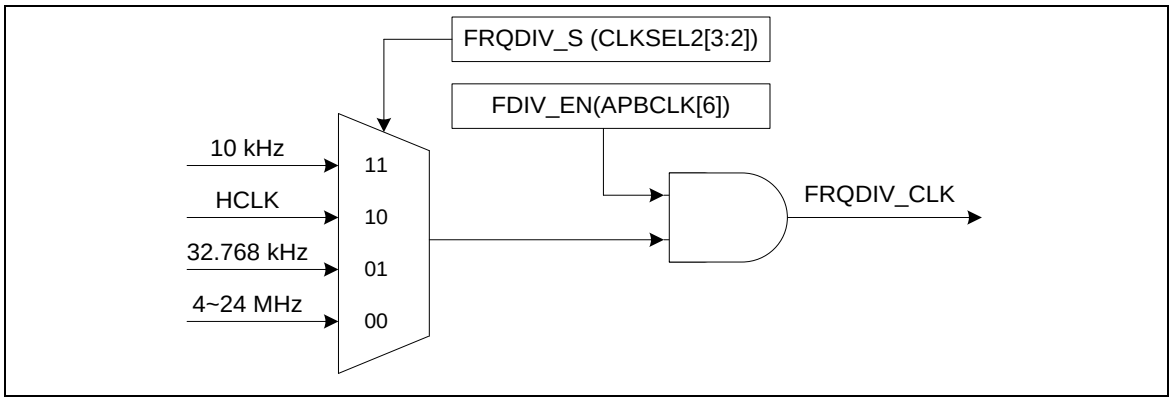


Figure 5-8 Clock Source of Frequency Divider

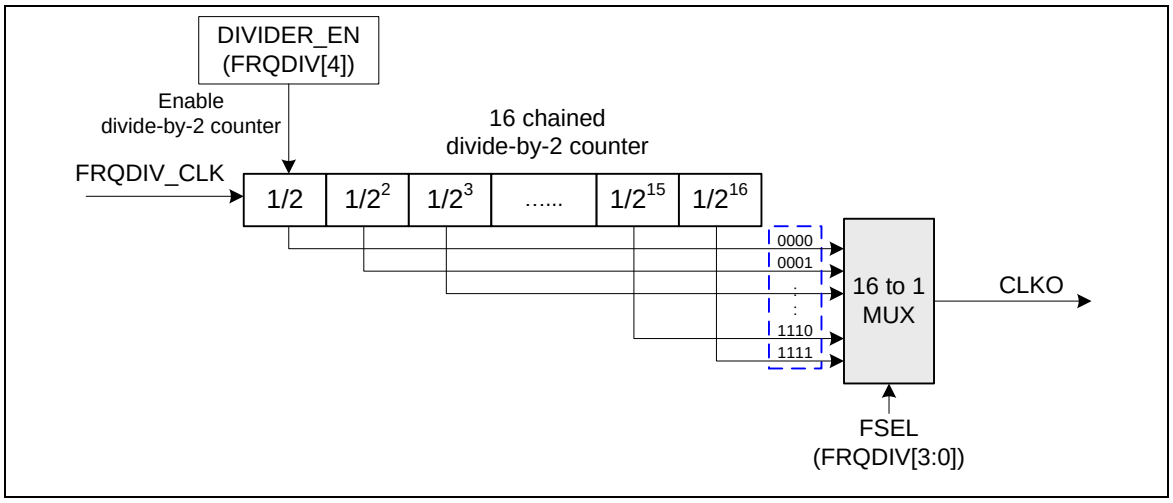


Figure 5-9 Frequency Divider Block Diagram

5.4 USB Device Controller (USB)

5.4.1 Overview

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

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

There are 6 endpoints in this controller. Each of the endpoint can be configured as IN or OUT endpoint. All the operations including Control, Bulk, Interrupt and Isochronous transfer are implemented in this block. The block of ENDPOINT CONTROL is also used to manage the data sequential synchronization, endpoint states, current start address, transaction status, and data buffer status for each endpoint.

There are four different interrupt events in this controller. They are the wake-up function, device plug-in or plug-out event, USB events, e.g. IN ACK, OUT ACK, and BUS events, e.g. suspend and resume. Any event will cause an interrupt, and users just need to check the related event flags in interrupt event status register (USB_INTSTS) to acknowledge what kind of interrupt occurring, and then check the related USB Endpoint Status Register (USB_EPSTS) to acknowledge what kind of event occurring in this endpoint.

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

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

5.4.2 Features

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

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

5.4.3 Block Diagram

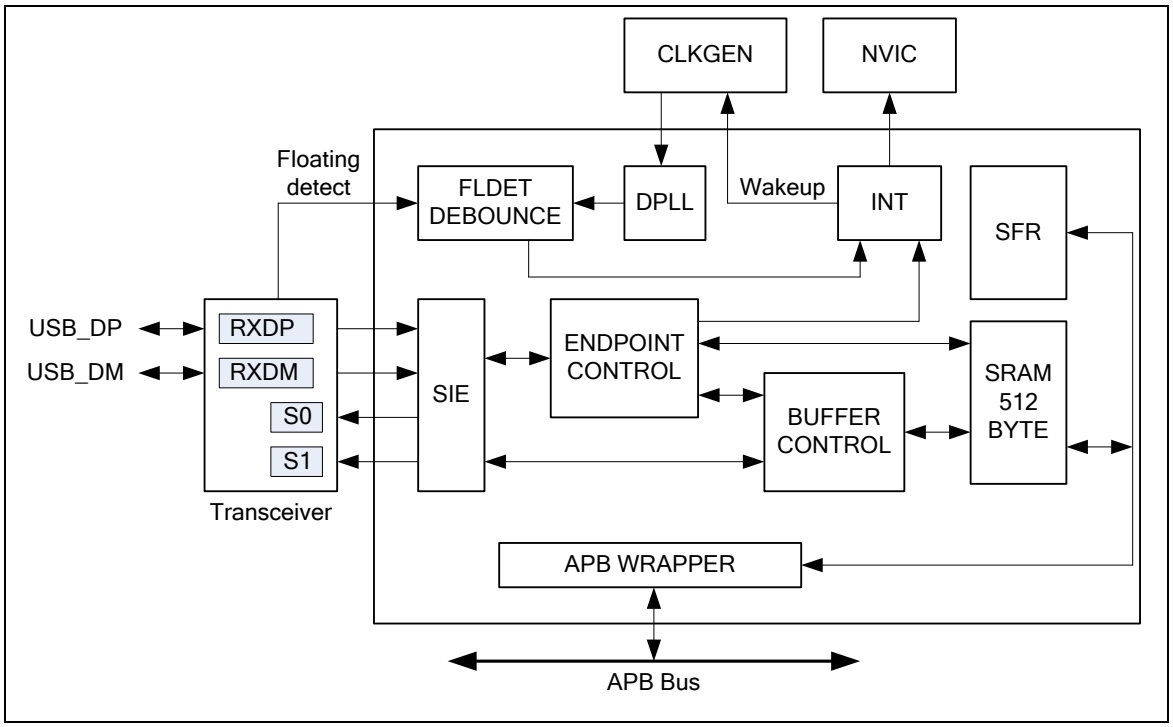


Figure 5-10 USB Block Diagram

5.4.4 Functional Description

5.4.4.1 *SIE (Serial Interface Engine)*

The SIE is the front-end of the device controller and handles most of the USB packet protocol. The SIE typically comprehends signaling up to the transaction level. The functions that it handles could include:

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

5.4.4.2 *Endpoint Control*

There are 6 endpoints in this controller. Each of the endpoint can be configured as Control, Bulk, Interrupt, or Isochronous transfer type. All the operations including Control, Bulk, Interrupt and Isochronous transfer are implemented in this block. It is also used to manage the data sequential synchronization, endpoint state control, current endpoint start address, current transaction status, and data buffer status in each endpoint.

5.4.4.3 *Digital Phase Lock Loop*

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

5.4.4.4 *Floating De-bounce*

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

5.4.4.5 *Interrupt*

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

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

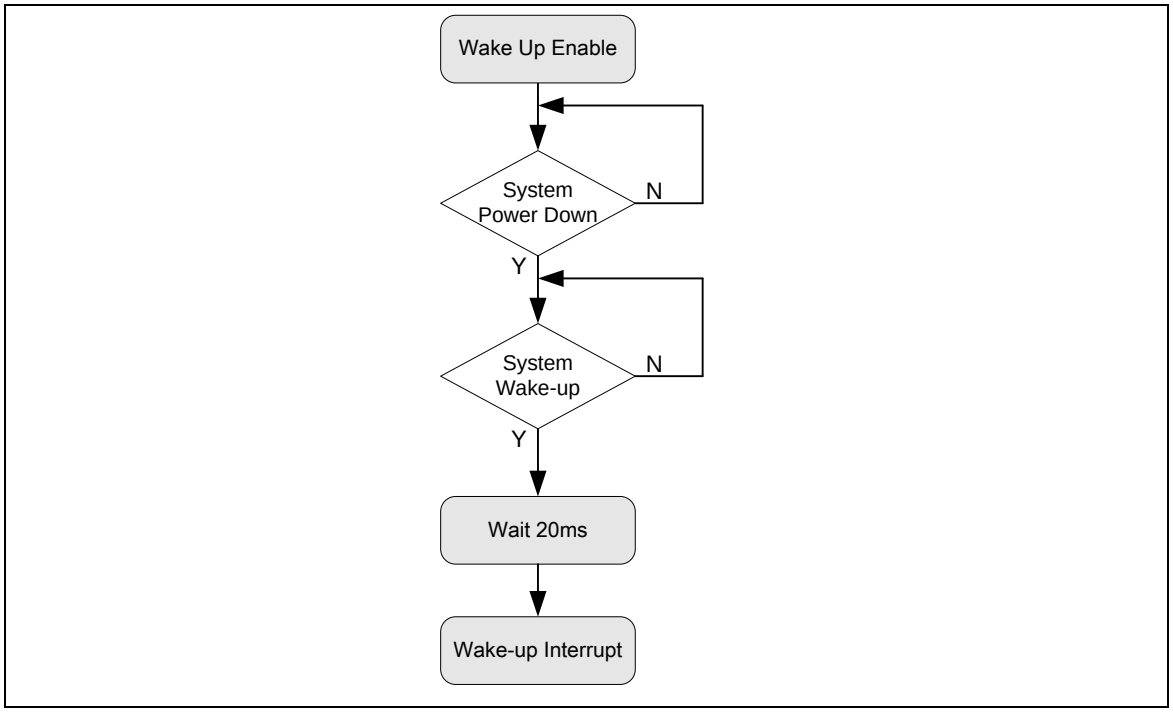


Figure 5-11 Wake-up Interrupt Operation Flow

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

Same as USB interrupt, BUS interrupt notifies users of some bus events, like USB reset, suspend, time-out, and resume. A user can read USB_ATTR to acknowledge bus events.

5.4.4.6 *Power Saving*

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

5.4.4.7 *Buffer Control*

There is 512 bytes SRAM in the controller and the 6 endpoints share this buffer. User shall configure each endpoint's effective starting address in the buffer segmentation register before the USB function active. The BUFFER CONTROL block is used to control each endpoint's effective starting address and its SRAM size is defined in the MXPLD register.

Figure 5-12 depicts the starting address for each endpoint according the content of USB_BUFSEGx and USB_MXPLDx registers. If the USB_BUFSEG0 is programmed as 0x08h and USB_MXPLD0 is set as 0x40h, the SRAM size of endpoint 0 is start from USB_BA+0x108h and end in USB_BA+0x148h. (**Note:** The USB SRAM base is USB_BA+0x100h).

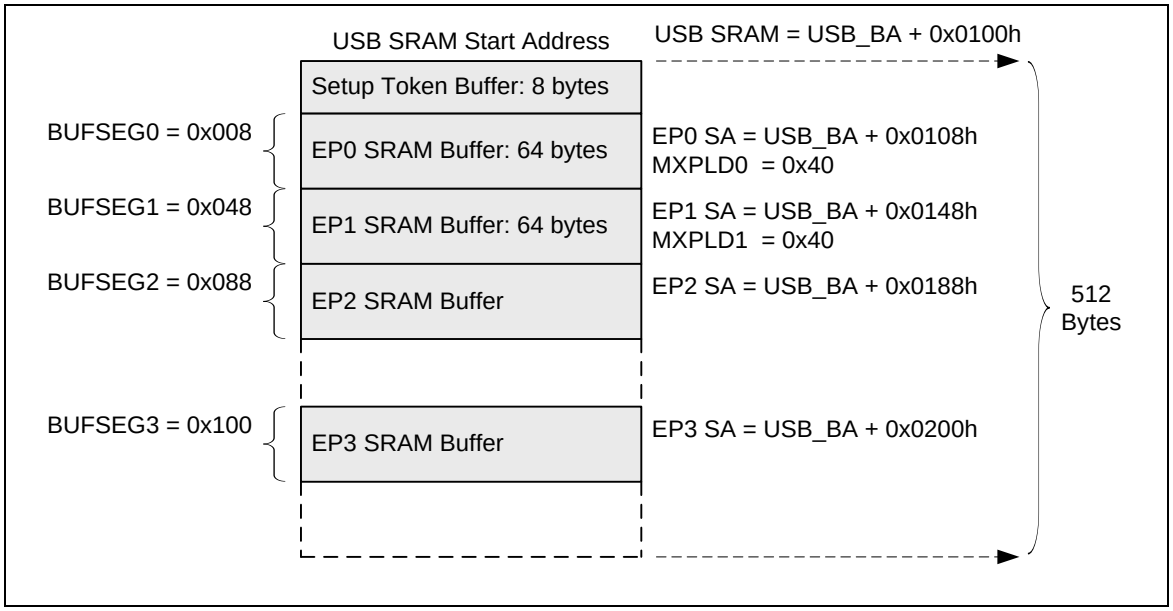


Figure 5-12 Endpoint SRAM Structure

5.4.4.8 *Handling Transactions with USB Device Peripheral*

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

When USB host has requested data from a device controller, user needs to prepare related data in the specified endpoint buffer in advance. After buffering the required data, user needs to write the actual data length in the specified MAXPLD register. Once this register is written, the internal signal "In_Rdy" will be asserted and the buffering data will be transmitted immediately after receiving associated IN token from Host. Note that after transferring the specified data, the signal "In_Rdy" will de-assert automatically by hardware.

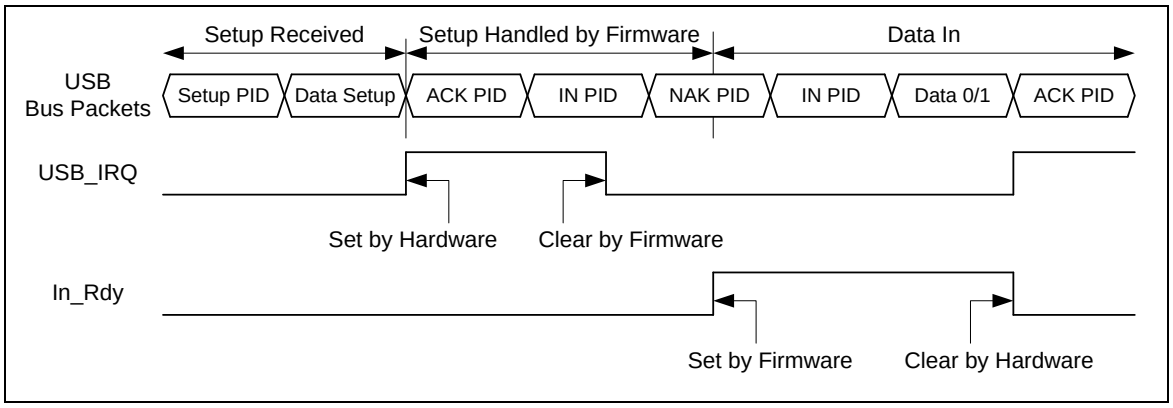


Figure 5-13 Setup Transaction followed by Data in Transaction

Alternatively, when USB host wants to transmit data to the OUT endpoint in the device controller, hardware will buffer these data to the specified endpoint buffer. After this transaction is completed, hardware will record the data length in related MAXPLD register and de-assert the signal "Out_Rdy". This will avoid hardware accepting next transaction until user moves out the current data in the related endpoint buffer. Once users have processed this transaction, the related register "MAXPLD" needs to be written by firmware to assert the signal "Out_Rdy" again to accept the next transaction.

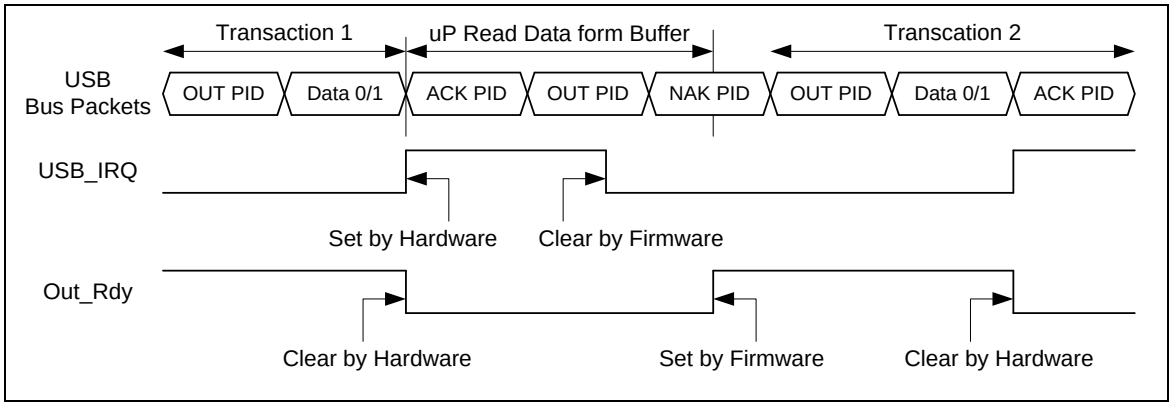


Figure 5-14 Data Out Transfer

5.5 General Purpose I/O (GPIO)

5.5.1 Overview

The NuMicro™ NUC200 series has up to 80 General Purpose I/O pins to be shared with other function pins depending on the chip configuration. These 80 pins are arranged in 6 ports named as GPIOA, GPIOB, GPIOC, GPIOD, GPIOE and GPIOF. The GPIOA/B/C/D/E port has the maximum of 16 pins and GPIOF port has the maximum of 4 pins. Each of the 80 pins is independent and has the corresponding register bits to control the pin mode function and data.

The I/O type of each of I/O pins can be configured by software individually as input, output, open-drain or Quasi-bidirectional mode. After reset, the I/O mode of all pins are depending on Config0[10] setting. 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.

5.5.2 Features

- Four I/O modes:
  - Quasi-bidirectional
  - Push-pull output
  - Open-Drain output
  - Input only with high impendence
- TTL/Schmitt trigger input selectable by GPx_TYPE[15:0] in GPx_MFP[31:16]
- I/O pin configured as interrupt source with edge/level setting
- Configurable default I/O mode of all pins after reset by Config0[10] setting
  - If Config[10] is 0, all GPIO pins in input tri-state mode after chip reset
  - If Config[10] is 1, all GPIO pins in Quasi-bidirectional mode after chip reset
- I/O pin internal pull-up resistor enabled only in Quasi-bidirectional I/O mode
- Enabling the pin interrupt function will also enable the pin wake-up function.

5.5.3 Functional Description

5.5.3.1 *Input Mode Explanation*

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

5.5.3.2 *Push-pull Output Mode Explanation*

Set GPIOx_PMD (PMDn[1:0]) to 01b as the GPIOx 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 GPIOx_DOUT is driven on the pin.

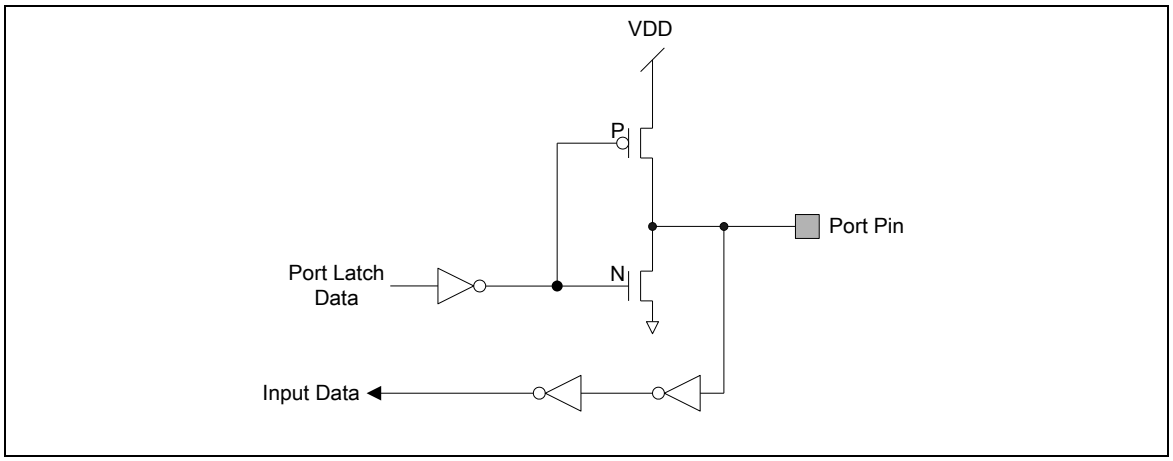


Figure 5-15 Push-Pull Output

5.5.3.3 *Open-drain Output Mode Explanation*

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

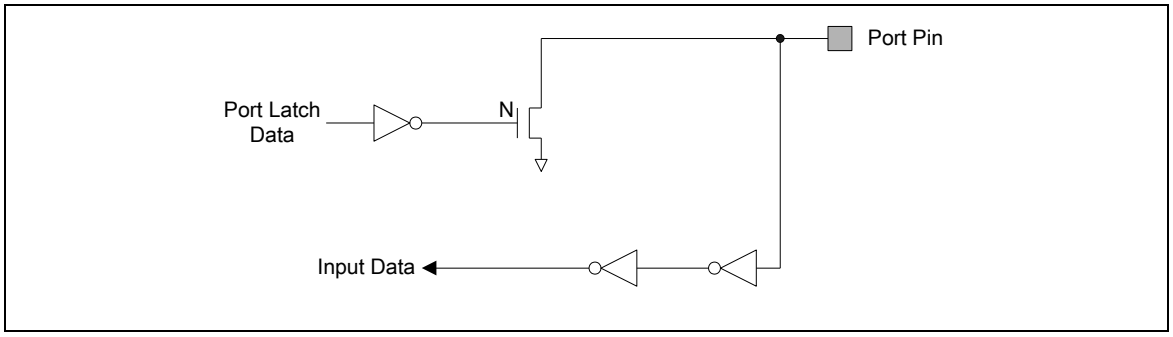


Figure 5-16 Open-Drain Output

5.5.3.4 *Quasi-bidirectional Mode Explanation*

Set GPIOx_PMD (PMDn[1:0]) to 11b as the GPIOx port [n] pin is in Quasi-bidirectional mode and the I/O pin supports digital output and input function at the same time but the source current is only up to hundreds of uA. Before the digital input function is performed the corresponding bit in GPIOx_DOUT must be set to 1. The Quasi-bidirectional output is common on the 80C51 and most of its derivatives. If the bit value in the corresponding bit [n] of GPIOx_DOUT is 0, the pin drive a “low” output on the pin. If the bit value in the corresponding bit [n] of GPIOx_DOUT is 1, the pin will check the pin value. If pin value is high, no action takes. If pin state is low, then pin will drive strong high with 2 clock cycles on the pin and then disable the strong output drive and then the pin status is control by internal pull-up resistor. Note that the source current capability in Quasi-bidirectional mode is only about 200 uA to 30 uA for V_{DD} is form 5.0 V to 2.5 V.

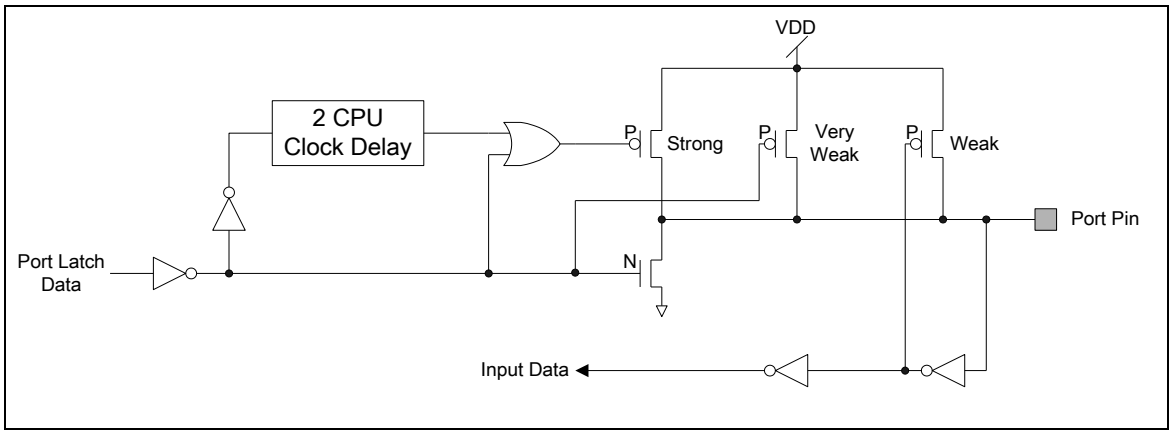


Figure 5-17 Quasi-bidirectional I/O Mode

5.5.3.5 *GPIO Interrupt and Wake-up Function*

Each GPIO pin can be set as chip interrupt source by setting correlative GPIOx_IEN bit and GPIOx_IMD. There are four types of interrupt condition can be selected: low level trigger, high level trigger, falling edge trigger and rising 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 is one thing need to be noticed if using GPIO as chip wake-up source

1. **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.

5.6 I²C Serial Interface Controller (I²C)

5.6.1 Overview

I²C is a two-wire, bidirectional serial bus that provides a simple and efficient method of data exchange between devices. The I²C standard is a true multi-master bus including collision detection and arbitration that prevents data corruption if two or more masters attempt to control the bus simultaneously.

Data is transferred between a Master and a Slave. 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 Figure 5-18 for more detailed I²C BUS Timing.

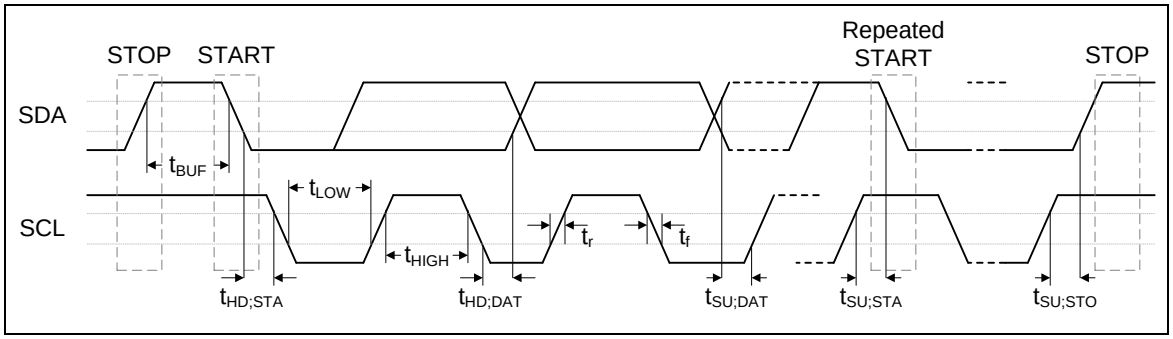


Figure 5-18 I²C Bus Timing

The device's on-chip I²C logic provides a serial interface that meets the I²C bus standard mode specification. The I²C port handles byte transfers autonomously. To enable this port, the bit ENS1 in I2CON should be set to '1'. The I²C H/W interfaces to the I²C bus via two pins: SDA and SCL. Pull-up resistor is needed for I²C operation as the SDA and SCL are open drain pins. When I/O pins are used as I²C ports, user must set the pins function to I²C in advance.

5.6.2 Features

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

- Master/Slave mode
- Bidirectional data transfer between masters and slaves
- Multi-master bus (no central master)
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- Serial clock synchronization allowing devices with different bit rates to communicate via one serial bus
- Serial clock synchronization used as a handshake mechanism to suspend and resume serial transfer
- A built-in 14-bit time-out counter requesting the I²C interrupt if the I²C bus hangs up and timer-out counter overflows.
- External pull-up resistors needed for high output
- Programmable clocks allowing for versatile rate control
- Supports 7-bit addressing mode
- Supports multiple address recognition (four slave addresses with mask option)

5.6.3 Functional Description

5.6.3.1 *I²C Protocol*

Normally, a standard communication consists of four parts:

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

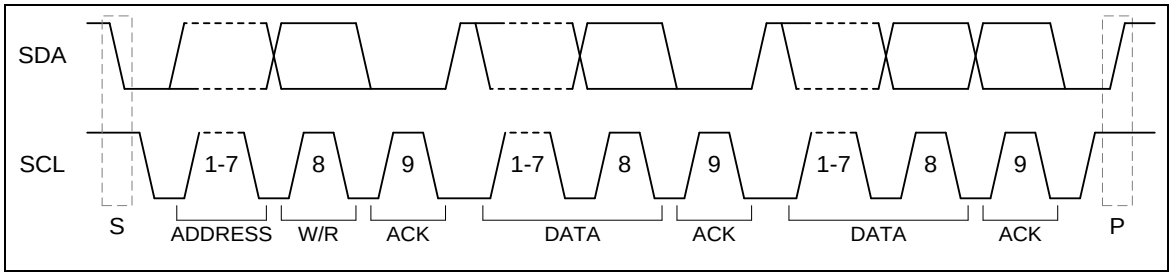


Figure 5-19 I²C Protocol

5.6.3.2 *Data transfer on the I²C-bus*

Figure 5-20 shows a master transmits data to slave. A master addresses a slave with a 7-bit address and 1-bit write index to denote that the master wants to transmit data to the slave. The master keeps transmitting data after the slave returns acknowledge to the master.

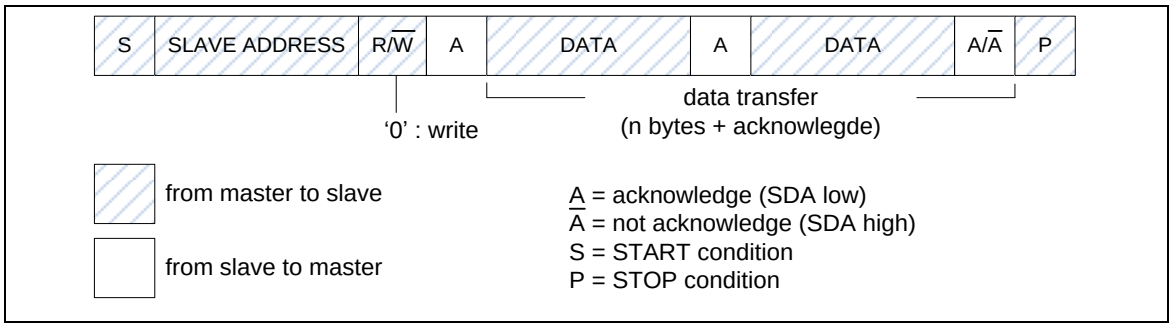


Figure 5-20 Master Transmits Data to Slave

Figure 5-21 shows a master read data from slave. A master addresses a slave with a 7-bit address and 1-bit read index to denote that the master wants to read data from the slave. The slave will start transmitting data after the slave returns acknowledge to the master.

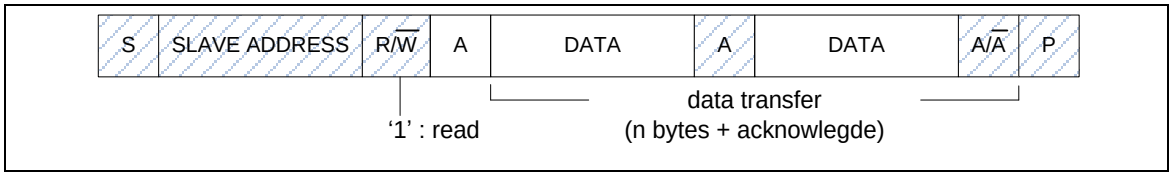


Figure 5-21 Master Reads Data from Slave

5.6.3.3 *START or Repeated START signal*

When the bus is free/idle, meaning no master device is engaging the bus (both SCL and SDA lines are high), a master can initiate a transfer by sending a START signal. A START signal, usually referred to as the S-bit, is defined as a HIGH to LOW transition on the SDA line while SCL is HIGH. The START signal denotes the beginning of a new data transmission.

A Repeated START (Sr) is not a STOP signal between two START signals. The master uses this method to communicate with another slave or the same slave in a different transfer direction (e.g. from writing to a device to reading from a device) without releasing the bus.

STOP signal

The master can terminate the communication by generating a STOP signal. A STOP signal, usually referred to as the P-bit, is defined as a LOW to HIGH transition on the SDA line while SCL is HIGH.

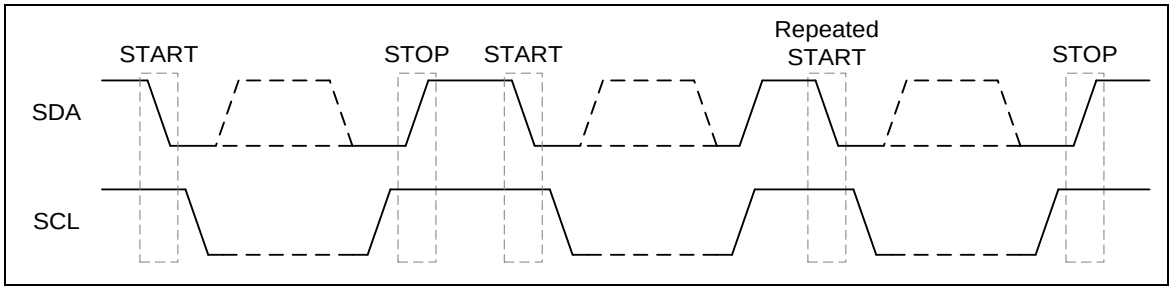


Figure 5-22 START and STOP Conditions

5.6.3.4 *Slave Address Transfer*

The first byte of data transferred by the master immediately after the START signal is the slave address. This is a 7-bit calling address followed by a RW bit. The RW bit signals the slave the data transfer direction. No two slaves in the system can have the same address. Only the slave with an address that matches the one transmitted by the master will respond by returning an acknowledge bit by pulling the SDA low at the 9th SCL clock cycle.

5.6.3.5 *Data Transfer*

When a slave receives a correct address with a RW bit, the data will follow RW bit specified to transfer. Each transferred byte is followed by an acknowledge bit on the 9th SCL clock cycle. If the slave signals a Not Acknowledge (NACK), the master can generate a STOP signal to abort the data transfer or generate a Repeated START signal and start a new transfer cycle.

If the master, as a receiving device, does Not Acknowledge (NACK) the slave, the slave releases the SDA line for the master to generate a STOP or Repeated START signal.

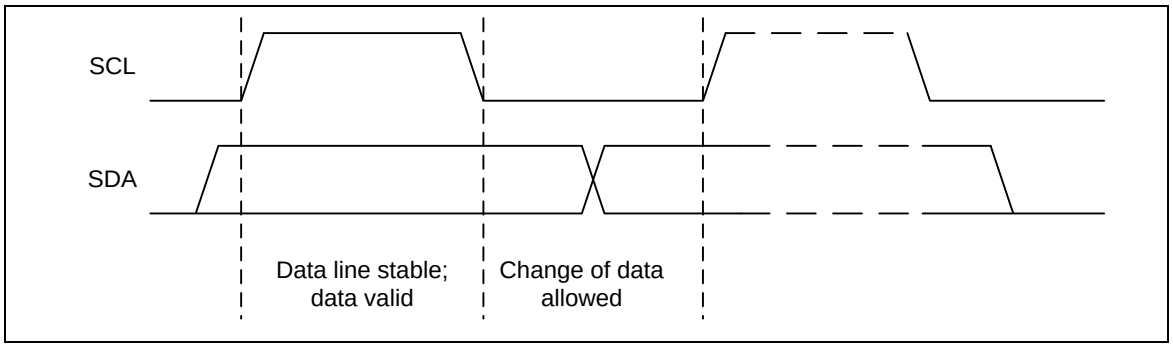


Figure 5-23 Bit Transfer on I²C Bus

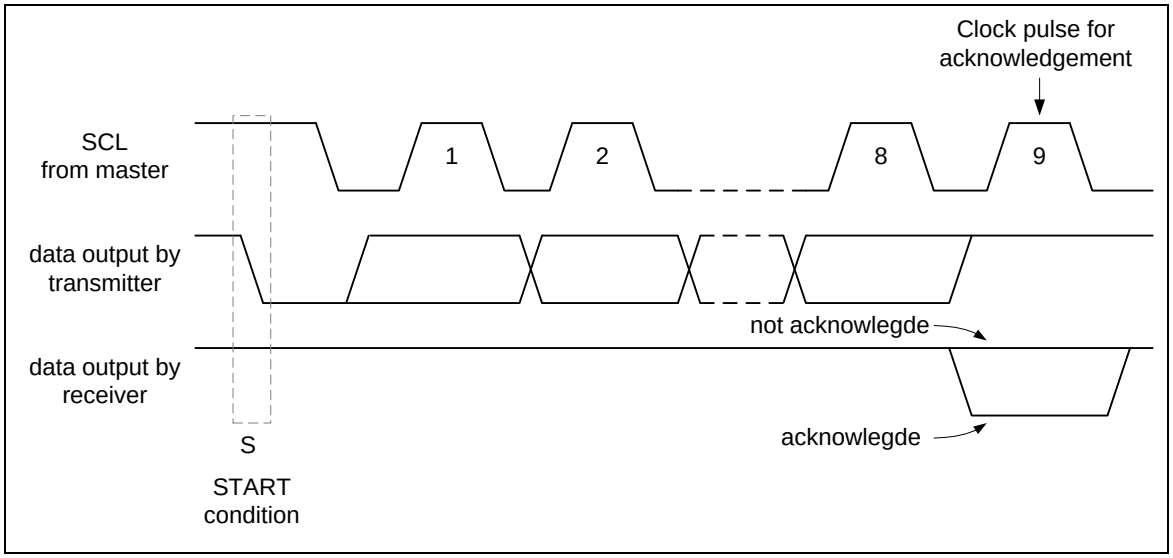


Figure 5-24 Acknowledge on I²C Bus

5.6.4 Protocol Registers

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

When the I²C port is enabled by setting ENS1 (I2CON [6]) to high, the internal states will be controlled by I2CON and I²C logic hardware. Once a new status code is generated and stored in I2CSTATUS, the I²C Interrupt Flag bit SI (I2CON [3]) will be set automatically. If the Enable Interrupt bit EI (I2CON [7]) is set at this time, the I²C interrupt will be generated. The bit field I2CSTATUS[7:3] stores the internal state code, the lowest 3 bits of I2CSTATUS are always zero and the content keeps stable until SI is cleared by software. The base address is 0x4002_0000 and 0x4012_0000.

5.6.4.1 *Address Registers (I2CADDR)*

The 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 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 the 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.

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 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 exactly the same as address register.

5.6.4.2 *Data Register (I2CDAT)*

This register contains a byte of serial data to be transmitted or a byte which just has been received. The CPU can be read from or written to the 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 presented 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 to I2CDATA[7:0] when sending I2CDATA[7:0] to bus. In the case of sending data, serial data bits are shifted out from I2CDAT [7:0] on the falling edge of SCL clocks, and is shifted to I2CDAT [7:0] on the rising edge of SCL clocks.

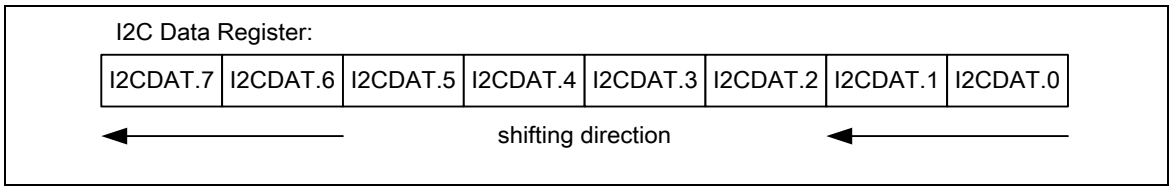


Figure 5-25 I²C Data Shifting Direction

5.6.4.3 *Control Register (I2CON)*

The CPU can be read from and written 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.

EI Enable Interrupt.

ENS1 Set to enable I²C serial function controller. When ENS1=1 the I²C serial function is enabled. The function of multi-function pins must be set to I²C.

STA I²C START Control Bit. Set STA to logic 1 to enter Master mode, and the I²C hardware sends a START or repeat START condition to bus when the bus is free.

STO I²C STOP Control Bit. In Master mode, set STO to transmit a STOP condition to bus, then I²C hardware will check the bus condition if a STOP condition is detected this flag will be cleared by hardware automatically. In Slave mode, set STO resets I²C hardware to the defined “not addressed” Slave mode. This means it is NO LONGER in Slave Receiver mode to receive data from the master.

SI I²C Interrupt Flag. When a new I²C state is present in the I2CSTATUS register, the SI flag is set by hardware, and if the bit EI (I2CON [7]) is set, the I²C interrupt is requested. SI must be cleared by software. Clear SI by writing 1 to this bit. All states are listed in section 5.6.6.

AA Assert Acknowledge Control Bit. When AA=1 prior to address or data received, an acknowledged (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line when 1.) A slave is acknowledging the address sent from the master, 2.) The receiver devices are acknowledging the data sent by a transmitter. When AA=0 prior to address or data received, a Not acknowledged (high level to SDA) will be returned during the acknowledge clock pulse on the SCL line.

5.6.4.4 *Status Register (I2CSTATUS)*

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

In addition, the state 00H stands for a Bus Error, which occurs when a START or STOP condition is present at an incorrect position in the I^2C format frame. A Bus Error may occur during the serial transfer of an address byte, a data byte or an acknowledge bit. To recover I^2C from bus error, STO should be set and SI should be cleared to enter Not Addressed Slave mode. Then STO is cleared to release bus and to wait for a new communication. The I^2C bus cannot recognize stop condition during this action when a bus error occurs.

Master Mode		**Slave Mode**	
STATUS	**Description**	**STATUS**	**Description**
0x08	Start	0xA0	Slave Transmit Repeat Start or Stop
0x10	Master Repeat Start	0xA8	Slave Transmit Address ACK
0x18	Master Transmit Address ACK	0xB0	Slave Transmit Arbitration Lost
0x20	Master Transmit Address NACK	0xB8	Slave Transmit Data ACK
0x28	Master Transmit Data ACK	0xC0	Slave Transmit Data NACK
0x30	Master Transmit Data NACK	0xC8	Slave Transmit Last Data ACK
0x38	Master Arbitration Lost	0x60	Slave Receive Address ACK
0x40	Master Receive 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:** The status “0xF8” exists in both Master/Slave modes, and it will not raise any interrupt.		

Table 5-5 I^2C Status Code Description Table

5.6.4.5 *I²C Clock Baud Rate Bits (I2CLK)*

The data baud rate of I²C is determined by the I2CLK [7:0] register when I²C is in Master mode, and it is not necessary in Slave mode. In Slave mode, I²C will automatically synchronize it with any clock frequency from a 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), the data baud rate of I²C = 16 MHz/ (4x (40 +1)) = 97.5 Kbits/sec.

5.6.4.6 *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=1) and generates I²C interrupt to CPU or stops counting by clearing ENTI to 0. When time-out counter is enabled, writing 1 to the SI flag will reset the counter and re-start up counting after SI is cleared. If I²C bus hangs up, it causes the I2CSTATUS and flag SI are not updated for a period, the 14-bit time-out counter may overflow and acknowledge CPU the I²C interrupt. Refer to Figure 5-26 for the 14-bit time-out counter. User may write 1 to clear TIF to 0.

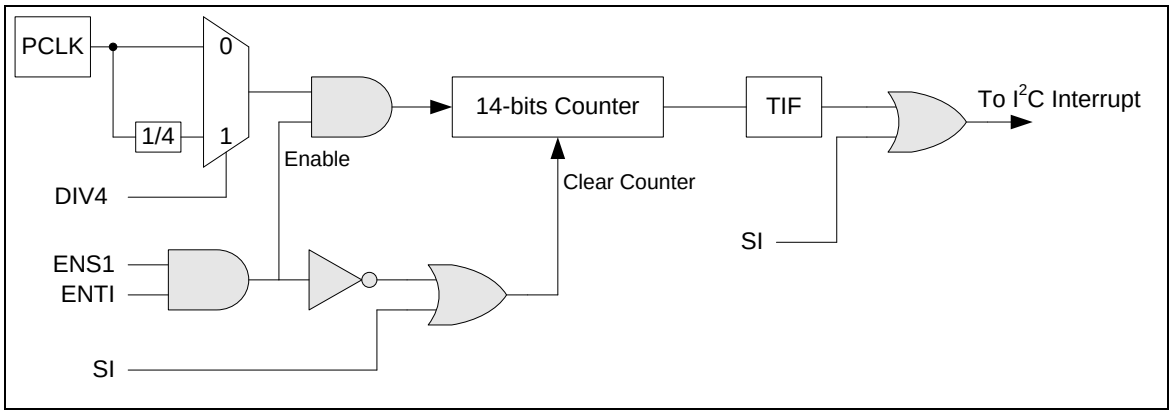


Figure 5-26 I²C Time-out Count Block Diagram

5.6.5 Operation Modes

The on-chip I^2C ports support five operation modes, Master Transmitter, Master Receiver, Slave Transmitter, Slave Receiver, and GC Call.

In a given application, I^2C port may operate as a master or as a slave. In Slave mode, the I^2C port hardware looks for its own slave address and the general call address. If one of these addresses is detected, and if the slave is willing to receive or transmit data from/to master (by setting the AA bit), acknowledge bit will be transmitted out on the 9th clock, hence an interrupt is requested on both master and slave devices if interrupt is enabled. When the microcontroller wishes to become the bus master, hardware waits until the bus is free before entering Master mode so that a possible slave action is not be interrupted. If bus arbitration is lost in Master mode, I^2C port switches to Slave mode immediately and can detect its own slave address in the same serial transfer.

Bits STA, STO and AA in I2CON register will determine the next state of the I^2C hardware after SI flag is cleared. Upon completion of the new action, a new status code will be updated and the SI flag will be set. If the I^2C 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.

In the following figure about the five operation modes, detailed data flow is represented. The legend for those data flow figures is shown in Figure 5-27

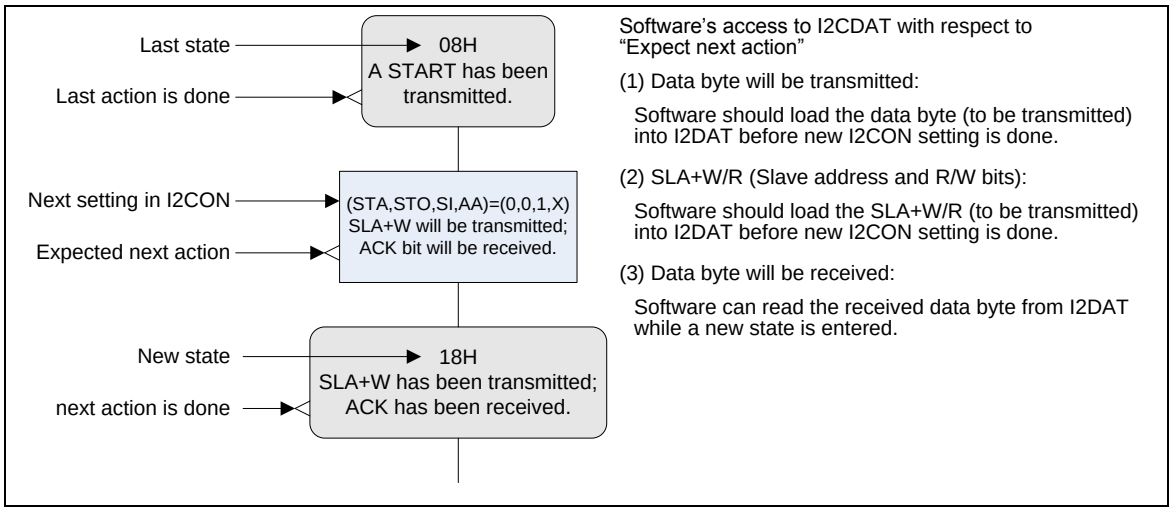


Figure 5-27 Legend for the Following Five Figures

5.6.5.1 *Master Transmitter Mode*

As shown in Figure 5-28, in Master Transmitter mode, serial data is output through SDA while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7-bit) and the data direction bit. In this case the data direction bit (R/W) will be logic 0, and it is represented by “W” in Figure 5-28. Thus the first byte transmitted is SLA+W. Serial data is transmitted 8-bit at a time. After each byte is transmitted, an acknowledge bit is received. START and STOP conditions are output to indicate the beginning and the end of a serial transfer.

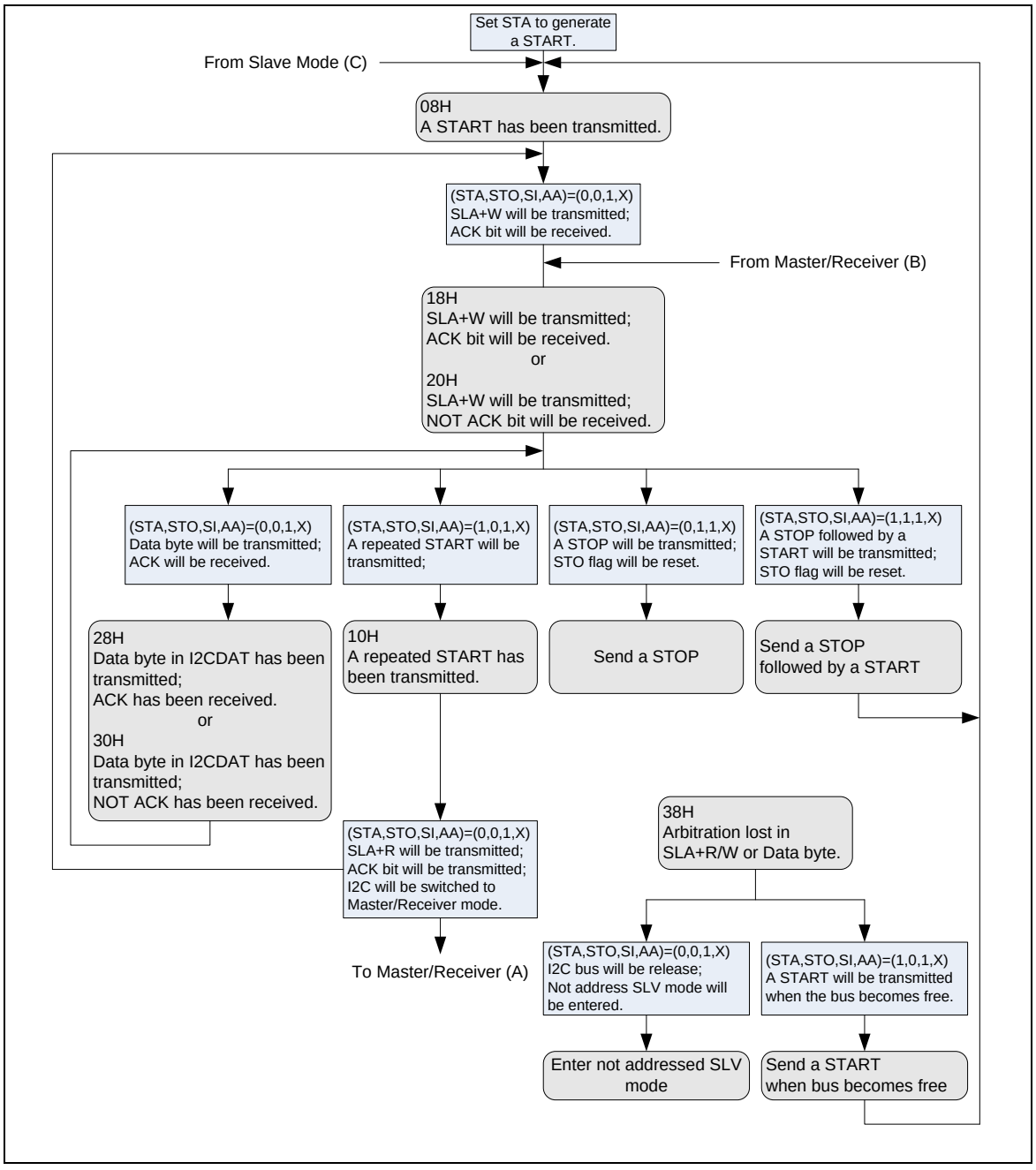


Figure 5-28 Master Transmitter Mode

5.6.5.2 *Master Receiver Mode*

As shown in Figure 5-29, in this case the data direction bit (R/W) will be logic 1, and it is represented by "R" in Figure 5-29. Thus the first byte transmitted is SLA+R. Serial data is received via SDA while SCL outputs the serial clock. Serial data is received 8-bit at a time. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are output to indicate the beginning and end of a serial transfer.

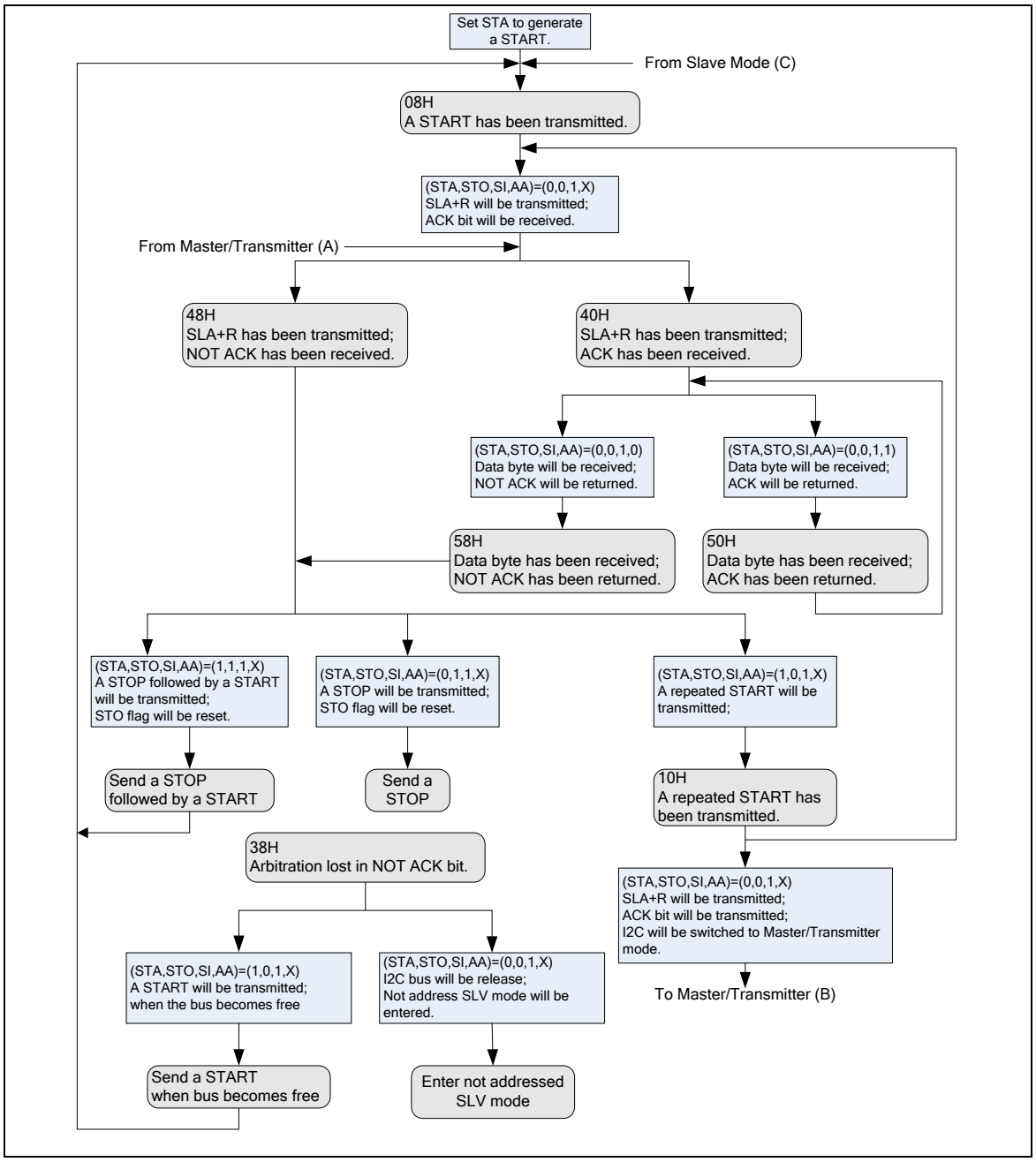


Figure 5-29 Master Receiver Mode

5.6.5.3 *Slave Receiver Mode*

As shown in Figure 5-30, serial data and the serial clock are received through SDA and SCL. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and direction bit.

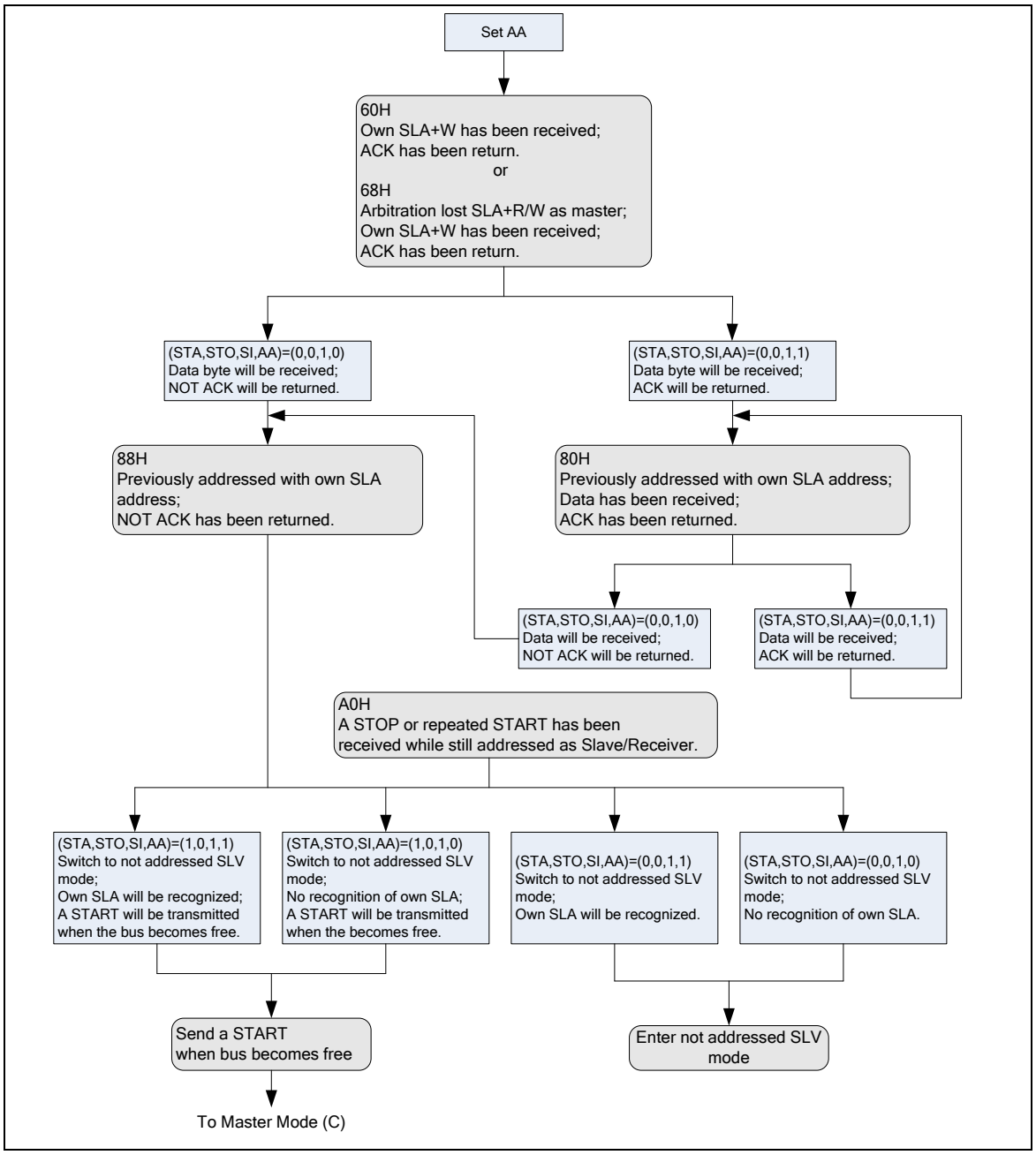


Figure 5-30 Slave Receiver Mode

5.6.5.4 *Slave Transmitter Mode*

As shown in Figure 5-31, the first byte is received and handled as in the slave receiver mode.

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

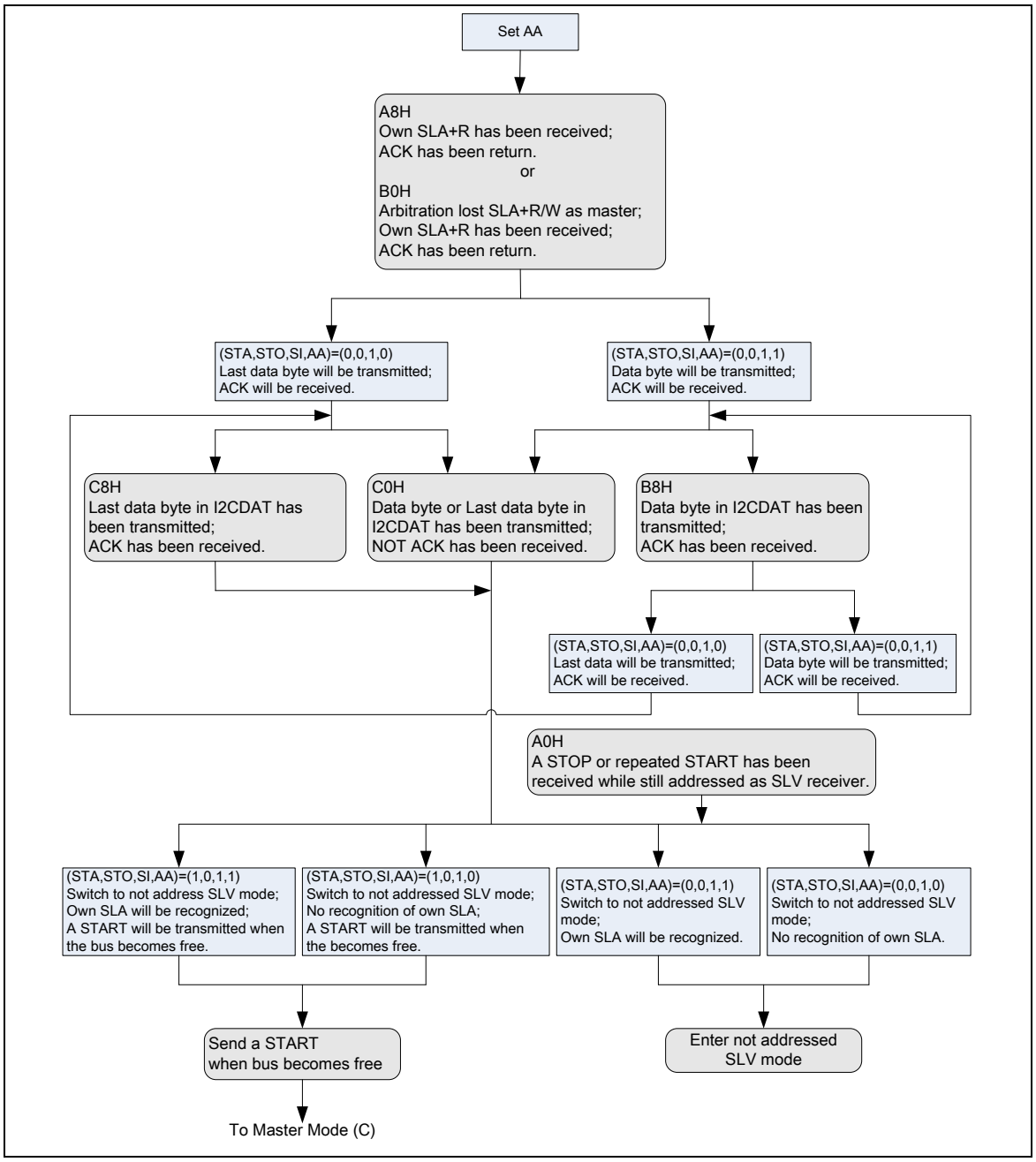


Figure 5-31 Slave Transmitter Mode

5.6.5.5 *General Call (GC) Mode*

As shown in Figure 5-32, if the GC bit (I2CADDRn [0]) is set, the I^2C port hardware will respond to General Call address (00H). Clear GC bit to disable general call function. When GC bit is set and the I^2C is in Slave mode, it can receive the general call address by 00H after Master send general call address to I^2C bus, then it will follow status of GC mode. Serial data and the serial clock are received through SDA and SCL. After each byte is received, an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and direction bit.

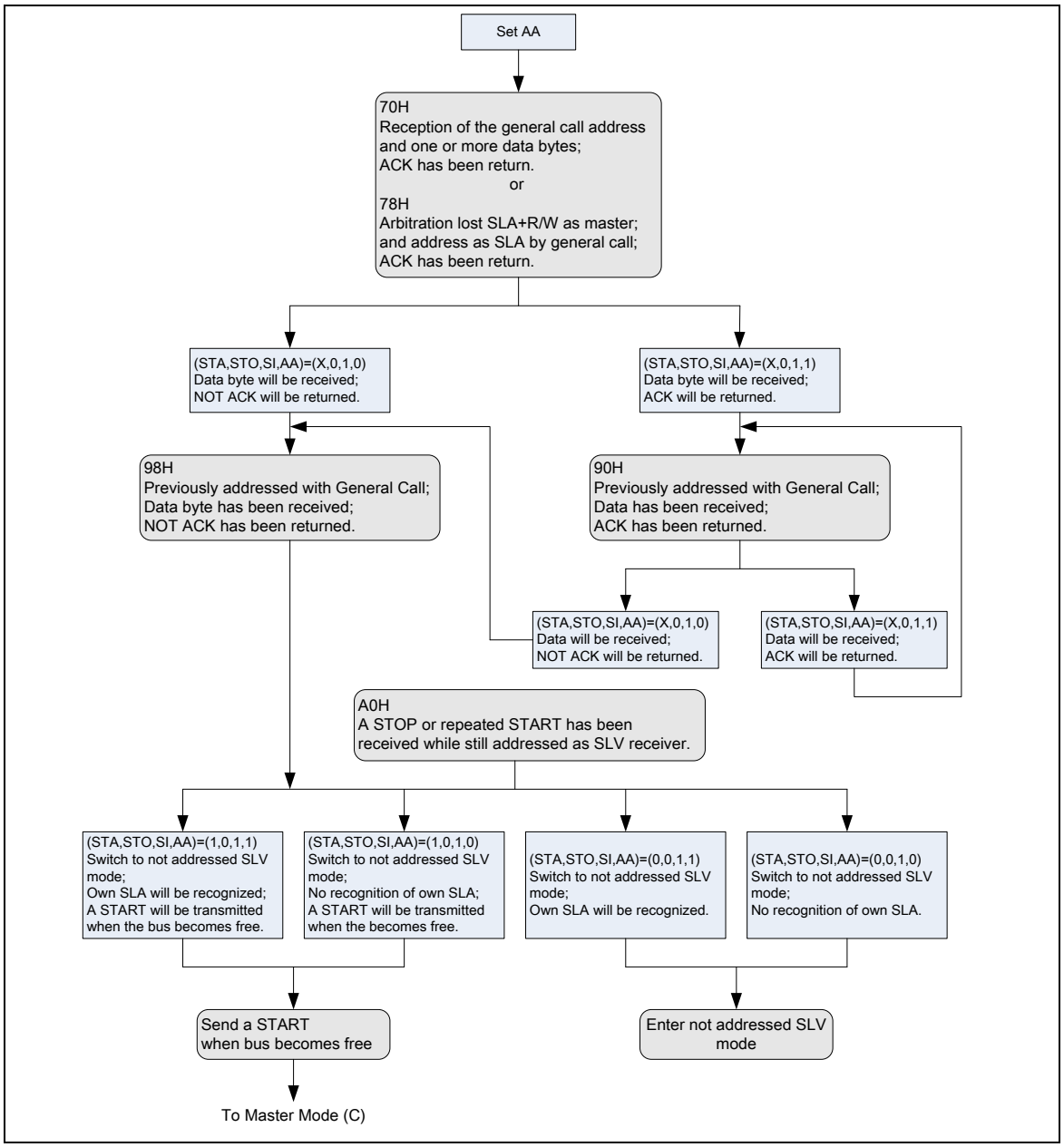


Figure 5-32 GC Mode

5.6.5.6 *Example for Random Read on EEPROM*

The following steps are used to configure the I²C related registers when using I²C to read data from EEPROM.

1. Set the multi-function pin in the “GPA_MFP” & “ALT_MFP” registers as I²C.
2. Enable I²C APB clock, I2C0_EN=1 and I2C1_EN=1, in the “APBCLK” register.
3. Set I2C0_RST=1 and I2C1_RST=1 to reset I²C controller then set I²C controller to normal operation, I2C0_RST=0 and I2C1_RST=0, in the “IPRSTC2” register
4. Set ENS1=1 to enable I²C controller in the “I2CON” register
5. Give I²C clock a divided register value for I²C clock rate in the “I2CLK”
6. Set SETENA=0x00040000 in the “NVIC_ISER” register to set I²C IRQ
7. Set EI=1 to enable I²C Interrupt in the “I2CON” register
8. Set I²C address registers which are “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. This operation has two processes in below example. One is “Transmitter Operation”, and another one is “Receive Operation”. Figure 5-33 shows the EEPROM random read operation.

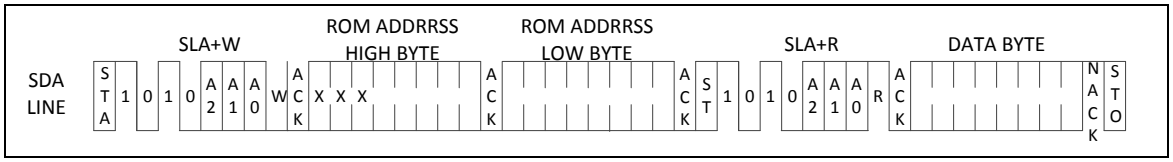


Figure 5-33 Random Read of EEPROM

■ **Transmitter Operation**

The I^2C controller sends a SLA(Slave address)+W to EERPOM first, then sends a word length (16 bits) data address to set the inside data pointer. In this operation steps can refer to Figure 5-34. The Transmitter Operation steps are as follows:

1. Send a START bit, and wait I2CSTATUS becomes to 0x08

   --- (STA, STO, SI, AA) = (1, 0, 1, X)

2. When I2CSTAUS = 0x08, the START bit has been transmitted. Then write a 7-bit EEPROM slave address with control bit (Write) to I2CDAT register and wait a response of ACK (Acknowledge bit) back from EEPROM.

   --- I2C0DAT=SLA+W, and clear SI (STA, STO, SI, AA) = (0, 0, 1, X);

3. When I2CSTAUS = 0x18, an ACK is received. Then write data pointer high byte to EEPROM and wait a response of ACK. If I2CSTATUS=0x20, an NACK is received it means that the EEPROM didn't get correct address or EEPROM is not exist. In this situation, send a STOP to terminate this communication.

   --- Get ACK (0x18): I2CDAT=Data pointer (Hi Byte) and clear SI (STA, STO, SI, AA) = (0,0,1,X);

   --- Get NACK (0x20): (STA, STO, SI, AA) = (0, 1, 1, X)

4. When I2CSTATUS=0x28, an ACK is received. Send data pointer low byte to EEPROM and wait next ACK back. If I2CSTATUS=0x30, I^2C should be terminate this communication.

   --- Get ACK (0x28): I2CDATA=Data pointer (Low Byte) and clear SI (STA, STO, SI, AA) = (0, 0, 1, X);

   --- Get NACK(0x30): (STA, STO, SI, AA) = (0, 1, 1, X)

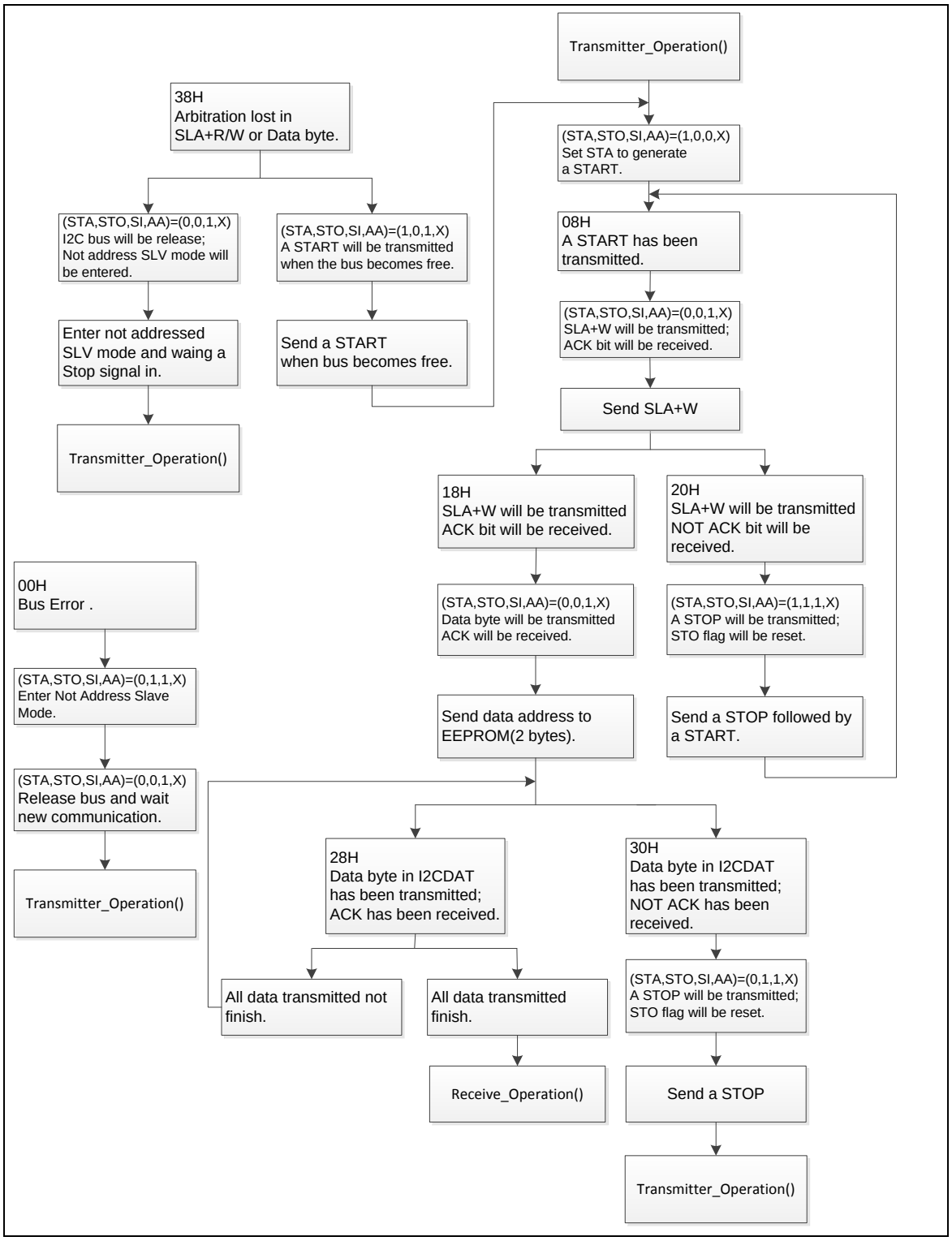


Figure 5-34 Transmitter Operation of Random Read

■ **Receive Operation**

Another one process of random read operation is “Receive Operation”. The I^2C controller sends a SLA+R to EERPOM then gets a data byte back where the data pointer pointed to. Receive operation is continuation of the “Transmitter Operation”. Refer to the following figure for the steps.

The Receive Operation steps are as follows:

1. When I2CSTATUS=0x28, send a RESTART bit. If I2CSTATUS=0x30, I^2C should be terminate this communication.

   --- Get ACK (0x28): (STA, STO, SI, AA) = (1, 0, 1, X)

   --- Get NACK (0x30): (STA, STO, SI, AA) = (0, 1, 1, X)

2. When I2CSTATUS=0x10, write an 7-bit EEPROM slave address with a control bit (Read) to I2CDATA register.

   --- I2CDATA=SLA+R, and clear SI (STA, STO, SI, AA) = (0, 0, 1, X)

3. When I2CSTATAUS=0x40, send a NACK. At this status, I^2C controller will send 8 clocks to receive EEPROM data back and at ninth clock send NACK out to EEPROM.

   --- (STA, STO, SI, AA) = (0, 0, 1, 0)

4. When I2CSTATUS=0x58, data is ready received, then send a STOP bit to terminate this communication.

   --- (STA, STO, SI, AA) = (0, 1, 1, 0)

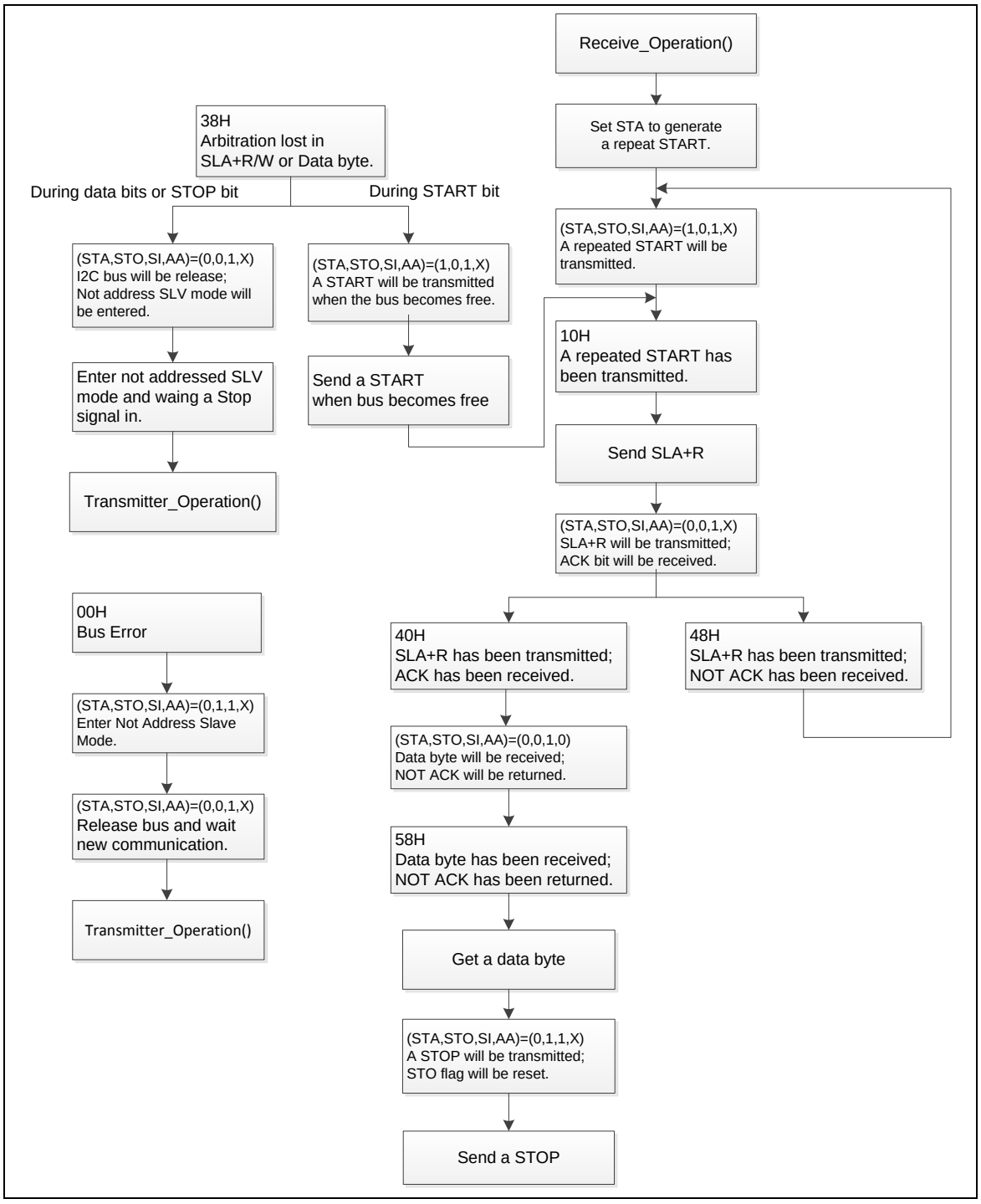


Figure 5-35 Receive Operation of Random Read

- **Other Events (Multi-Master)**

In some applications, there are two or more master microcontrollers on the same I^2C bus to access slaves, and the masters maybe transmit data simultaneously. The NuMicro™ microcontroller supports true multi-master bus including collision detection and arbitration to prevent data corruption.

1. When I2CSTATUS=0x38, an “Arbitration Lost” is received. Arbitration lost event maybe occur during the send START bit, data bits or STOP bit. In the process the START bit receive this status, I^2C can send start to try again. Exception during START bit, I^2C sends a STOP to terminate this communication.

   --- During START bit, (STA, STO, SI, AA) = (1, 0, 1, X)

   --- During data bits or STOP bit, (STA, STO, SI, AA) = (0, 0, 1, X)

2. When I2CSTATUS=0x00, a “Bus Error” is received. To recover I^2C bus from a bus error, STO should be set and SI should be cleared, and then STO is cleared to release bus.

   --- Send (STA, STO, SI, AA) = (0, 1, 1, X) first then send (STA, STO, SI, AA) = (0, 0, 1, X)

5.7 PWM Generator and Capture Timer (PWM)

5.7.1 Overview

The NuMicro™ NUC200 series has 2 sets of PWM group supporting a total of 4 sets of PWM generators that can be configured as 8 independent PWM outputs, PWM0~PWM7, or as 4 complementary PWM pairs, (PWM0, PWM1), (PWM2, PWM3), (PWM4, PWM5) and (PWM6, PWM7) with 4 programmable Dead-zone generators.

Each PWM generator has one 8-bit prescaler, one clock divider with 5 divided frequencies (1, 1/2, 1/4, 1/8, 1/16), two PWM Timers including two clock selectors, two 16-bit PWM counters for PWM period control, two 16-bit comparators for PWM duty control and one Dead-zone generator. The 4 sets of PWM generators provide eight independent PWM interrupt flags set by hardware when the corresponding PWM period down counter reaches 0. Each PWM interrupt source with its corresponding enable bit can cause CPU to request PWM interrupt. The PWM generators can be configured as one-shot mode to produce only one PWM cycle signal or auto-reload mode to output PWM waveform continuously.

When PCR.DZEN01 is set, PWM0 and PWM1 perform complementary PWM paired function; the paired PWM period, duty and Dead-time are determined by PWM0 timer and Dead-zone generator 0. Similarly, the complementary PWM pairs of (PWM2, PWM3), (PWM4, PWM5) and (PWM6, PWM7) are controlled by PWM2, PWM4 and PWM6 timers and Dead-zone generator 2, 4 and 6, respectively. Refer to Figure 5-36 and Figure 5-43 for the architecture of PWM Timers.

To prevent PWM driving output pin with unsteady waveform, the 16-bit period down counter and 16-bit comparator are implemented with double buffer. When user writes data to counter/comparator buffer registers the updated value will be load into the 16-bit down counter/ comparator at the time down counter reaching 0. The double buffering feature avoids glitch at PWM outputs.

When the 16-bit period down counter reaches 0, the interrupt request is generated. If PWM-timer is set as auto-reload mode, when the down counter reaches 0, it is reloaded with PWM Counter Register (CNRx) automatically then start decreasing, repeatedly. If the PWM-timer is set as one-shot mode, the down counter will stop and generate one interrupt request when it reaches 0.

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

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

The maximum captured frequency that PWM can capture is confined by the capture interrupt latency. When capture interrupt occurred, software will do at least three steps, including: Read PIIR to get interrupt source and Read CRLRx/CFLRx(x=0~3) to get capture value and finally write 1 to clear PIIR to 0. If interrupt latency will take time T0 to finish, the capture signal mustn't transition during this interval (T0). In this case, the maximum capture frequency will be 1/T0. For example:

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

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

5.7.2 Features

5.7.2.1 *PWM function:*

- Up to 2 PWM groups (PWMA/PWMB) to support 8 PWM channels or 4 complementary PWM paired channels
- Each PWM group has two PWM generators with each PWM generator supporting one 8-bit prescaler, two clock divider, two PWM-timers, one Dead-zone generator and two PWM outputs.
- Up to 16-bit resolution
- PWM Interrupt request synchronized with PWM period
- One-shot or Auto-reload mode PWM
- Edge-aligned type or Center-aligned type option

5.7.2.2 *Capture Function:*

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

5.7.3 Block Diagram

Figure 5-36 to Figure 5-43 illustrate the architecture of PWM in pair (e.g. PWM-Timer 0/1 are in one pair and PWM-Timer 2/3 are in another one).

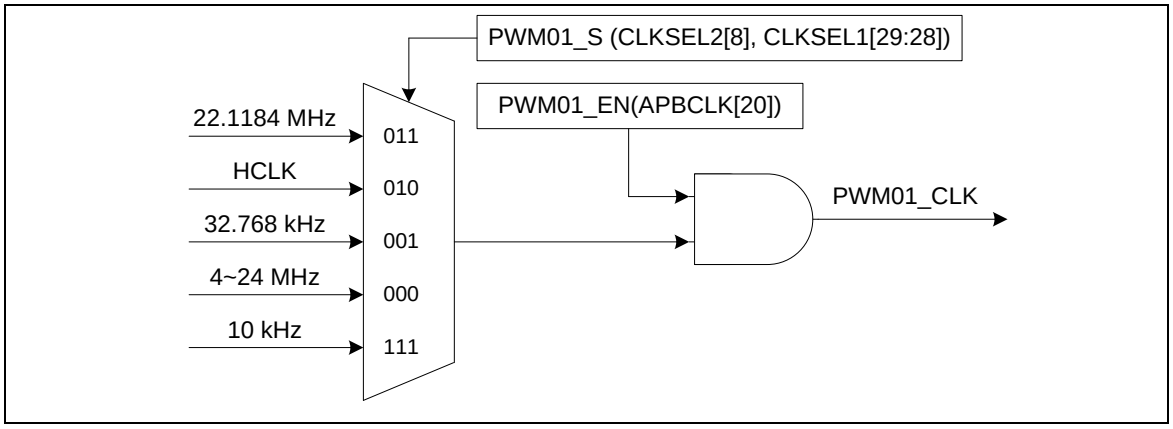


Figure 5-36 PWM Generator 0 Clock Source Control

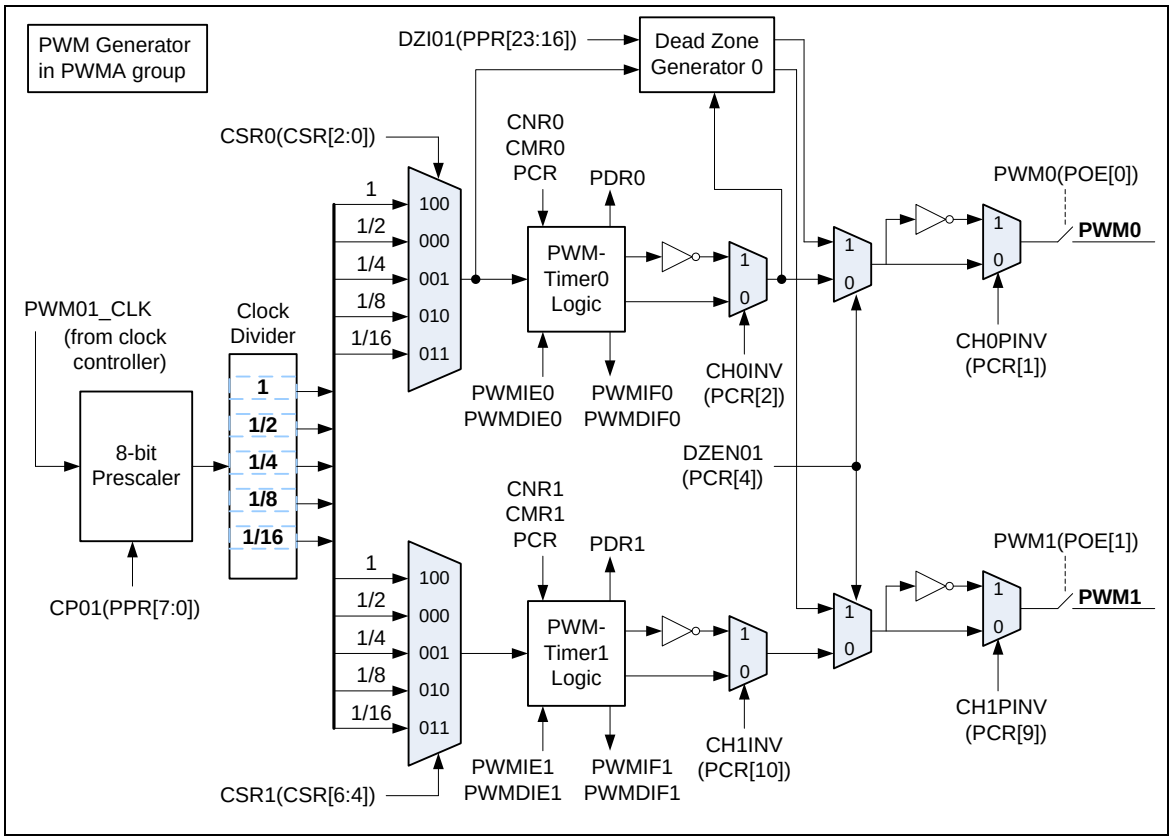


Figure 5-37 PWM Generator 0 Architecture Diagram

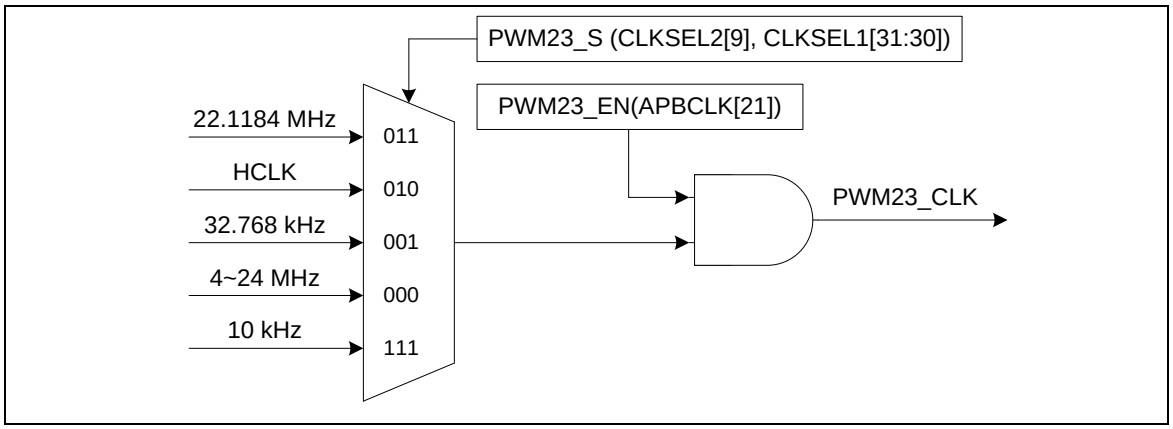


Figure 5-38 PWM Generator 2 Clock Source Control

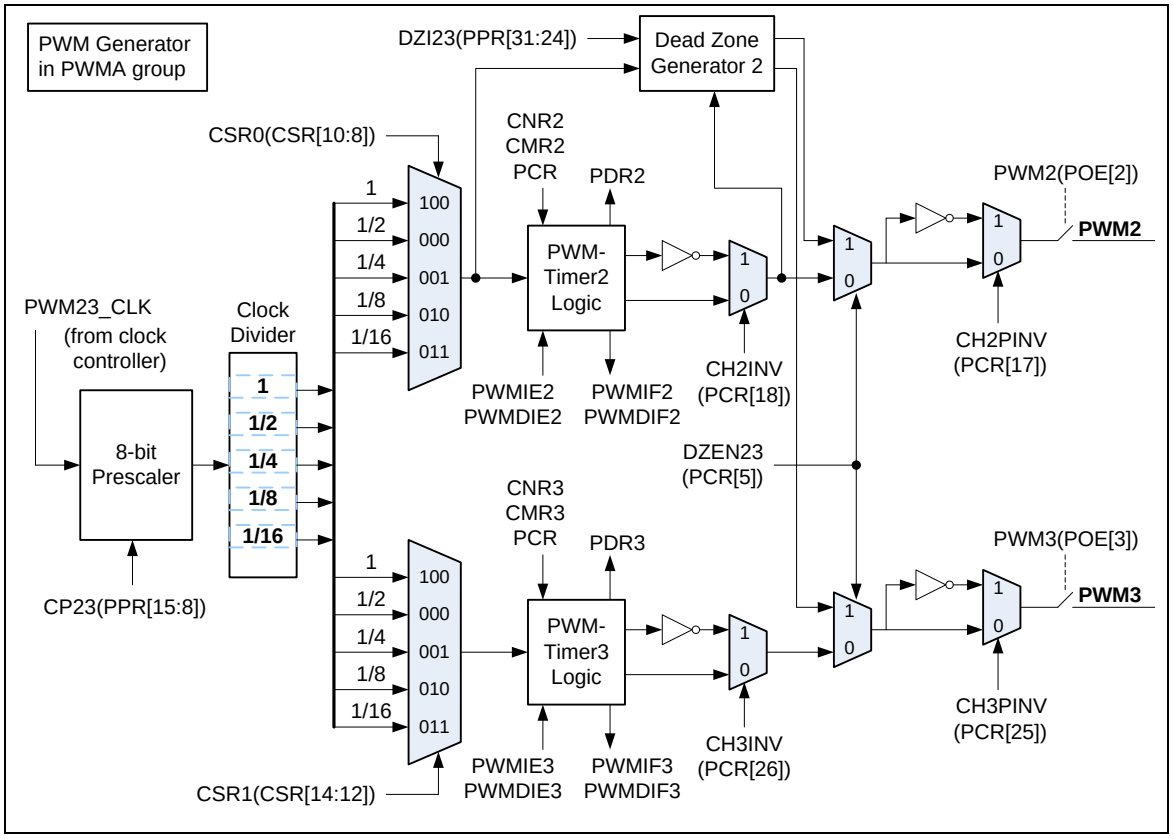


Figure 5-39 PWM Generator 2 Architecture Diagram

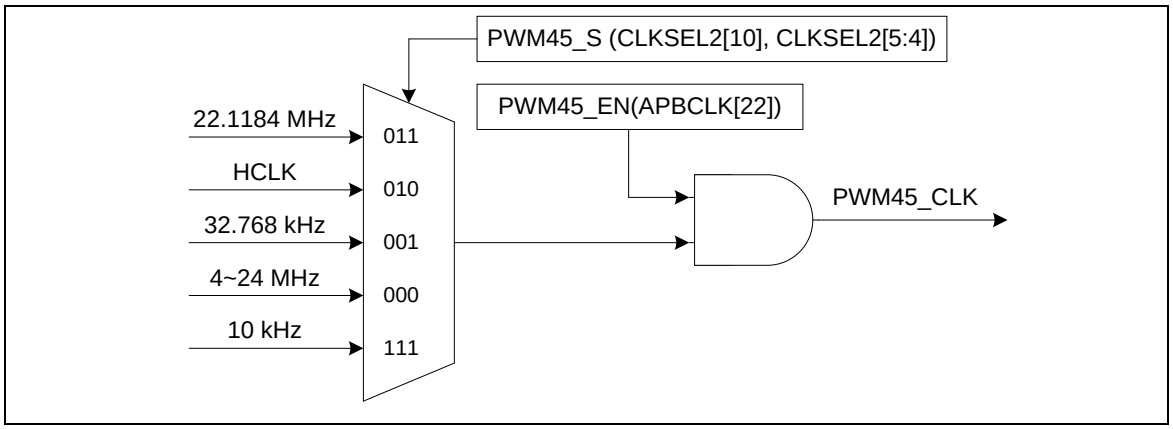


Figure 5-40 PWM Generator 4 Clock Source Control

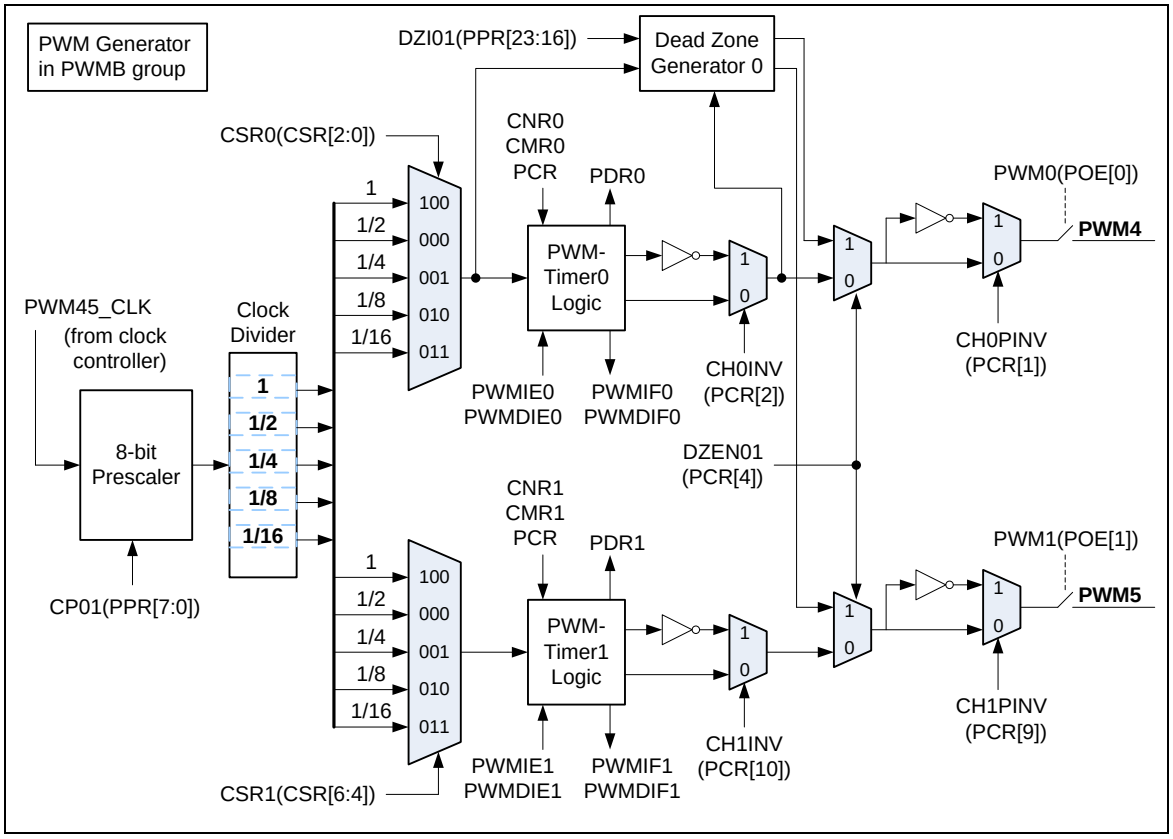


Figure 5-41 PWM Generator 4 Architecture Diagram

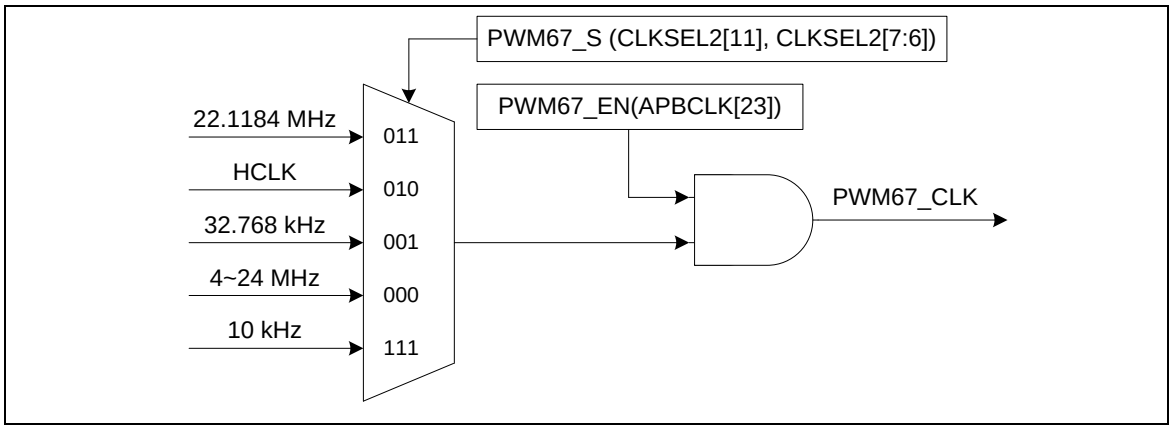


Figure 5-42 PWM Generator 6 Clock Source Control

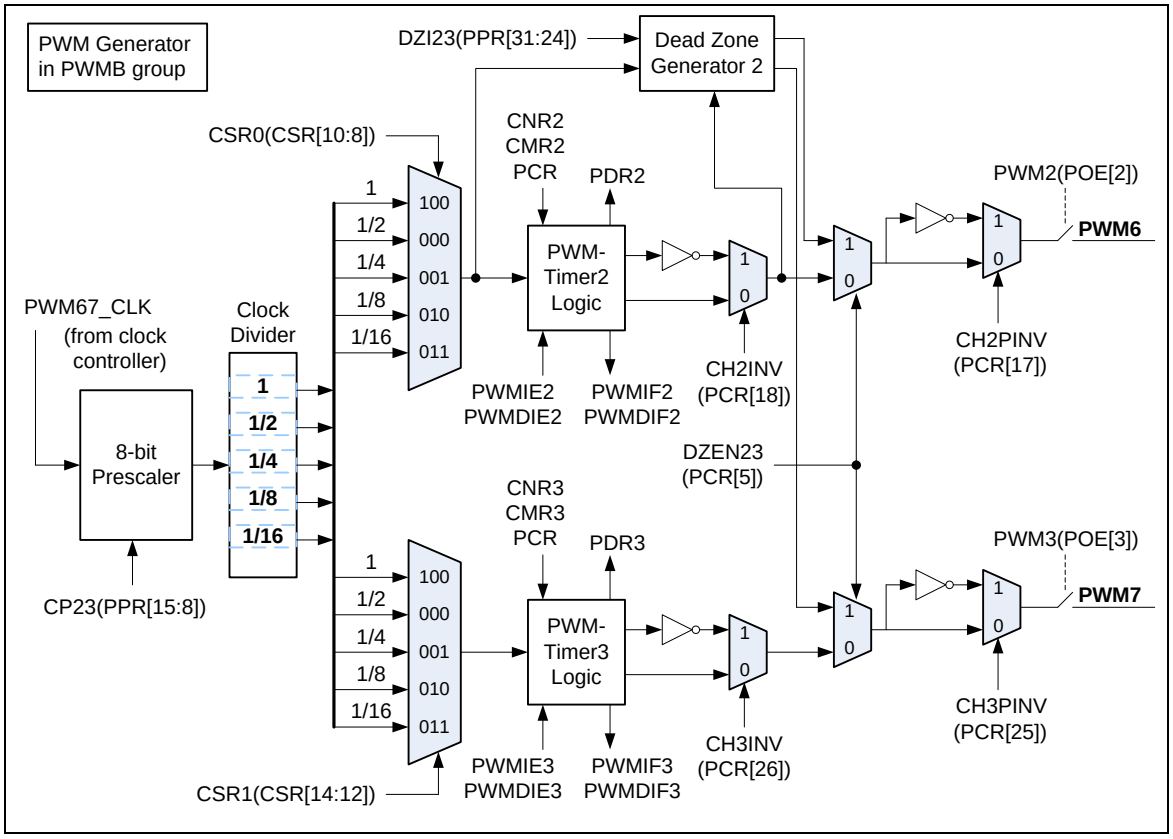


Figure 5-43 PWM Generator 6 Architecture Diagram

5.7.4 Functional Description

5.7.4.1 *PWM-Timer Operation*

The PWM controller supports 2 operation types: Edge-aligned and Center-aligned type.

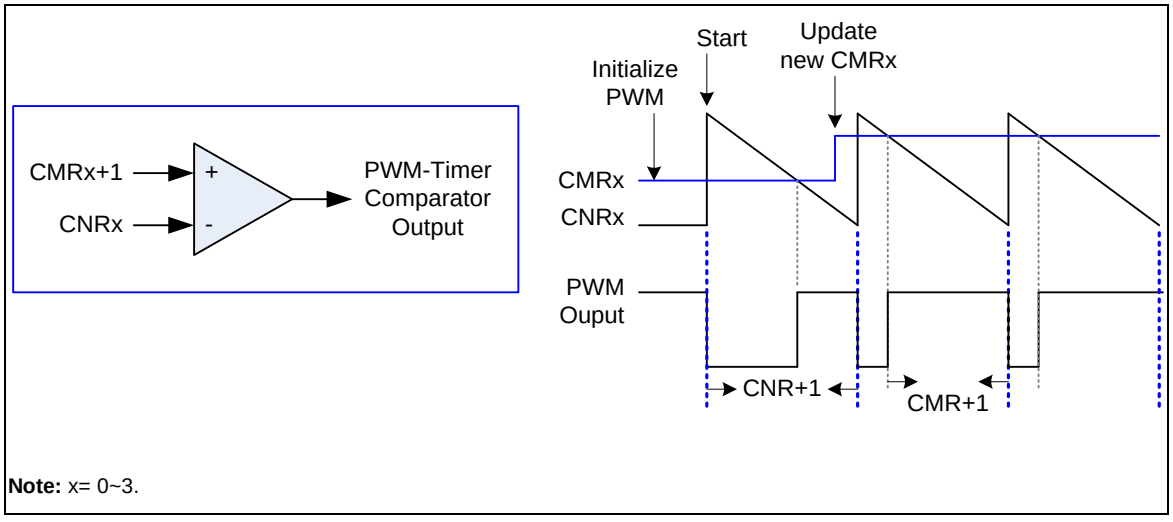
5.7.4.1.1 Edge-aligned PWM (down-counter)

In Edge-aligned PWM Output mode, the 16 bits PWM counter will starts down-counting from CNRn to match with the value of the duty cycle CMRn (old), when this happen it will toggle the PWMn generator output to low. The counter will continue down-counting to 0, at this moment, it toggles the PWMn generator output to high and CMRn(new) and CNRn(new) are updated with CHnMODE=1 and request the PWM interrupt if PWM interrupt is enabled(PIER.n=1).

The PWM period and duty control are configured by PWM down-counter register (CNR) and PWM comparator register (CMR). The PWM-timer timing operation is shown in Figure 5-45. The pulse width modulation follows the formula below and the legend of PWM-Timer Comparator is shown as Figure 5-44. Note that the corresponding GPIO pins must be configured as PWM function (enable POE and disable CAPENR) for the corresponding PWM channel.

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

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



Note: x= 0~3.

Figure 5-44 Legend of Internal Comparator Output of PWM-Timer

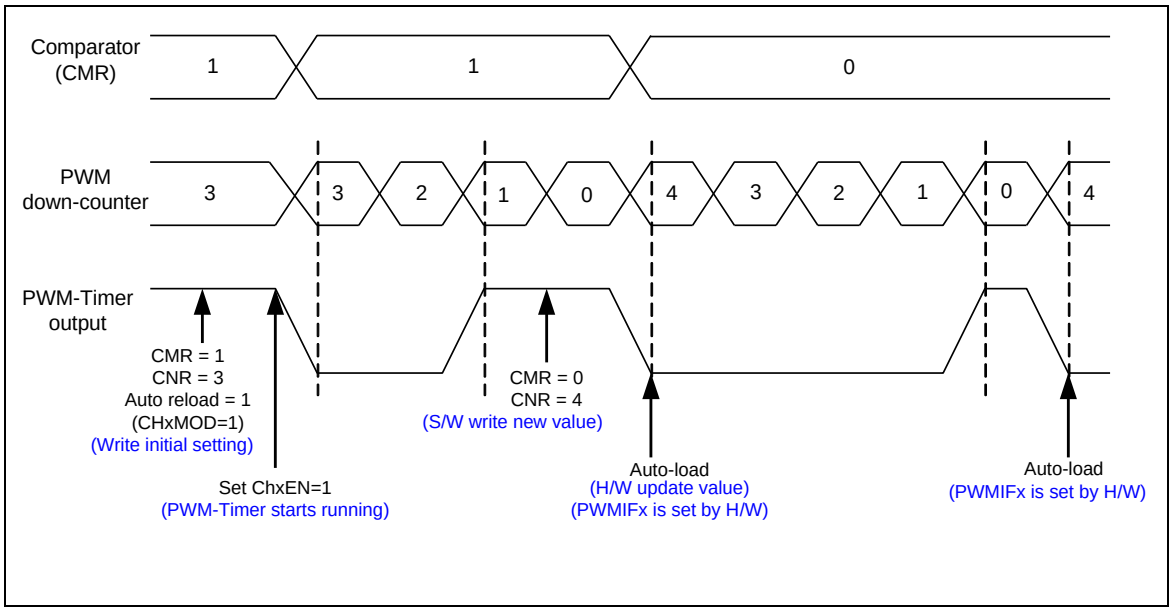


Figure 5-45 PWM-Timer Operation Timing

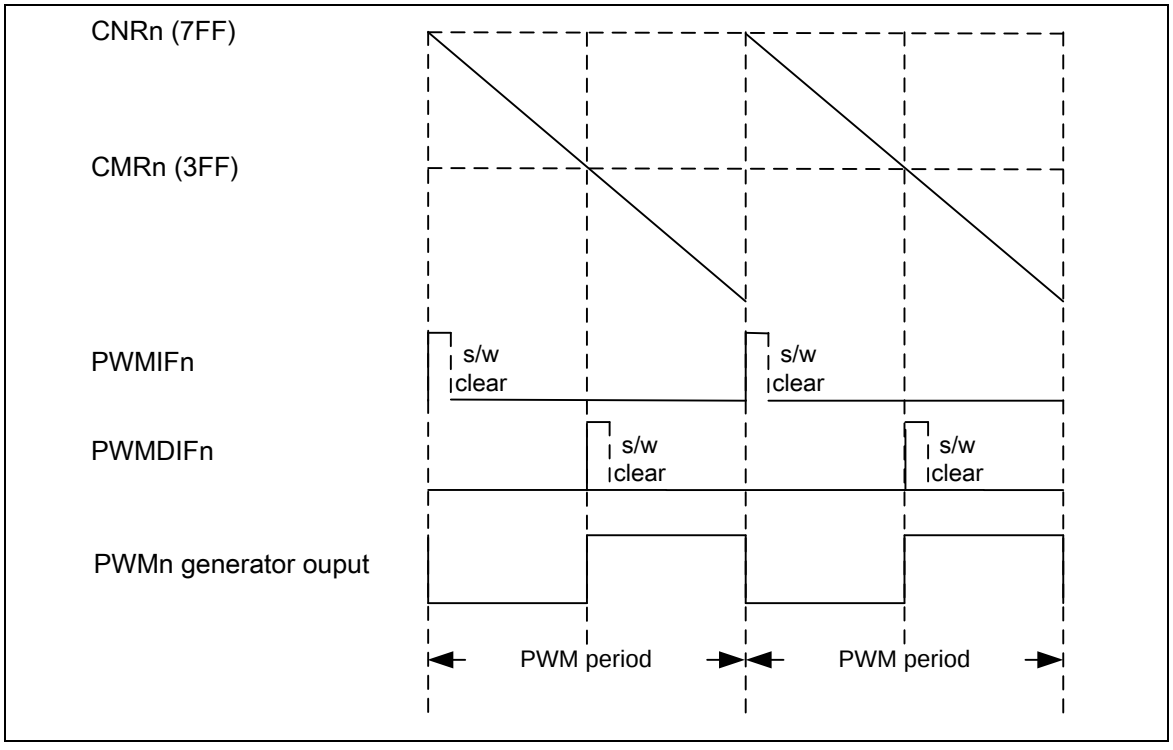


Figure 5-46 PWM Edge-aligned Interrupt Generate Timing Waveform

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

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

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

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

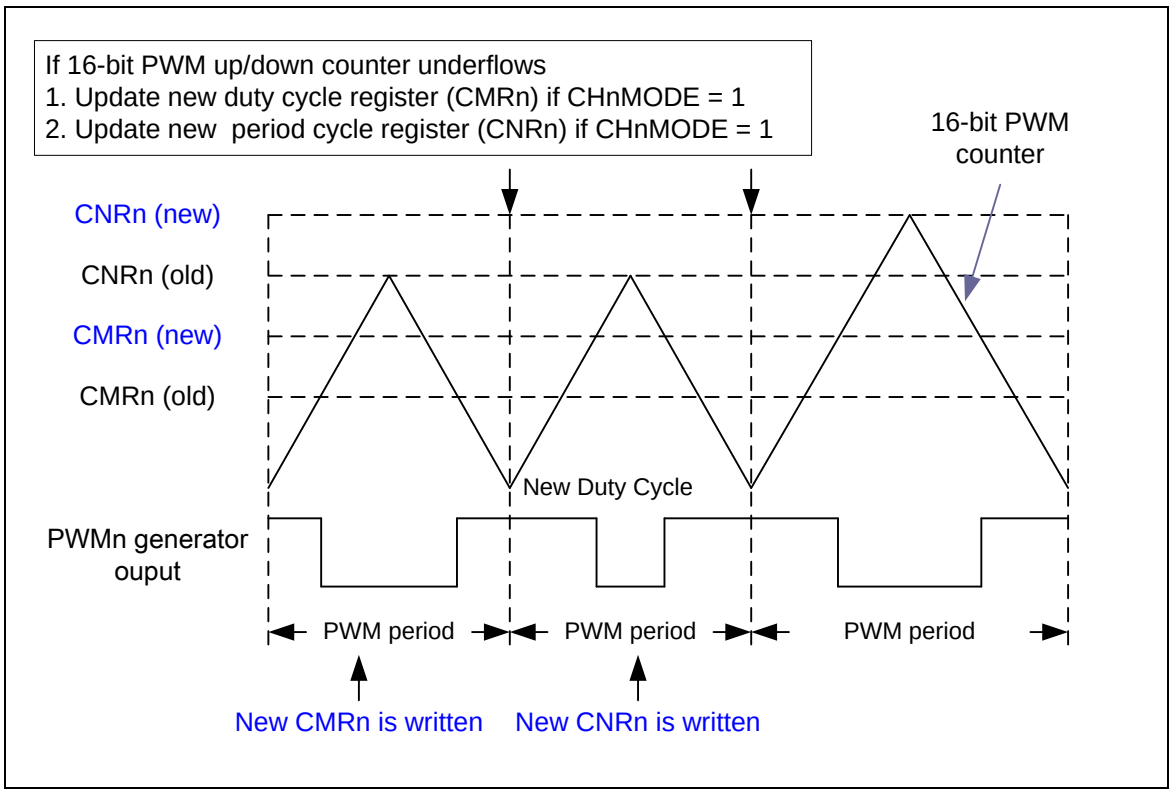


Figure 5-47 Center-aligned Type Output Waveform

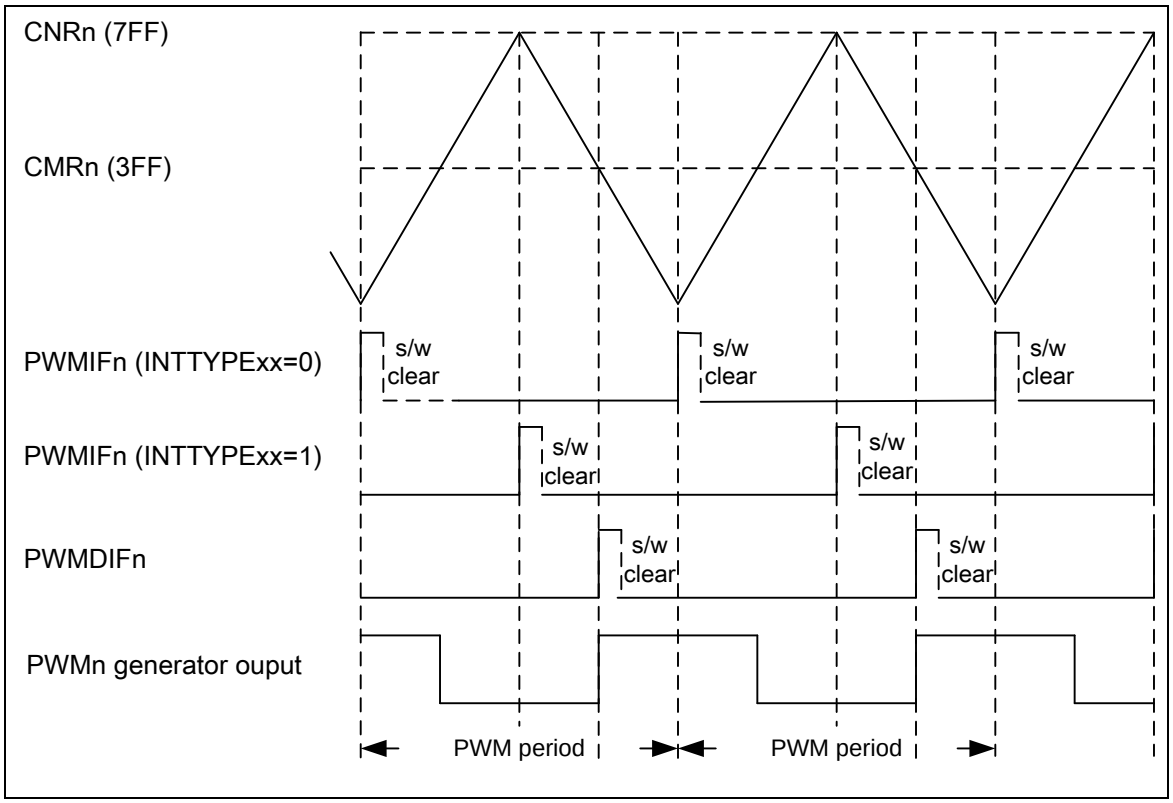


Figure 5-48 PWM Center-aligned Interrupt Generate Timing Waveform

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

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

PWM0 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 PWM0 operating mode before set CH0EN bit to 1 to enable PWM0 counter start running because the content of CNR0 and CMR0 will be cleared to 0 to reset the PWM0 period and duty setting when PWM0 operating mode is changed. As PWM0 operate in One-shot mode, CMR0 and CNR0 should be written first and then set CH0EN bit to 1 to enable PWM0 counter start running. After PWM0 counter down count from CNR0 value to 0, CNR0 and CMR0 will be cleared to 0 by hardware and PWM counter will be held. Software need to write new CMR0 and CNR0 value to set next one-shot period and duty. When re-start next one-shot operation, the CMR0 should be written first because PWM0 counter will auto re-start counting when CNR0 is written a non-zero value. As PWM0 operates at auto-reload mode, CMR0 and CNR0 should be written first and then set CH0EN bit to 1 to enable PWM0 counter start running. The value of CNR0 will reload to PWM0 counter when it down count reaches 0. If CNR0 is set to 0, PWM0 counter will be held. PWM1~PWM7 performs the same function as PWM0.

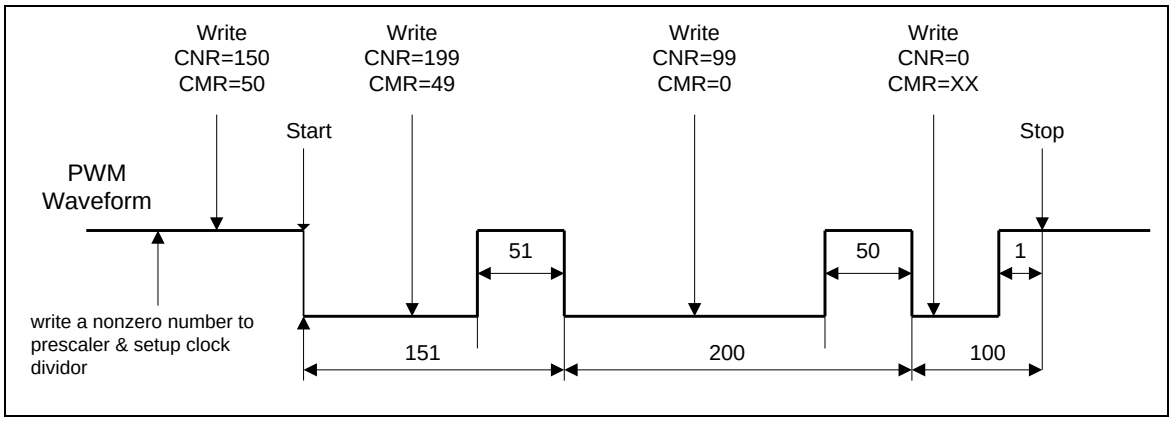


Figure 5-49 PWM Double Buffering Illustration

5.7.4.3 *Modulate Duty Ratio*

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

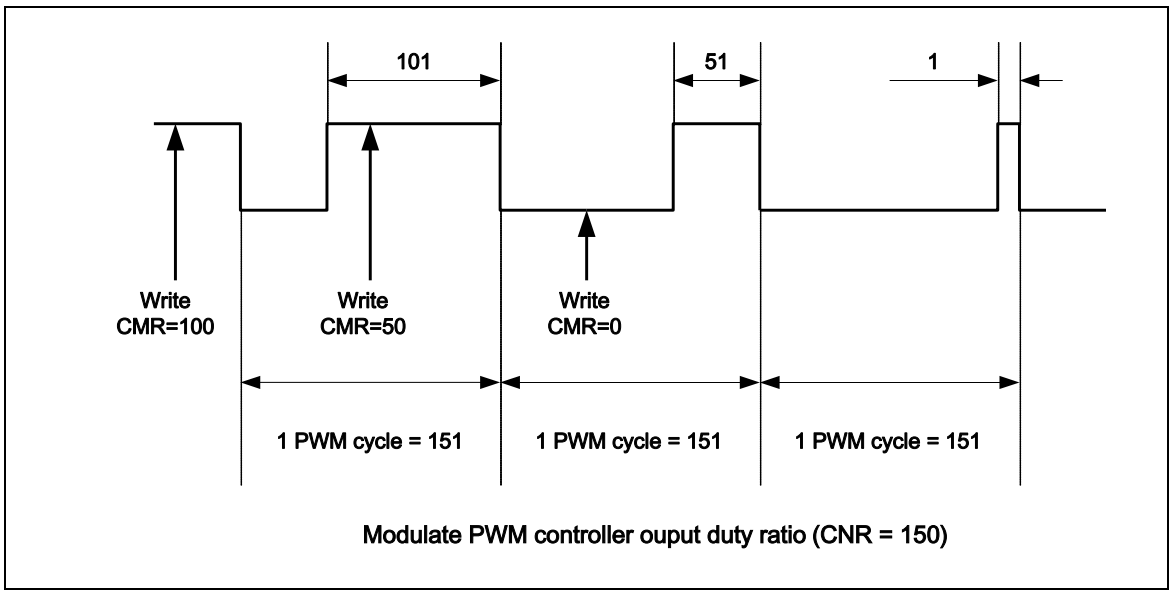


Figure 5-50 PWM Controller Output Duty Ratio

5.7.4.4 *Dead-Zone Generator*

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

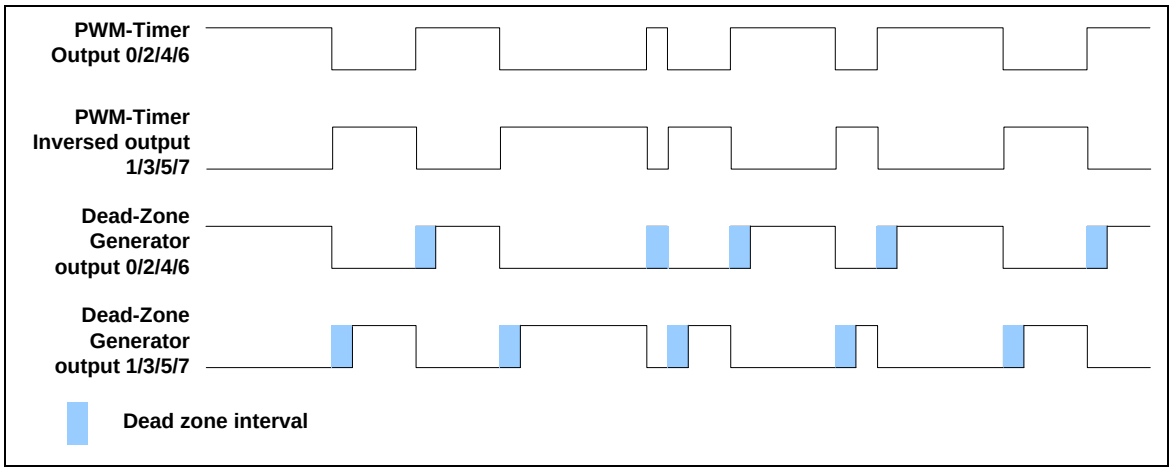


Figure 5-51 Paired-PWM Output with Dead-zone Generation Operation

5.7.4.5 *PWM Center-aligned Trigger ADC Function*

PWM can trigger ADC to start conversion when PWM counter up count to CNR in Center-aligned type by setting PWMnTEN to “1”.

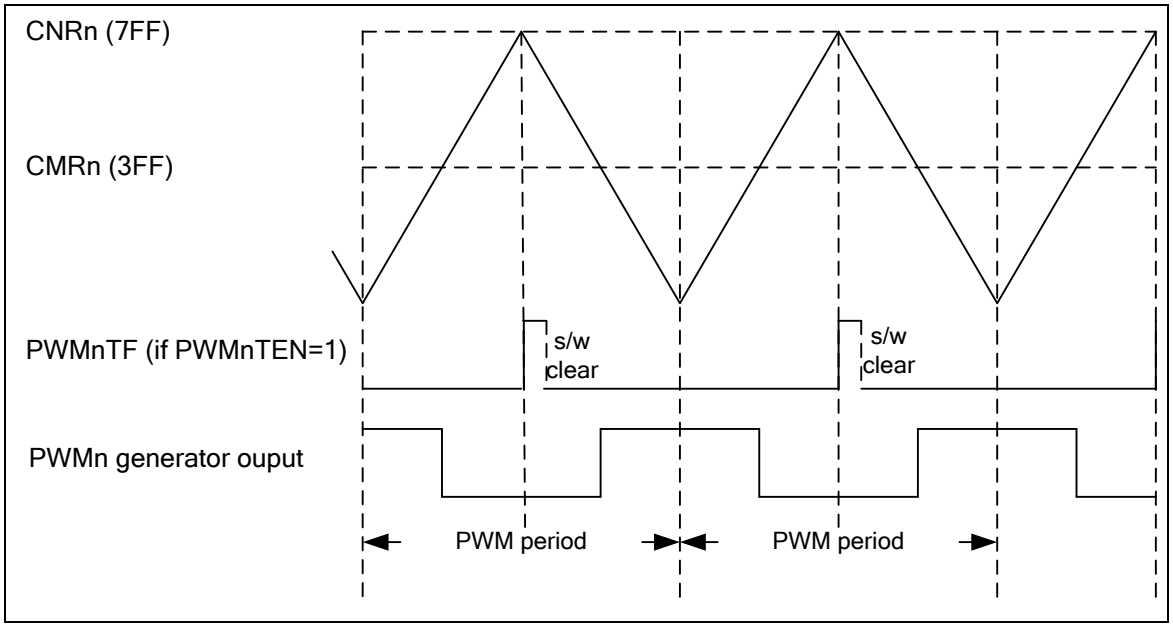


Figure 5-52 PWM trigger ADC to conversion in Center-aligned type Timing Waveform

5.7.4.6 *Capture Operation*

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

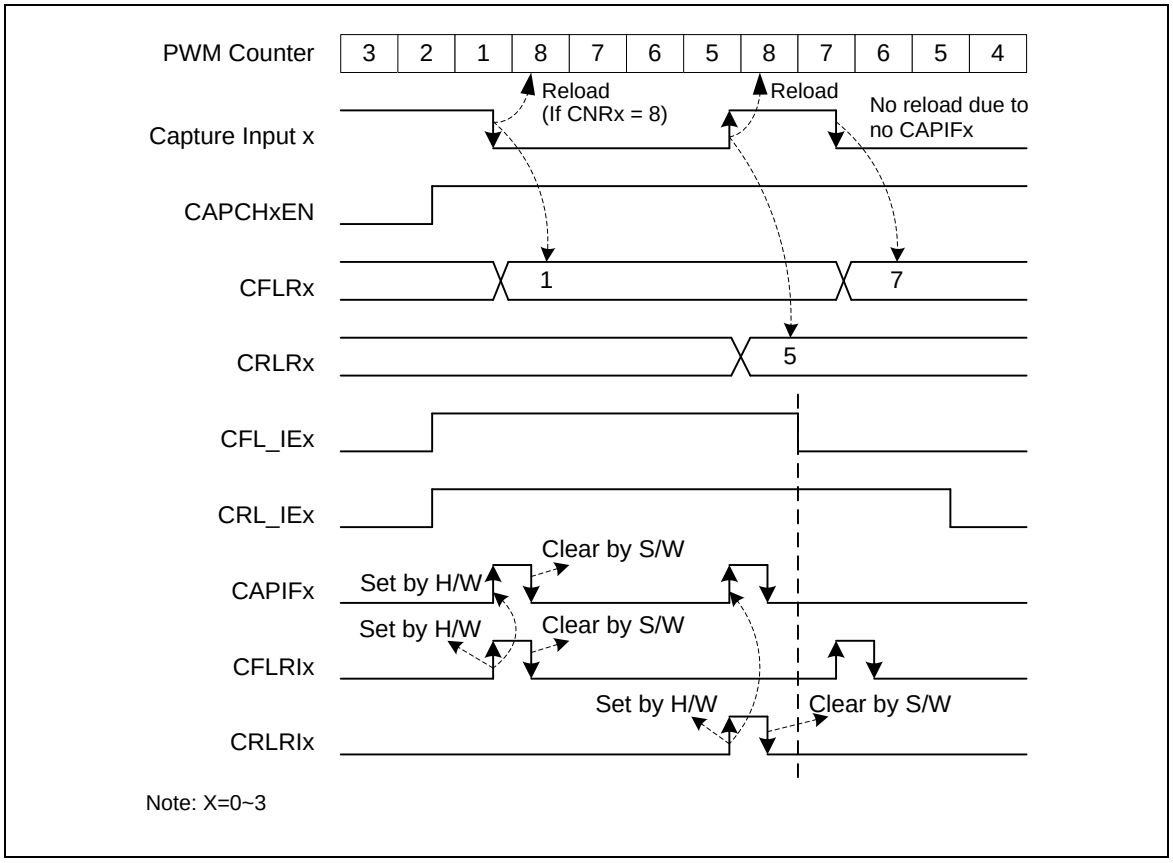


Figure 5-53 Capture Operation Timing

In this case, the CNR is 8:

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

5.7.4.7 *PWM-Timer Interrupt Architecture*

There are eight PWM interrupts, PWM0_INT~PWM7_INT, which are divided into PWMA_INT and PWMB_INT for Advanced Interrupt Controller (AIC). PWM 0 and Capture 0 share one interrupt, PWM1 and Capture 1 share the same interrupt and so on. Therefore, PWM function and Capture function in the same channel cannot be used at the same time. Figure 5-54 and Figure 5-48 demonstrates the architecture of PWM-Timer interrupts.

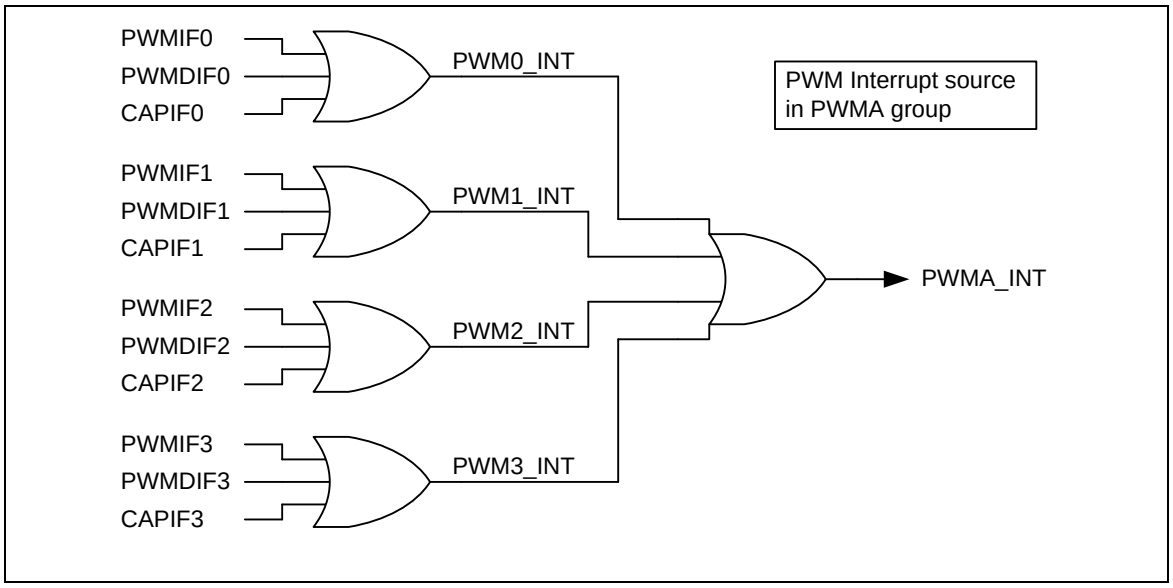


Figure 5-54 PWM Group A PWM-Timer Interrupt Architecture Diagram

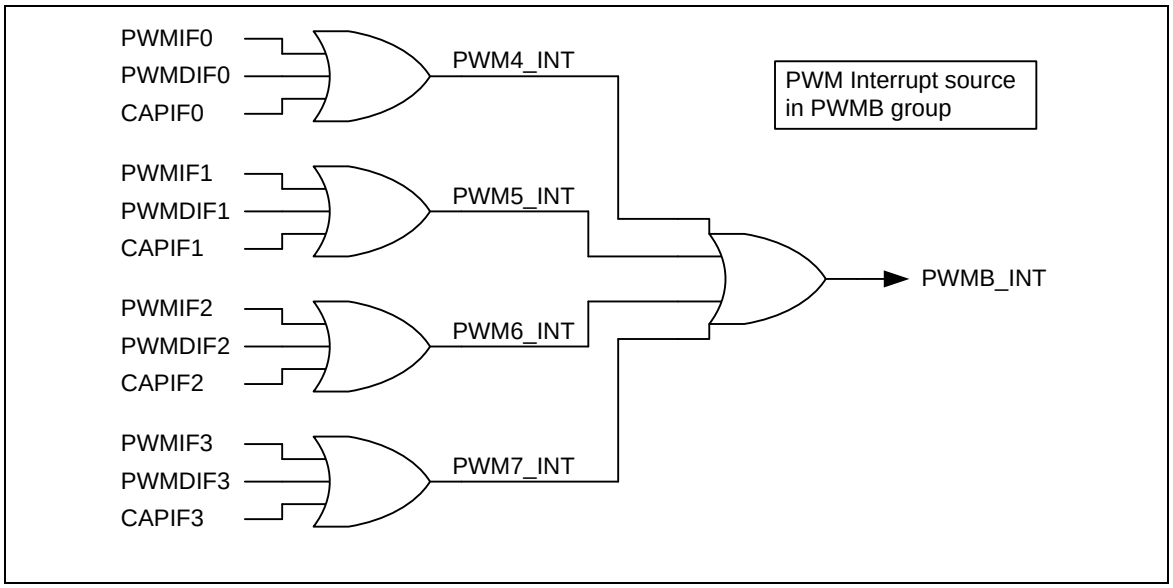


Figure 5-55 PWM Group B PWM-Timer Interrupt Architecture Diagram

5.7.4.8 *PWM-Timer Start Procedure*

The following procedure is recommended for starting a PWM drive.

1. Setup clock source divider select register (CSR)
2. Wait until SYNCBUSYn be set to 0 by hardware (if PWM clock source is not from HCLK)
3. Setup prescaler (PPR)
4. Wait until SYNCBUSYn be set to 0 by hardware (if PWM clock source is not from HCLK)
5. Setup inverter on/off, Dead-zone generator on/off, Auto-reload/One-shot mode and Stop PWM-timer (PCR)
6. Wait until SYNCBUSYn be set to 0 by hardware (if PWM clock source is not from HCLK)
7. Setup comparator register (CMR) for setting PWM duty.
8. Wait until SYNCBUSYn be set to 0 by hardware (if PWM clock source is not from HCLK)
9. Setup PWM down-counter register (CNR) for setting PWM period.
10. Setup interrupt enable register (PIER) (optional)
11. Setup corresponding GPIO pins as PWM function (enable POE and disable CAPENR) for the corresponding PWM channel.
12. Enable PWM timer start running (Set CHxEN = 1 in PCR)

5.7.4.9 *Modify PWM counter register (CNR), comparator register (CMR), Clock prescaler(CP01/CP23) and PWM operation mode(CHnMOD in PCR register bit3) Procedure*

The following procedure is recommended for modifying CNR/CMR/Clock prescaler/PWM operation mode.

1. Wait until SYNCBUSYn be set to 0 by hardware (if PWM clock source is not from HCLK)
2. Modify CMRn/CNRn/CHnMOD/CP01/CP23

5.7.4.10 *PWM-Timer Re-Start Procedure in Single-shot mode*

After PWM waveform is generated once in PWM One-shot mode, PWM-Timer will be stopped automatically and both of CNR and CMR will be cleared by hardware. Software must fill CMR and CNR value again to re-start another PWM one-shot waveform. The following procedure is recommended for re-starting PWM one-shot waveform.

1. Wait until SYNCBUSYn be set to 0 by hardware (if PWM clock source is not from HCLK)
2. Setup comparator register (CMR) for setting PWM duty.
3. Wait until SYNCBUSYn be set to 0 by hardware (if PWM clock source is not from HCLK)
4. Setup PWM down-counter register (CNR) for setting PWM period. After setup CNR, PWM wave will be generated.

5.7.4.11 *PWM-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 PWM-Timer (CHxEN in PCR). ***(Recommended)***

Method 2:

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

Method 3:

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

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

5.7.4.12 *Capture Start Procedure*

1. Setup clock source divider select register (CSR)
2. Wait until SYNCBUSYn be set to 0 by hardware (if PWM clock source is not from HCLK)
3. Setup prescaler (PPR)
4. Setup channel enabled, rising/falling interrupt enable and input signal inverter on/off (CCR0, CCR2)
5. Wait until SYNCBUSYn be set to 0 by hardware (if PWM clock source is not from HCLK)
6. Setup Auto-reload mode, Edge-aligned type and Stop PWM-timer (PCR)
7. Wait until SYNCBUSYn be set to 0 by hardware (if PWM clock source is not from HCLK)
8. Setup PWM down-counter (CNR)
9. Enable PWM timer start running (Set CHxEN = 1 in PCR)
10. Setup corresponding GPIO pins as capture function (disable POE and enable CAPENR) for the corresponding PWM channel.

5.8 Real Time Clock (RTC)

5.8.1 Overview

The Real Time Clock (RTC) controller provides user with the real time and calendar message. The clock source of RTC controller is from an external 32.768 kHz low speed crystal which connected at pins X32_IN and X32_OUT (refer to pin Description) or from an external 32.768 kHz low speed oscillator output fed at pin X32_IN. The RTC controller provides the real time message (hour, minute, second) in TLR (RTC Time Loading Register) as well as calendar message (year, month, day) in CLR (RTC Calendar Loading Register). It also offers RTC alarm function that user can preset the alarm time in TAR (RTC Time Alarm Register) and alarm calendar in CAR (RTC Calendar Alarm Register). The data format of RTC time and calendar message are all expressed in BCD format.

The RTC controller supports periodic RTC Time Tick and Alarm Match interrupts. The periodic RTC Time Tick interrupt has 8 period interval options 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second which are selected by TTR (TTR[2:0] Time Tick Register). When real time and calendar message in TLR and CLR are equal to alarm time and calendar settings in TAR and CAR, the AIF (RIIR [0] RTC Alarm Interrupt Flag) is set to 1 and the RTC alarm interrupt signal is generated if the AIER (RIER [0] Alarm Interrupt Enable) is enabled.

Both RTC Time Tick and Alarm Match interrupt signal can cause chip to wake-up from Idle or Power-down mode if the correlate interrupt enable bit (AIER or TIER) is set to 1 before chip enters Idle or Power-down mode.

5.8.2 Features

- Supports real time counter in TLR (hour, minute, second) and calendar counter in CLR (year, month, day) for RTC time and calendar check
- Supports alarm time (hour, minute, second) and calendar (year, month, day) settings in TAR and CAR
- Selectable 12-hour or 24-hour time scale in TSSR register
- Supports Leap Year indication in LIR register
- Supports Day of the Week counter in DWR register
- Frequency of RTC clock source compensate by FCR register
- All time and calendar message expressed in BCD format
- Supports periodic RTC Time Tick interrupt with 8 period interval options 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second
- Supports RTC Time Tick and Alarm Match interrupt
- Supports chip wake-up from Idle or Power-down mode while a RTC interrupt signal is generated

5.8.3 Block Diagram

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

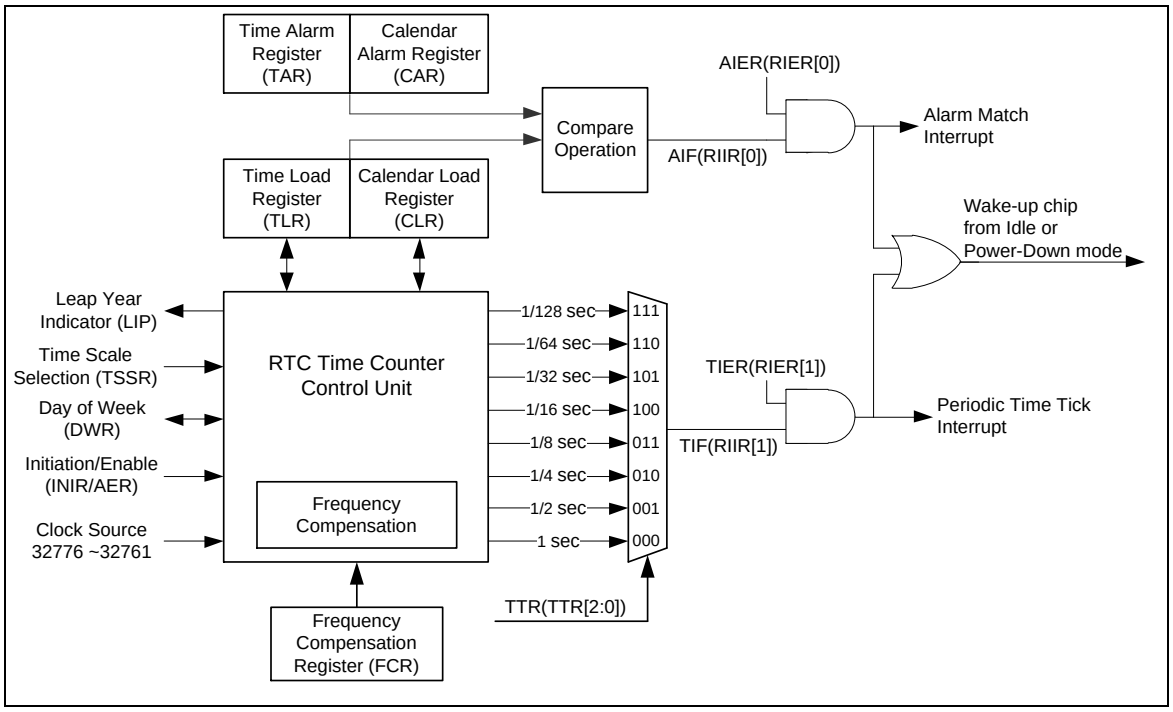


Figure 5-56 RTC Block Diagram

5.8.4 Functional Description

5.8.4.1 *RTC Initiation*

When a RTC block is powered on, RTC is at reset state. User has to write a number 0xa5eb1357 to INIR (INIR [31:0] RTC Initiation) register to make RTC leaving reset state. Once the INIR is written as 0xa5eb1357, the RTC will be in normal active state permanently. User can read Active (INIR[0] RTC Active Status) bit status to check the RTC is at normal active state or reset state.

5.8.4.2 *Access to RTC register*

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

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

5.8.4.3 *RTC Read/Write Enable*

AER [AER [15:0] RTC Register Access Enable Password] is served as read/write access of RTC registers to unlock RCT registers read/write protect function. If AER [15:0] is written to 0xA965, user can read ENF (AER [16] RTC Register Access Enable Flag) bit status to check the RTC registers are read/write access or locked. Once ENF bit enabled, RTC Access Enable function will keep effect at least 1024 RTC clocks (about 30ms) and ENF bit will be cleared automatically after 1024 RTC clocks.

5.8.4.4 *Frequency Compensation*

The RTC source clock may not precise to exactly 32768 Hz and the FCR register (Frequency Compensation Register) allows software to make digital compensation to the RTC source clock only if the frequency of RTC source clock is in the range from 32761 Hz to 32776 Hz.

Following are the compensation examples for the real RTC source clock is higher or lower than 32768 Hz.

Example 1: (RTC source clock > 32768 Hz)

RTC source clock measured: 32773.65 Hz (> 32768 Hz)

Integer part: 32773 => 0x8005

INTEGER (FCR [11:8] Integer Part) = 0x05 – 0x01 + 0x08 = 0x0c

Fraction part: 0.65

FRACTION (FCR [5:0] Fraction Part) = 0.65 x 60 = 39 = 0x27

FCR register should be as 0xC27

Example 2: (RTC source clock ≦ 32768 Hz)

RTC source clock measured: 32765.27 Hz (≦ 32768 Hz)

Integer part: 32765 => 0x7FFD

INTEGER (FCR [11:8] Integer Part) = 0x0D – 0x01 – 0x08 = 0x04

Fraction part: 0.27

FRACTION (FCR [5:0] Fraction Part) = 0.27 x 60 = 16.2 = 0x10

FCR register should be as 0x410

5.8.4.5 *Time and Calendar counter*

TLR and CLR are used to load the real time and calendar. TAR and CAR are used for setup alarm time and calendar.

5.8.4.6 *12/24 hour Time Scale Selection*

The 12/24 hour time scale selection depends on TSSR bit (TSSR [0] 24-Hour / 12-Hour Time Scale Selection).

5.8.4.7 *Day of the Week counter*

The RTC controller provides day of week in DWR (DWR [2:0] Day of the Week Register). The value is defined from 0 to 6 to represent Sunday to Saturday respectively.

5.8.4.8 *Periodic Time Tick Interrupt*

The Periodic Time Tick interrupt has 8 period interval options 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second that are selected by TTR (TTR[2:0] Time Tick Register). When Periodic Time Tick interrupt is enabled by setting TIER (RIER [1] Time Tick Interrupt Enable) to 1, the Periodic Time Tick interrupt is requested periodically in the period selected by TTR[2:0] settings.

5.8.4.9 *Alarm Interrupt*

When the real time and calendar message in TLR and CLR are equal to alarm time and calendar settings in TAR and CAR, the AIF (RIIR [0] RTC Alarm Interrupt Flag) is set to 1 and the RTC alarm interrupt signal is generated if the AIER (RIER[0] Alarm Interrupt Enable) is enabled.

5.8.4.10 *Application Note:*

1. All data in TAR, CAR, TLR and CLR registers are all expressed in BCD format.
2. Programmer has to make sure that the loaded values are reasonable. For example, Load CLR as 201a (year), 13 (month), 00 (day), or CLR does not match with DWR, etc.
3. Registers value after powered on or reset:

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

4. In CLR and CAR, only 2 BCD digits are used to express “year”. The 2 BCD digits of xy means 20xy, rather than 19xy or 21xy.

5.9 Serial Peripheral Interface (SPI)

5.9.1 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™ NUC200 series contains up to four sets of SPI controllers performing a serial-to-parallel conversion on data received from a peripheral device, and a parallel-to-serial conversion on data transmitted to a peripheral device. Each set of SPI controller can be configured as a master or a slave device.

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

5.9.2 Features

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

5.9.3 Block Diagram

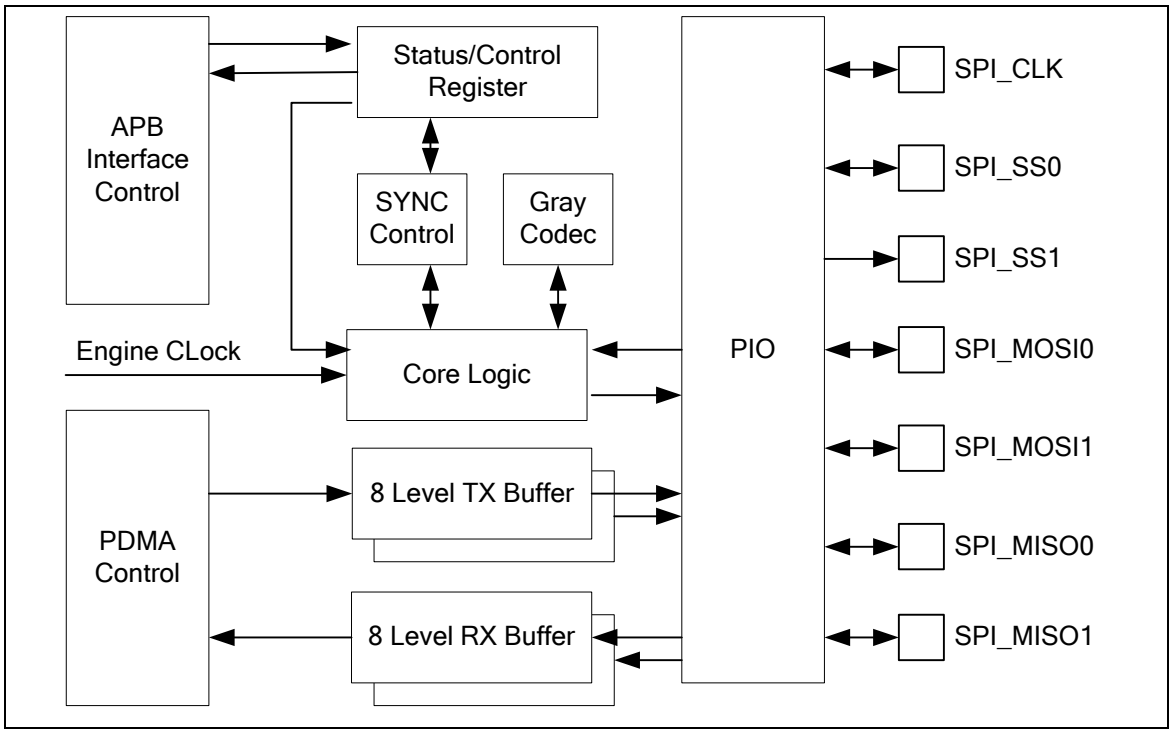


Figure 5-57 SPI Block Diagram

5.9.4 Functional Description

SPI Engine Clock and SPI Serial Clock

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

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

Master/Slave Mode

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

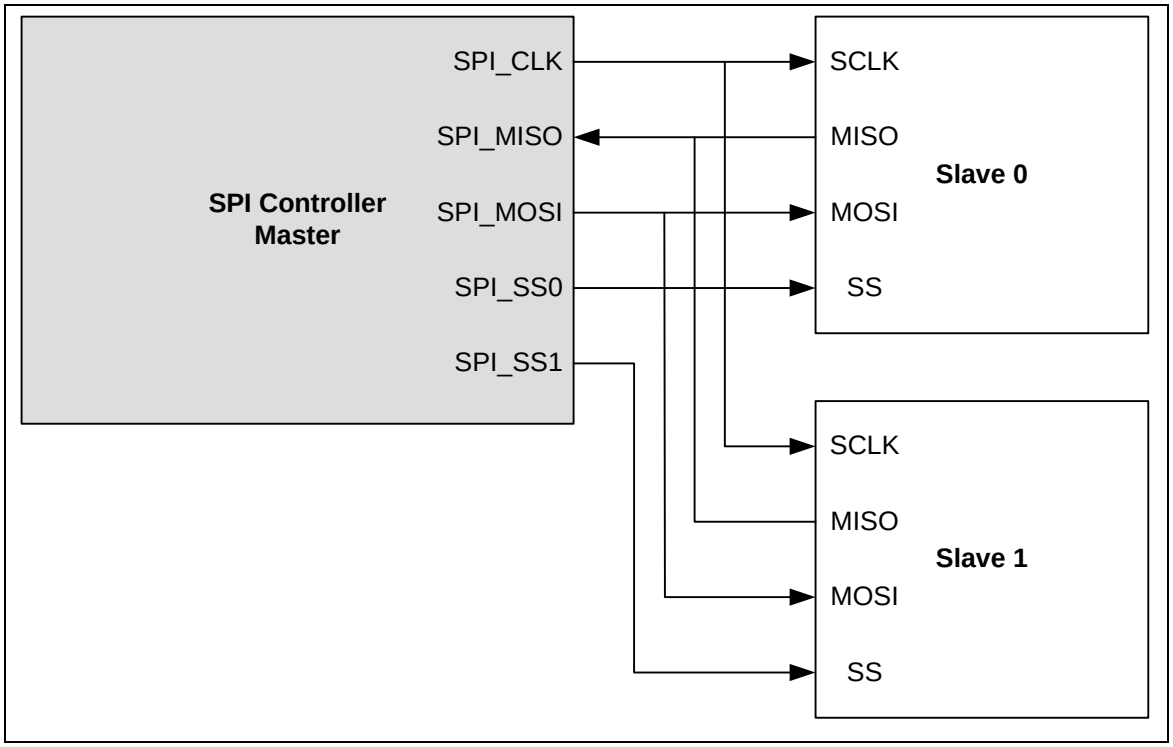


Figure 5-58 SPI Master Mode Application Block Diagram

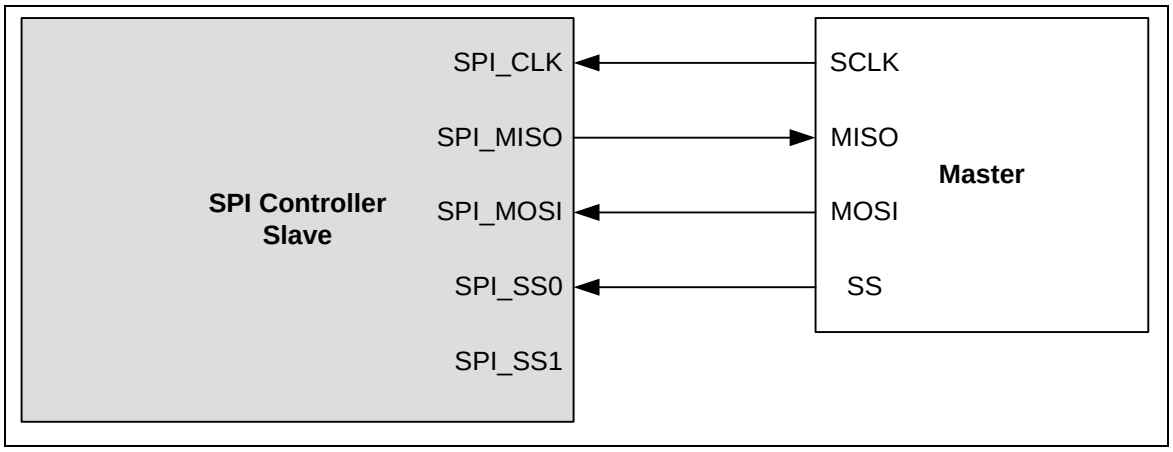


Figure 5-59 SPI Slave Mode Application Block Diagram

Slave Selection

In Master mode, the SPI controller can drive up to two off-chip slave devices through the slave select output pins SPI_SS0 and SPI_SS1. In Slave mode, the off-chip master device drives the slave select signal from the SPI_SS0 input port to this SPI controller. In Master/Slave mode, the active state of slave select signal can be programmed to low or high active in SS_LVL bit (SPI_SSR[2]), and the SS_LTRIG bit (SPI_SSR[4]) defines the slave select signal SPI_SS0/1 is level-triggered or edge-triggered. The selection of trigger conditions depends on what type of peripheral slave/master device is connected.

In Slave mode, if the SS_LTRIG bit is configured as level trigger, the LTRIG_FLAG bit (SPI_SSR[5]) is used to indicate if the received bits among one transaction meets the requirement defined in TX_BIT_LEN.

Level-trigger/Edge-trigger

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

Automatic Slave Selection

In Master mode, if the bit AUTOSS (SPI_SSR[3]) is set, the slave select signals will be generated automatically and output to the SPI_SS0 and SPI_SS1 pins according to whether SSR[0] (SPI_SSR[0]) and SSR[1] (SPI_SSR[1]) are enabled or not. This means that the slave select signals, which are selected in SSR[1:0], will be asserted by the SPI controller when the SPI data transfer is started by setting the GO_BUSY bit (SPI_CNTRL[0]) and will be de-asserted after the data transfer is finished. If the AUTOSS bit is cleared, the slave select output signals will be asserted/de-asserted by manual setting/clearing the related bits of SPI_SSR[1:0]. The active state of the slave select output signals is specified in SS_LVL bit (SPI_SSR[2]).

In Master mode, if the value of SP_CYCLE[3:0] is less than 3 and the AUTOSS is set as 1, the

slave select signal will be kept in active state between two successive transactions.

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

Variable Serial Clock Frequency

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

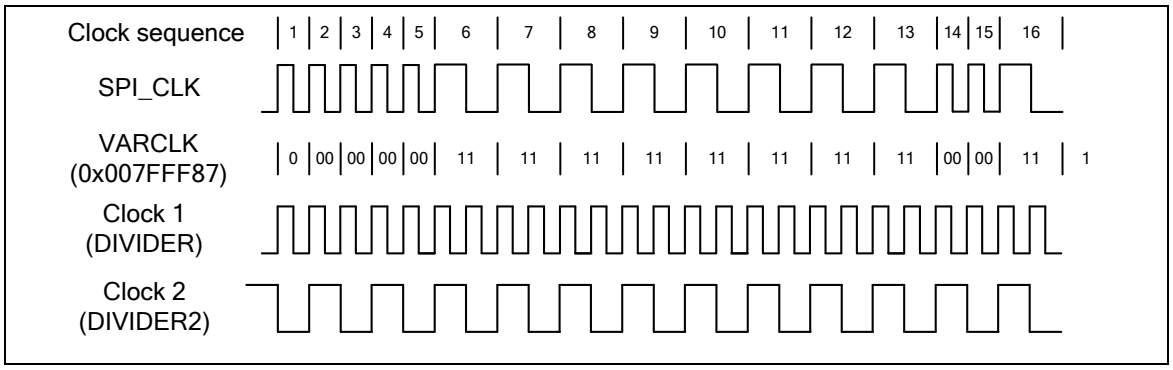


Figure 5-60 Variable Serial Clock Frequency

Clock Polarity

The CLKP bit (SPI_CNTRL[11]) defines the SPI clock idle state. If CLKP = 1, the output SPI clock is idle at high state; if CLKP = 0, it is idle at low state.

Transmit/Receive Bit Length

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

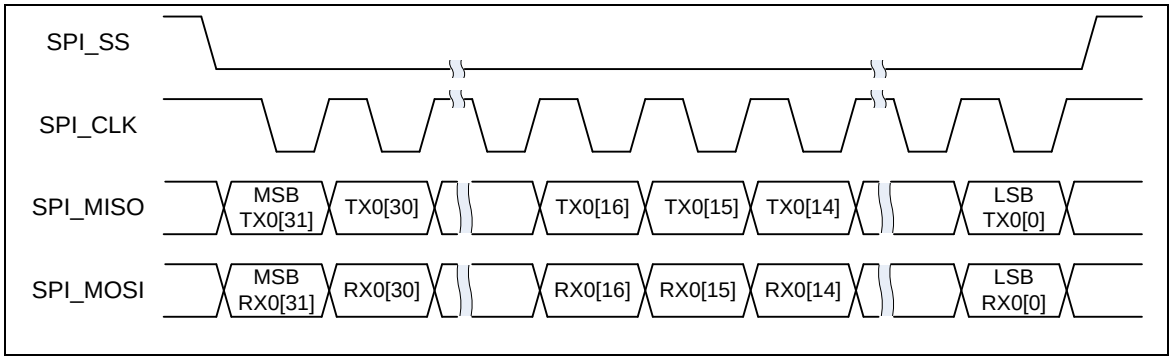


Figure 5-61 32-Bit in One Transaction

LSB/MSB First

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

Transmit Edge

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

Receive Edge

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

Word Suspend

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

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

Byte Reorder

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

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

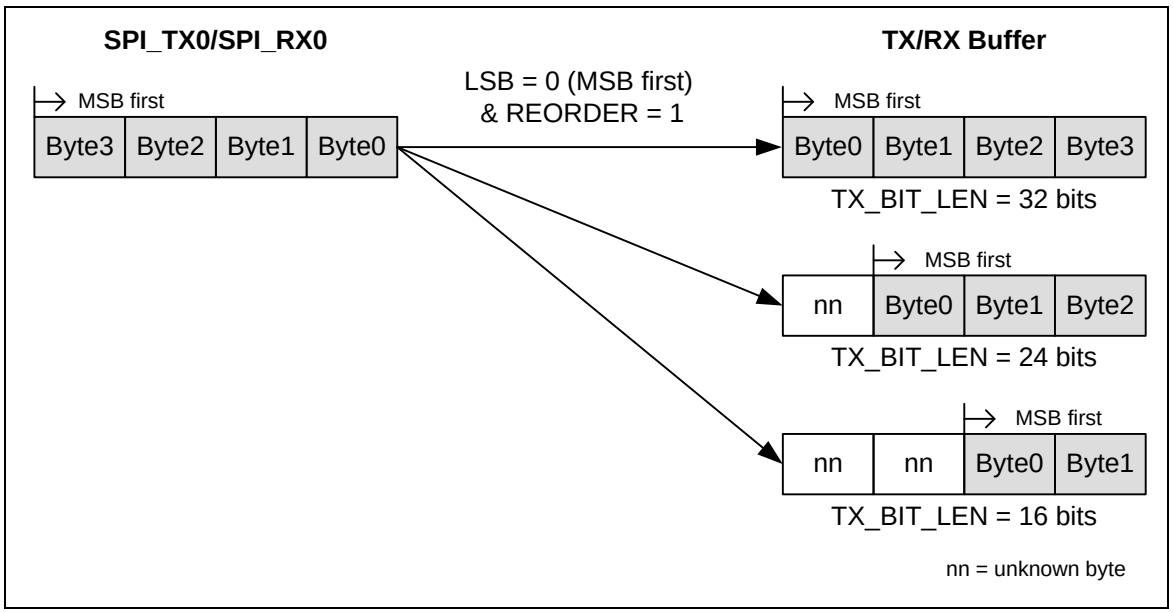


Figure 5-62 Byte Reorder Function

Byte Suspend

In Master mode, if 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.

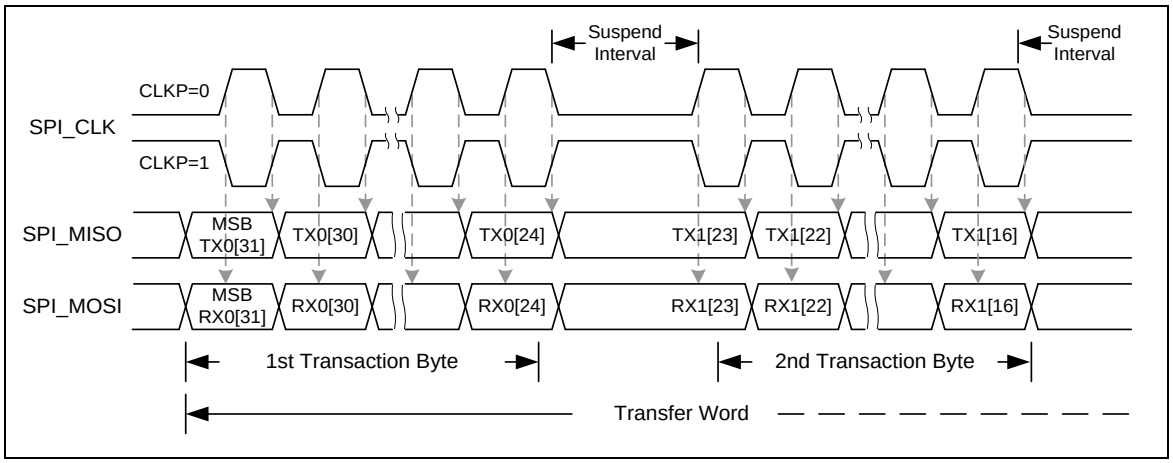


Figure 5-63 Timing Waveform for Byte Suspend

3-Wire Mode

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

2-Bit Mode

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

For example, in Master mode, the data stored in the SPI_TX0 and SPI_TX1 register will be transmitted through the SPI_MOSI0 and SPI_MOSI1 pin respectively. In the meanwhile, the SPI_RX0 and SPI_RX1 will store the data received from SPI_MISO0 pin and SPI_MISO1 pin respectively.

In Slave mode, the data stored in the SPI_TX0 and SPI_TX1 register will be transmitted through the SPI_MISO0 and SPI_MISO1 pin respectively. In the meanwhile, the SPI_RX0 and SPI_RX1 will store the data received from the SPI_MOSI0 and SPI_MOSI1 pin respectively.

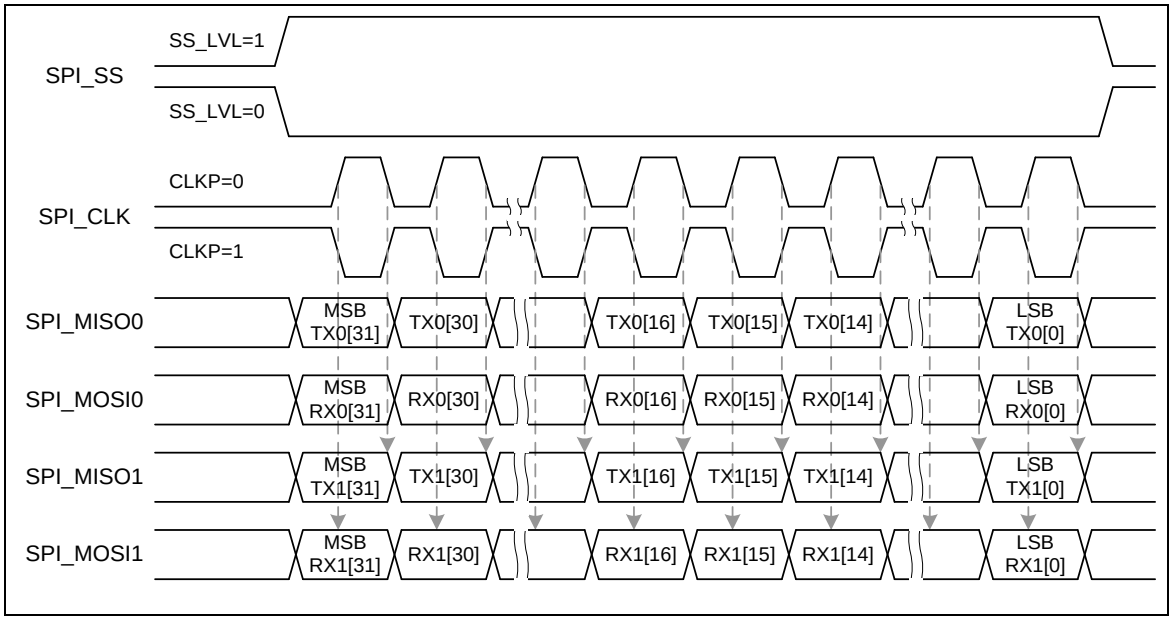


Figure 5-64 2-Bit Mode (Slave Mode)

Dual I/O Mode

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

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

If both the DUAL_IO_EN and DUAL_IO_DIR bits are set as 1, the SPI_MOSI0 is the even bit data output and the SPI_MISO0 will be set as the odd bit data output. If the DUAL_IO_EN is set as 1 and DUAL_IO_DIR is set as 0, both the SPI_MISO0 and SPI_MOSI0 will be set as data input ports.

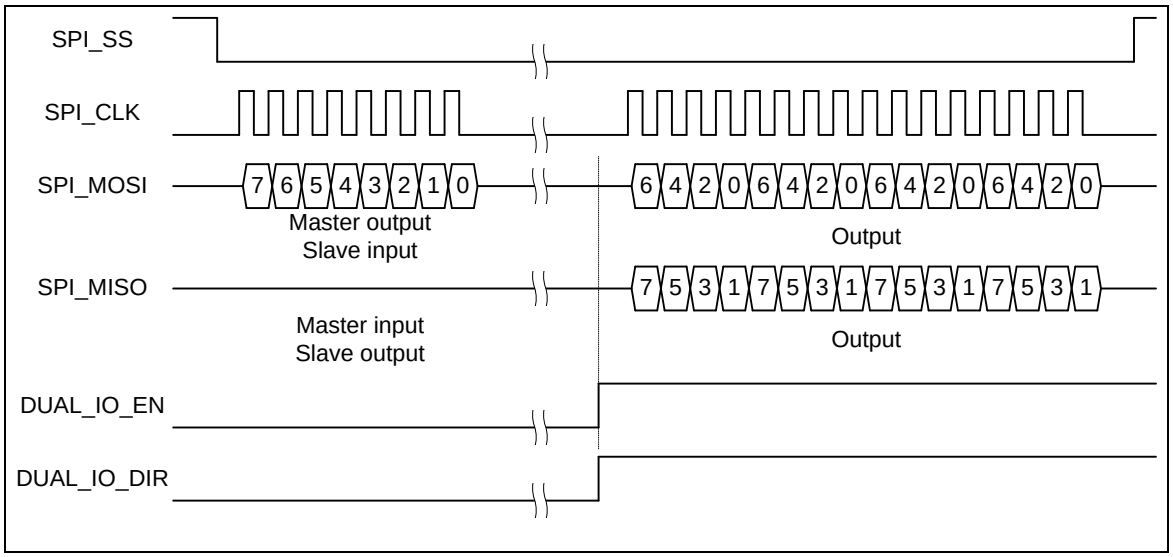


Figure 5-65 Bit Sequence of Dual Output Mode

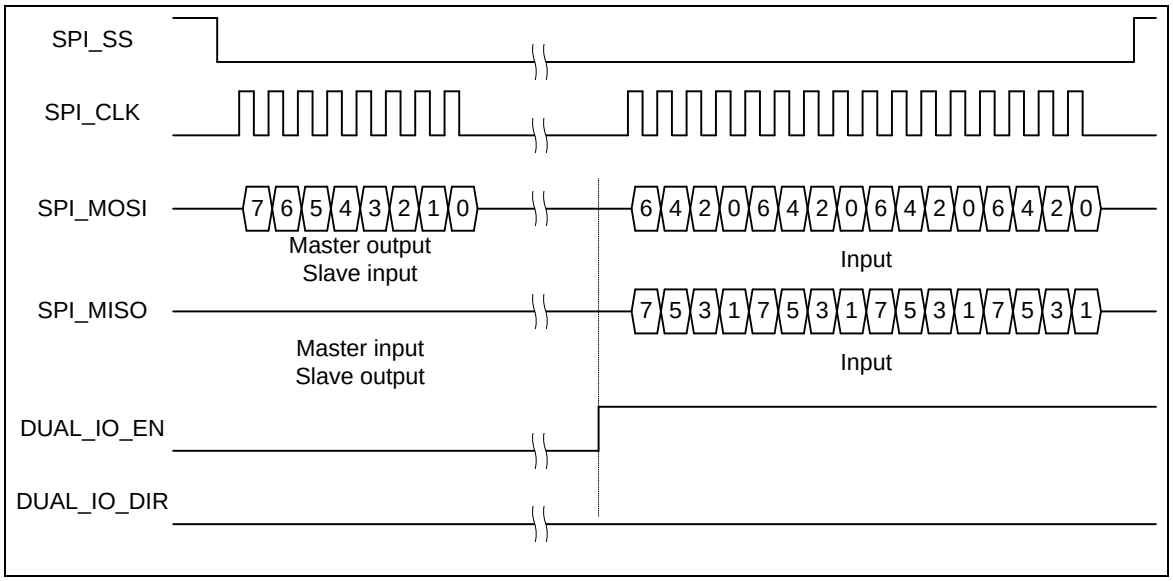


Figure 5-66 Bit Sequence of Dual Input Mode

FIFO Mode

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

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

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

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

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

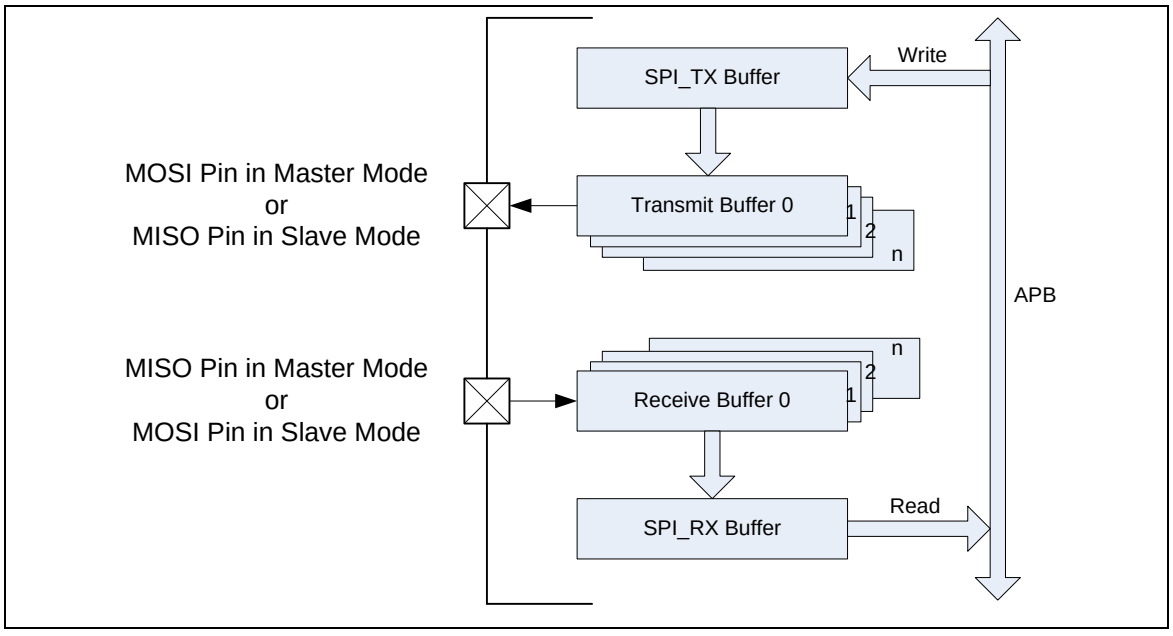


Figure 5-67 FIFO Mode Block Diagram

In Master mode, when the FIFO bit is set to 1 and the first datum is written to the SPI_TX0 register, the TX_EMPTY flag will be cleared to 0. The transmission immediately starts as long as the transmit FIFO buffer is not empty. User can write the next data into SPI_TX0 register immediately. The SPI controller will insert a suspend interval between two successive transactions in FIFO mode and the period of suspend interval is decided by the setting of SP_CYCLE (SPI_CNTRL [15:12]). User can write data into SPI_TX0 register as long as the TX_FULL flag is 0.

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

In Master mode, during receiving operation, the serial data are received from SPI_MISO0/1 pin and stored to receive FIFO buffer. The RX_EMPTY flag will be cleared to 0 while the receive FIFO buffer contains unread data. The received data can be read by software from SPI_RX0 register as long as the RX_EMPTY flag is 0. If the receive FIFO buffer contains 8 unread data, the RX_FULL flag will be set to 1. The SPI controller will stop receiving data until the SPI_RX0 register is read by software.

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

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

In Slave mode, during receiving operation, the serial data is received from SPI_MOSI0/1 pin and stored to SPI_RX0 register. The reception mechanism is similar to Master mode reception operation.

Interrupt

- SPI unit transfer interrupt

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

- SPI slave 3-wire mode start interrupt

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

- Receive FIFO time-out interrupt

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

- Transmit FIFO interrupt

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

- Receive FIFO interrupt

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

5.9.5 Timing Diagram

The active state of slave select signal can be defined by setting the SS_LVL bit (SPI_SSR[2]) and SS_LTRIG bit (SPI_SSR[4]). The SPI clock which is in 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 transmitting/receiving 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.

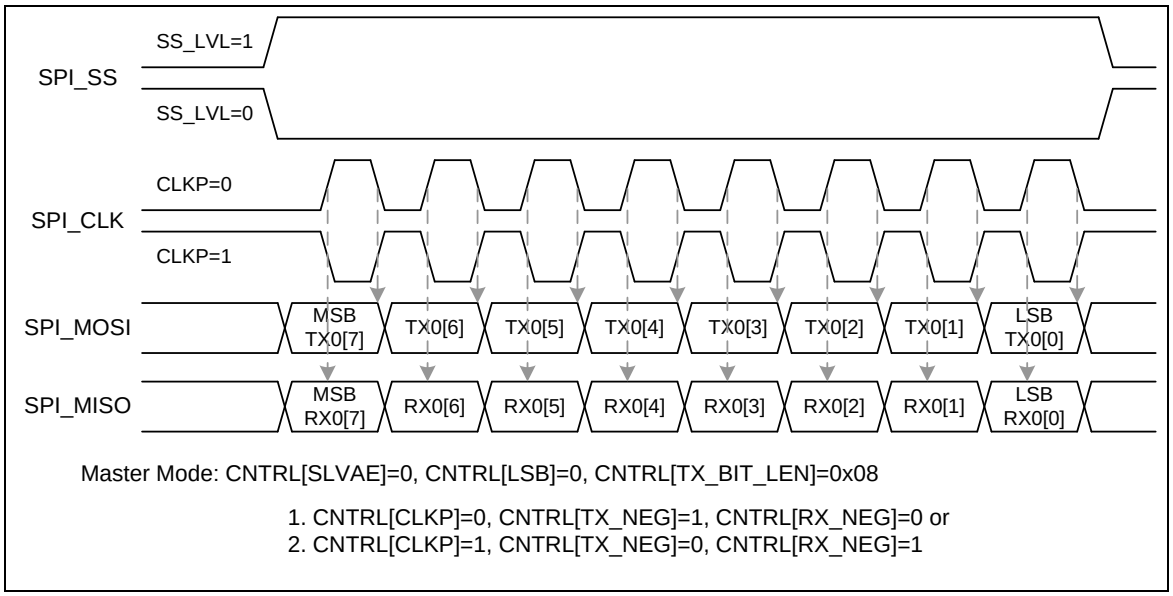


Figure 5-68 SPI Timing in Master Mode

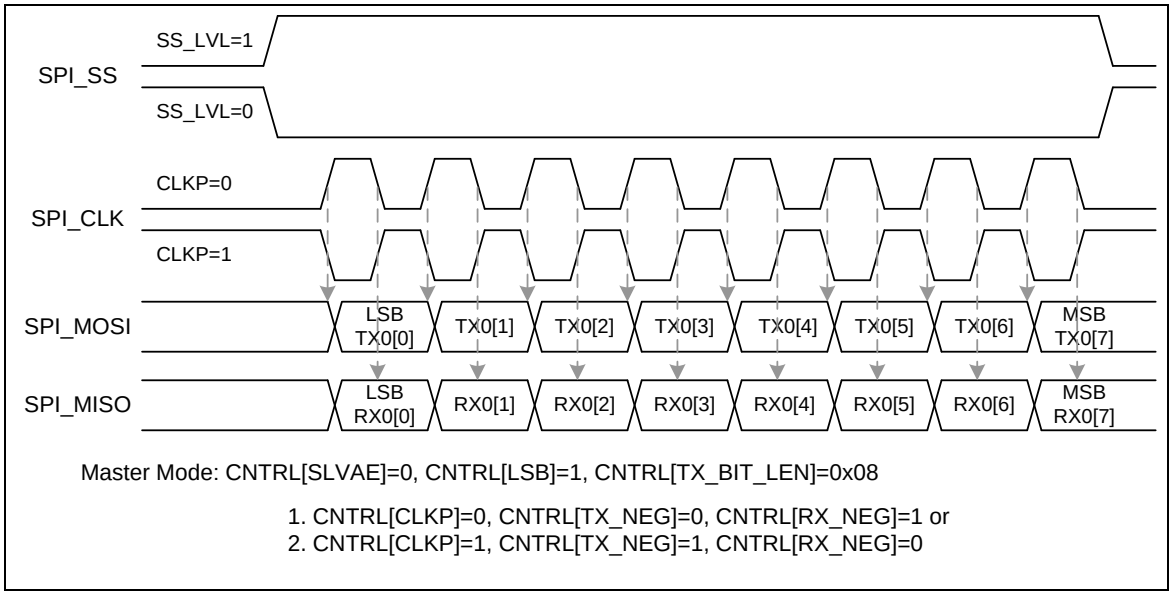


Figure 5-69 SPI Timing in Master Mode (Alternate Phase of SPICLK)

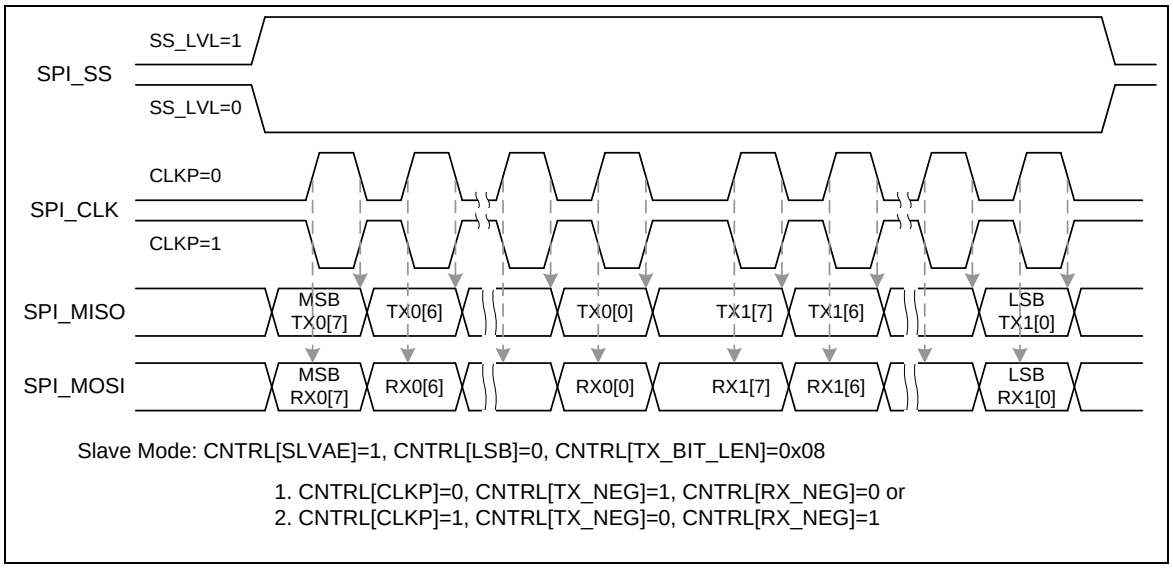


Figure 5-70 SPI Timing in Slave Mode

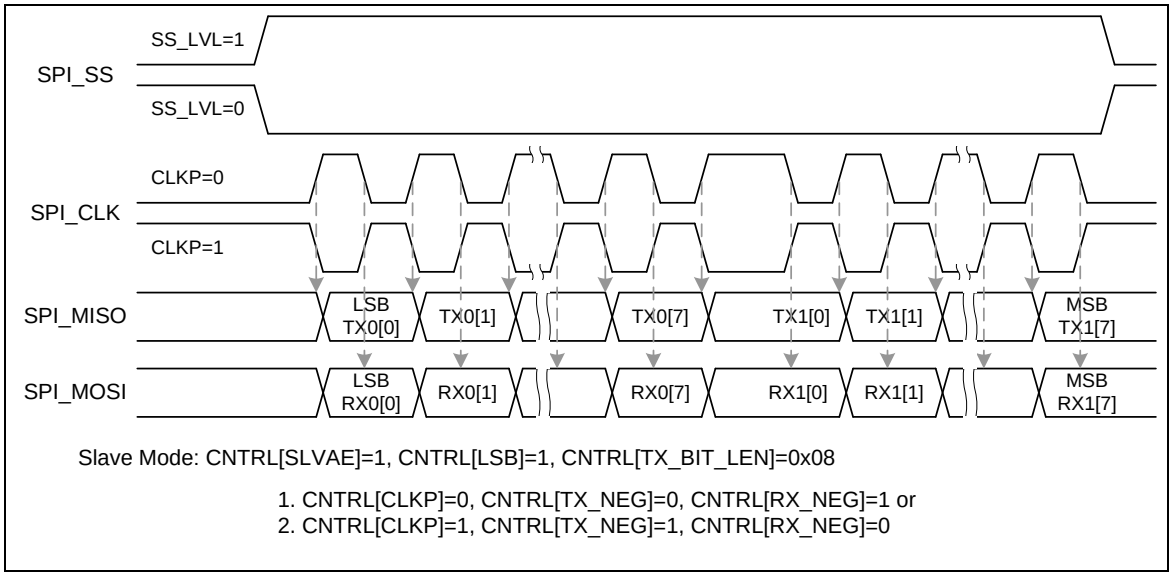


Figure 5-71 SPI Timing in Slave Mode (Alternate Phase of SPICLK)

5.9.6 Programming Examples

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

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

The operation flow is as follows.

1) Set the DIVIDER (SPI_DIVIDER [7:0]) register to determine the output frequency of SPI clock.
2) Write the SPI_SSR register a proper value for the related settings of Master mode:
    1. Disable the Automatic Slave Select bit AUTOSS(SPI_SSR[3] = 0).
    2. Select low level trigger output of slave select signal in the Slave Select Active Level bit SS_LVL (SPI_SSR[2] = 0) and Slave Select Level Trigger bit SS_LTRIG (SPI_SSR[4] = 1).
    3. Select slave select signal to be output active at the I/O pin by setting the Slave Select Register bits SSR[0] (SPI_SSR[0]) to active the off-chip slave device.
3) Write the related settings into the SPI_CNTRL register to control the SPI master actions
    1. Set this SPI controller as master device in SLAVE bit (SPI_CNTRL[18] = 0).
    2. Force the SPI clock idle state at low in CLKP bit (SPI_CNTRL[11] = 0).
    3. Select data transmitted at negative edge of SPI clock in TX_NEG bit (SPI_CNTRL[2] = 1).
    4. Select data latched at positive edge of SPI clock in RX_NEG bit (SPI_CNTRL[1] = 0).
    5. Set the bit length of word transfer as 8-bit in TX_BIT_LEN bit field. (SPI_CNTRL[7:3] = 0x08).
    6. Set MSB transfer first in MSB bit (SPI_CNTRL[10] = 0).
4) If this SPI master attempts to transmit (write) one byte data to the off-chip slave device, write the byte data that will be transmitted into the SPI_TX0 register.
5) If this SPI master just only attempts to receive (read) one byte data from the off-chip slave device and does not care what data will be transmitted, the SPI_TX0 register does not need to be updated by software.
6) Enable the GO_BUSY bit (SPI_CNTRL [0] = 1) to start the data transfer with the SPI interface.
7) Waiting for SPI interrupt (if the Interrupt Enable IE bit is set) or just polling the GO_BUSY bit till it is cleared to 0 by hardware automatically.
8) Read out the received one byte data from SPI_RX0[7:0].
9) Go to 4) to continue another data transfer or set SSR [0] to 0 to inactivate the off-chip slave

device.

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

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

The operation flow is as follows.

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

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

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

   1. Set the SPI controller as slave device in SLAVE bit (SPI_CNTRL[18] = 1).
   2. Select the SPI clock idle state at high in CLKP bit (SPI_CNTRL[11] = 1).
   3. Select data transmitted at negative edge of SPI clock in TX_NEG bit (SPI_CNTRL[2] = 1).
   4. Select data latched at positive edge of SPI clock in RX_NEG bit (SPI_CNTRL[1] = 0).
   5. Set the bit length of word transfer as 8-bit in TX_BIT_LEN bit field (SPI_CNTRL[7:3] = 0x08).
   6. Set LSB transfer first in LSB bit (SPI_CNTRL[10] = 1).

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

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

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

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

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

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

5.10 Timer Controller (TMR)

5.10.1 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, event counting, interval measurement, clock generation, and delay timing. The timer can generate an interrupt signal upon time-out, or provide the current value during operation.

5.10.2 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 and continuous counting operation modes
- Time-out period = (Period of timer clock input) * (8-bit prescale counter + 1) * (24-bit TCMP)
- Maximum counting cycle time = (1 / T MHz) * (2^8) * (2^{24}), T is the period of timer clock
- 24-bit up counter value is readable through TDR (Timer Data Register)
- Supports event counting function to count the event from external pin
- Supports external pin capture function for interval measurement
- Supports external pin capture function for reset timer counter
- Supports chip wake-up from Idle/Power-down mode if a timer interrupt signal is generated (TIF set to 1)

5.10.3 Block Diagram

Each channel is equipped with an 8-bit prescale counter, a 24-bit up counter, a 24-bit compare register (TCMPR) and an interrupt request signal (TIF). Refer to Figure 5-72 for the timer controller block diagram. There are six options of clock sources for each channel. Figure 5-73 illustrates the clock source control function. Software can program the 8-bit prescale counter to decide the clock period to 24-bit up counter.

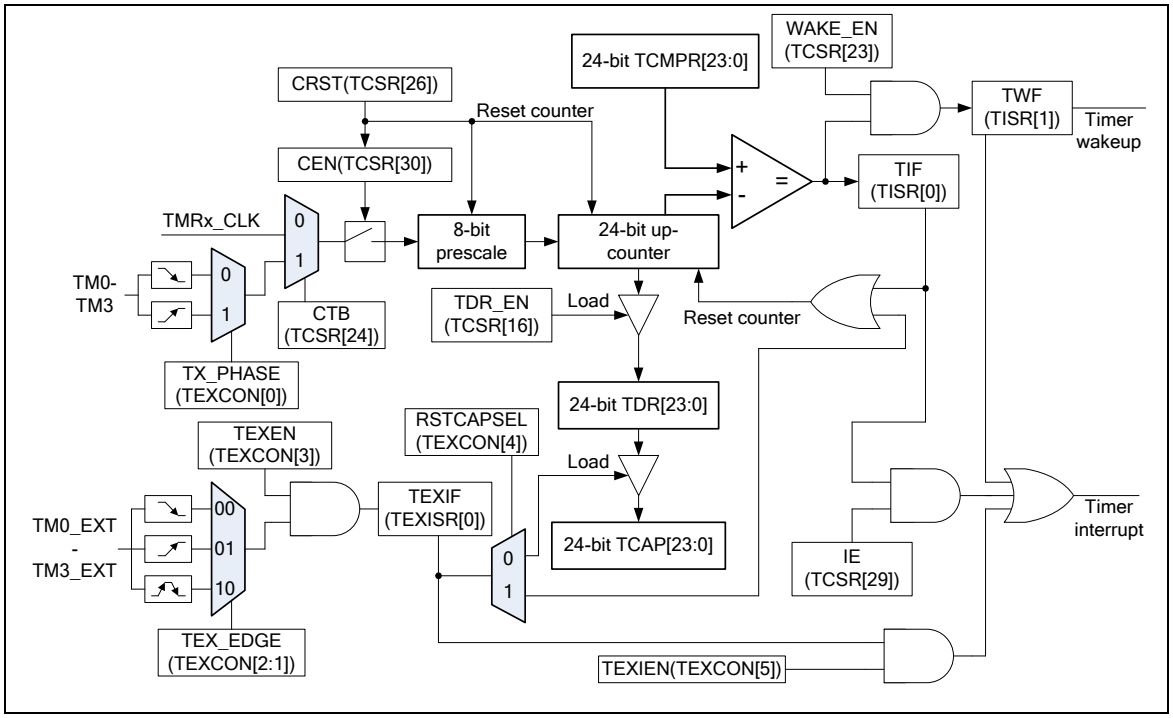


Figure 5-72 Timer Controller Block Diagram

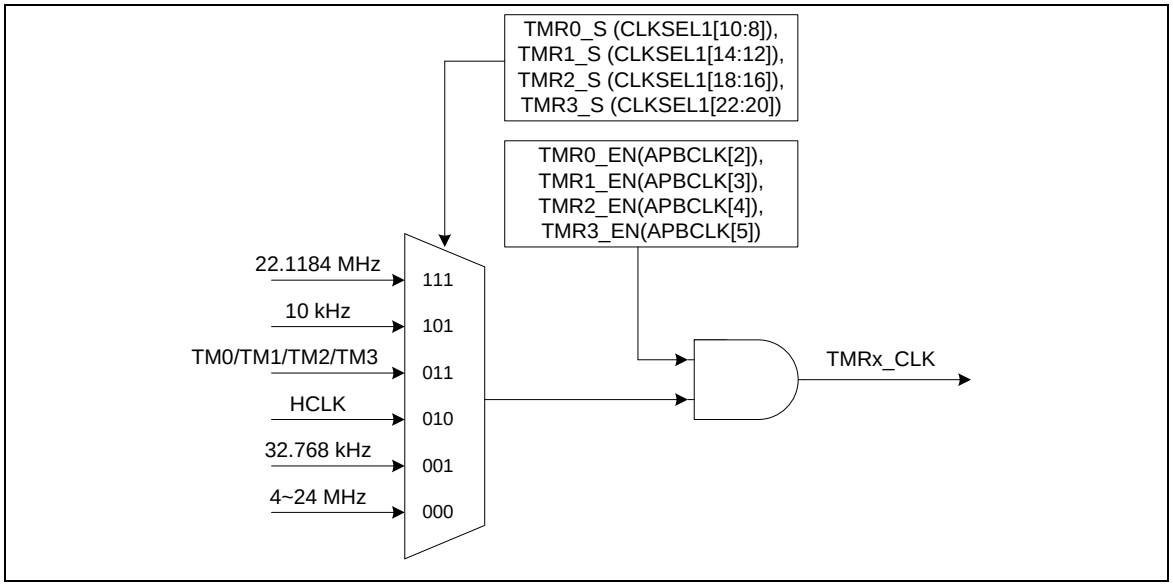


Figure 5-73 Clock Source of Timer Controller

5.10.4 Functional Description

Timer controller provides One-shot, Period, Toggle and Continuous Counting operation modes. The event counting function is also provided to count the events/counts from external pin and external pin capture function for interval measurement or reset timer counter. Each operating function mode is shown as follows:

5.10.4.1 **One–shot Mode**

If the timer is operated in One-shot mode (TCSR MODE[1:0] is 0x0) and CEN (TCSR[30] timer enable bit) is set to 1, the timer counter starts up counting. Once the timer counter value (TDR value) reaches timer compare register (TCMPR) value, the TIF (TISR[0] timer interrupt flag) will be set to 1. If IE (TCSR[29] interrupt enable bit) is set to 1, and TIF (TISR[0] timer interrupt flag) is 1 then the interrupt signal is generated and sent to NVIC to inform CPU for indicating that the timer counting overflow happens. If IE (TCSR[29] interrupt enable bit) is set to 0, no interrupt signal is generated.

In this operating mode, once the timer counter value (TDR value) reaches timer compare register (TCMPR) value, TIF (TISR[0] timer interrupt flag) will set to 1, timer counting operation stops and the timer counter value (TDR value) goes back to counting initial value then CEN (TCSR[30] timer enable bit) is cleared to 0 by timer controller automatically. That is to say, timer operates timer counting and compares with TCMPR value function only one time after programming the timer compare register (TCMPR) value and CEN (TCSR[30] timer enable bit) is set to 1. So, this operating mode is called One-Shot mode.

5.10.4.2 **Periodic Mode**

If the timer is operated in Period mode (TCSR MODE[1:0] is 0x1) and CEN (TCSR[30] timer enable bit) is set to 1, the timer counter starts up counting. Once the timer counter value (TDR value) reaches timer compare register (TCMPR) value, the TIF (TISR[0] timer interrupt flag) will set to 1. If IE (TCSR[29] interrupt enable bit) is set to 1, and TIF (TISR[0] timer interrupt flag) is 1 then the interrupt signal is generated and sent to NVIC to inform CPU for indicating that the timer counting overflow happens. If IE (TCSR[29] interrupt enable bit) is set to 0, no interrupt signal is generated.

In this operating mode, once the timer counter value (TDR value) reaches timer compare register (TCMPR) value, TIF (TISR[0] timer interrupt flag) will set to 1, the timer counter value (TDR value) goes back to counting initial value and CEN (TCSR[30] timer enable bit) is kept at 1 (counting enable continuously) and timer counter operates up counting again. If TIF (TISR[0] timer interrupt flag) is cleared by software, once the timer counter value (TDR value) reaches timer compare register (TCMPR) value again, TIF (TISR[0] timer interrupt flag) will set to 1 also. That is to say, timer operates timer counting and compares with TCMPR value function periodically. The timer counting operation does not stop until the CEN (TCSR[30] timer enable bit) is set to 0. The interrupt signal is also generated periodically. So, this operating mode is called Periodic mode.

5.10.4.3 **Toggle Mode**

If the timer is operated in Toggle mode (TCSR MODE[1:0] is 0x2) and CEN (TCSR[30] timer enable bit) is set to 1, the timer counter starts up counting. Once the timer counter value (TDR value) reaches timer compare register (TCMPR) value, the TIF (TISR[0] timer interrupt flag) will set to 1. If IE (TCSR[29] interrupt enable bit) is set to 1, and TIF (TISR[0] timer interrupt flag) is 1 then the interrupt signal is generated and sent to NVIC to inform CPU for indicating that the timer counting overflow happens. If IE (TCSR[29] interrupt enable bit) is set to 0, no interrupt signal is generated.

In this operating mode, once the timer counter value (TDR value) reaches timer compare register (TCMPR) value, TIF (TISR[0] timer interrupt flag) will set to 1, toggle out signal (TMx pin) is set to

1, the timer counter value (TDR value) goes back to counting initial value and CEN (TCSR[30] timer enable bit) is kept at 1 (counting enable continuously) and timer counter operates up counting again. If TIF (TISR[0] timer interrupt flag) is cleared by software, once the timer counter value (TDR value) reaches timer compare register (TCMPR) value again, TIF (TISR[0] timer interrupt flag) will set to 1 also and toggle out signal (TMx pin) is set to 0. The timer counting operation does not stop until the CEN (TCSR[30] timer enable bit) is set to 0. Thus, the toggle output signal (TMx pin) is changing back and forth with 50% duty cycle. So, this operating mode is called Toggle mode.

5.10.4.4 Continuous Counting Mode

If the timer is operated in Continuous Counting mode (TCSR MODE[1:0] is 0x3) and CEN (TCSR[30] timer enable bit) is set to 1, the timer counter starts up counting. Once the timer counter value (TDR value) reaches timer compare register (TCMPR) value, the TIF (TISR[0] timer interrupt flag) will set to 1. If IE (TCSR[29] interrupt enable bit) is set to 1, and TIF (TISR[0] timer interrupt flag) is 1 then the interrupt signal is generated and sent to NVIC to inform CPU for indicating that the timer counting overflow happens. If IE (TCSR[29] interrupt enable bit) is set to 0, no interrupt signal is generated.

In this operating mode, once the timer counter value (TDR value) reaches timer compare register (TCMPR) value, TIF (TISR[0] timer interrupt flag) will set to 1 and CEN (TCSR[30] timer enable bit) is kept at 1 (counting enable continuously) and timer counter continuous counting without reload the timer counter value (TDR value) to counting initial value. User can change different timer compare register (TCMPR) value immediately without disabling timer counter and restarting timer counter counting.

For example, the timer compare register (TCMPR) value is set as 80, first. (The timer compare register (TCMPR) should be less than 2^{24} and be greater than 1). Once the timer counter value (TDR value) reaches to 80, TIF (TISR[0] timer interrupt flag) will set to 1 and (TCSR[30] timer enable bit) is kept at 1 (counting enable continuously) and timer counter value (TDR value) will not goes back to 0, it continues counting to 81, 82, 83,⋯ to (2^{24} -1) then 0, 1, 2, 3, ⋯ to 2^{24} -1 again and again. Next, if user programs timer compare register (TCMPR) value as 200 and the TIF (TISR[0] timer interrupt flag) is cleared to 0, then TIF (TISR[0] timer interrupt flag) will set to 1 again when timer counter value (TDR value) reaches to 200. At last, user programs timer compare register (TCMPR) value as 500 and clears TIF (TISR[0] timer interrupt flag) to 0, then TIF (TISR[0] timer interrupt flag) will set to 1 again when timer counter value (TDR value) reaches to 500. In this mode, the timer counter value (TDR value) is keeping up counting always even if TIF (TISR[0] timer interrupt flag) is 1. So, this operation mode is called as Continuous Counting mode.

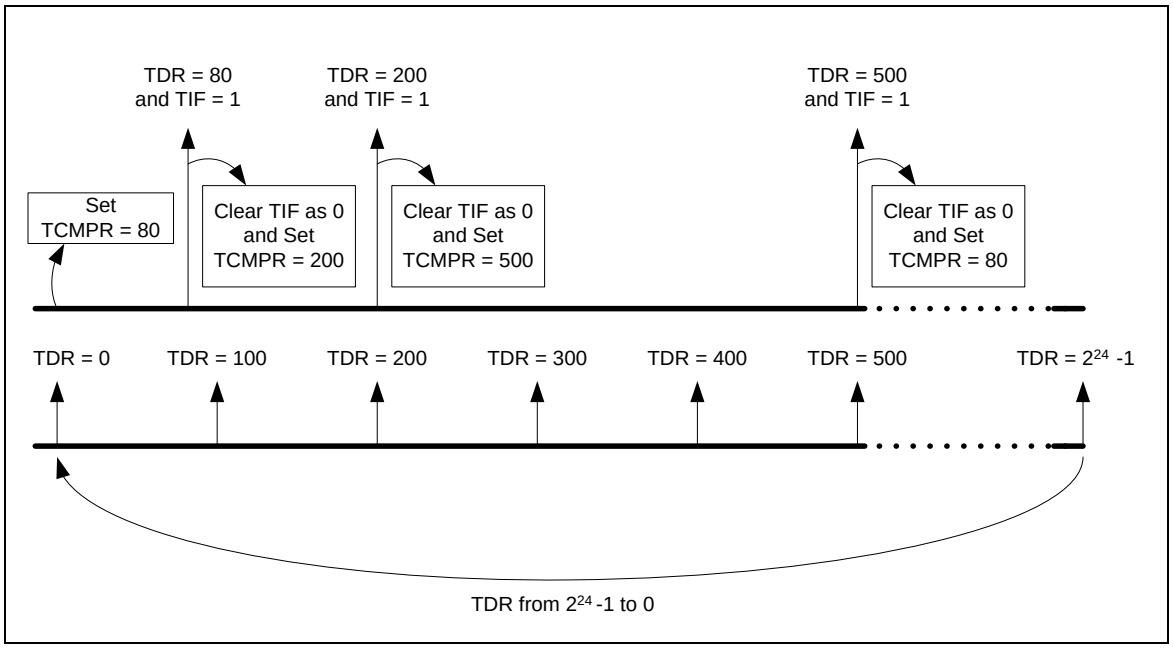


Figure 5-74 Continuous Counting Mode

5.10.4.5 Event Counting Function

An application which can count the events/counts from TMx pin (event counting pin) is called as event counting function. In this mode, most of the timer control registers are the same with the timer operating function mode except TCSR MODE[1:0] must set to 0x2 (toggle out mode will be disabled) and the clock source of timer controller, TMRx_CLK, in Figure 5-73 should be set as HCLK. When status transition on TMx pin, the event counter value (TDR value) will be counted according to TX_PHASE (TEXCON[0] timer external count phase) setting. TCDB (TEXCON[7] timer counter pin de-bounce enable) bit is for enabled or disabled edge detection de-bounce circuit of TMx pin. The max frequency of event counting source on TMx pin should be less than 1/3 HCLK if TCDB (TEXCON[7] timer counter pin de-bounce enable) is 0 or less than 1/8 HCLK if TCDB (TEXCON[7] timer counter pin de-bounce enable) is 1. Otherwise, the event counter value (TDR value) will not be counted normally.

5.10.4.6 External Pin Capture Function

In this mode, timer will monitor the transition of TMx_EXT pin (external capture pin) to save the timer counter value (TDR value) to timer capture value (TCAP value) or reset the timer counter value (TDR value) to 0. And the clock source of timer controller, TMRx_CLK, in Figure 5-73 should be set as HCLK.

If RSTCAPSEL (TEXCON[4] timer external reset counter / capture mode select) bit is 0, the transition on TMx_EXT pin is used as timer counter capture function. And when the transition of TMx_EXT pin matches the TEX_EDGE (TEXCON[2:1] timer external pin edge detect) setting, TEXIF (TEXISR[0] timer external interrupt flag) will set to 1 and the timer counter value (TDR value) will be saved into timer capture value (TCAP value).

If RSTCAPSEL (TEXCON[4] timer external reset counter / capture mode select) bit is 1, the transition on TMx_EXT pin is used as timer counter reset function. In this mode, once the transition of TMx_EXT pin matches TEX_EDGE (TEXCON[2:1] timer external pin edge detect) setting, TEXIF (TEXISR[0] timer external interrupt flag) will set to 1 and the timer counter value (TDR value) will be reset to 0.

The max frequency of external capture source on TMx_EXT pin should be less than 1/3 HCLK if TEXDB (TEXCON[6] timer external capture pin de-bounce enable) is 0 or less than 1/8 HCLK if TEXDB (TEXCON[6] timer external capture pin de-bounce enable) is 1. Otherwise, the transition on TMx_EXT pin will not be detected and TEXIF (TEXISR[0] timer external interrupt flag) will not set to 1 normally.

5.11 Watchdog Timer (WDT)

5.11.1 Overview

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

5.11.2 Features

- 18-bit free running up counter for Watchdog Timer 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.
- System kept in reset state for a period of (1 / WDT_CLK) * 63
- Supports selectable Watchdog Timer reset delay period, it includes (1024+2)、(128+2) 、(16+2) or (1+2) WDT_CLK reset delay period.
- Supports force Watchdog Timer enabled after chip powered on or reset while CWDTEN (Config0[31] watchdog enable) bit is set to 0.
- Supports Watchdog Timer time-out wake-up function when WDT clock source is selected to 10 kHz low speed oscillator.

5.11.3 Block Diagram

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

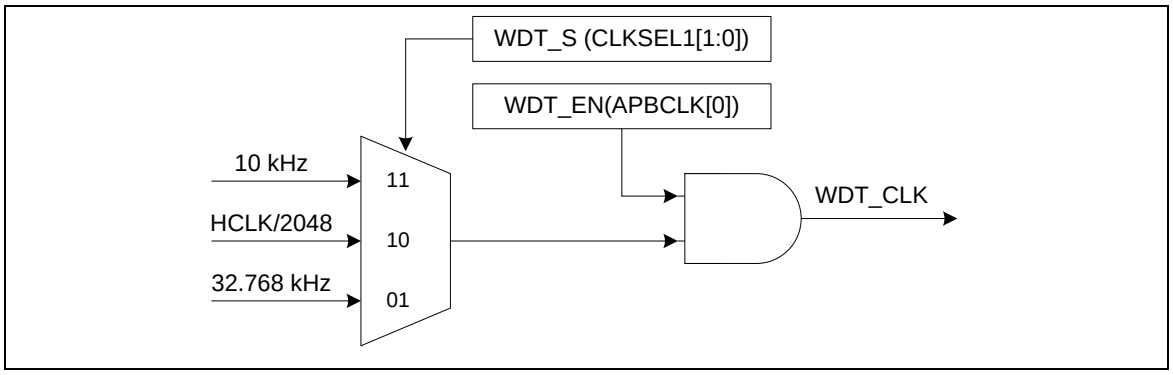


Figure 5-75 Watchdog Timer Clock Control

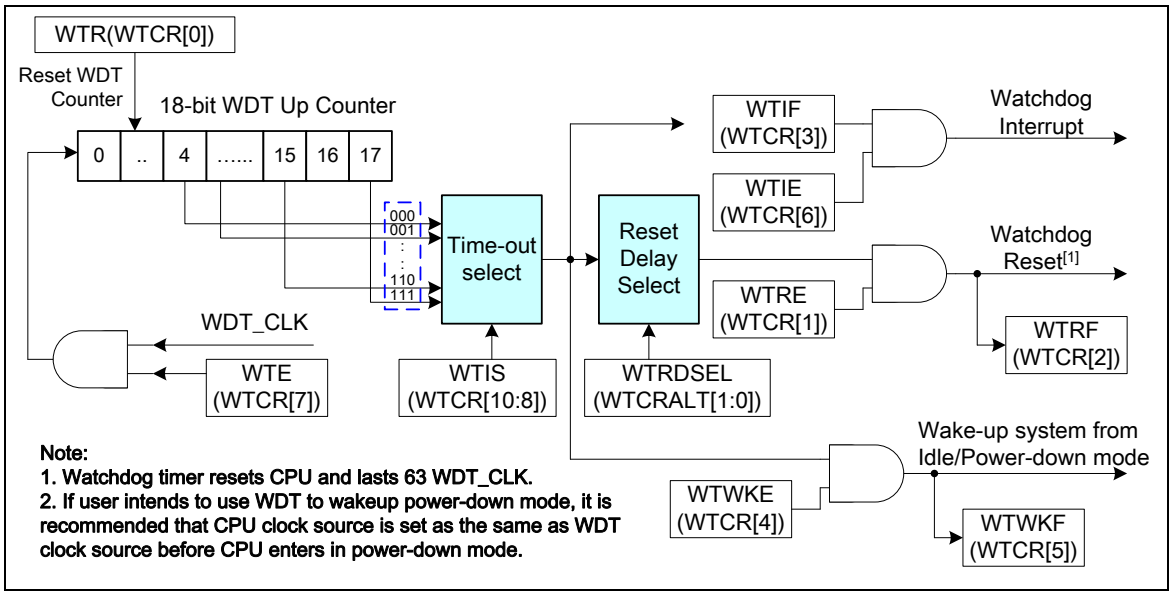


Figure 5-76 Watchdog Timer Block Diagram

5.11.4 Functional Description

The Watchdog Timer (WDT) includes an 18-bit free running up counter with programmable time-out intervals. Table 5-7 shows the WDT time-out interval selection and Figure 5-77 shows the timing of WDT time-out interval and reset period.

Setting WTE (WDTCR[7] WDT enable) bit to 1 will enable the WDT function and the WDT counter to start counting up. When the counter reaches the selected time-out interval (WTIS settings), WTIF (WTCR[3] WDT interrupt flag) will be set to 1 immediately, in the meanwhile, a specified WDT reset delay period (WTCRALT[1:0] WTRDSEL) follows the WTIF is setting to 1. User must set WTR (WDTCR[0] Reset WDT counter) 1 to reset the 18-bit WDT counter value to avoid generate WDT time-out reset signal before the WDT reset delay period expires. And WTR bit is cleared automatically by hardware after WDT counter is reset.

There are eight time-out interval period can be selected by setting WTIS (WTCR[10:8] WDT interval selection). If the WDT counter value has not been cleared after the specific WDT reset delay period expires, the WDT control will set WTRF (WTCR[2] WDT reset flag) to 1 if WTRE (WTCR[1] WDT reset enable) bit is enabled, then chip will reset immediately. This reset period will keep last 63 WDT clocks (T_{RST}) then chip restarts executing program from reset vector (0x0000_0000). And WTRF bit will not be cleared after WDT time-out reset the chip, user can check WTRF bit by software to recognize the system has been reset by WDT time-out reset or not.

The WDT also provides system wake-up function from Idle/Power-Down mode while WTIE (WTCR[6] WDT interrupt enable) bit is enabled and WTIF (WTCR[3] WDT interrupt flag) is set to 1.

WTIS	Time-out Interval Period T_{TIS}	Reset Delay Period T_{RSTD}
000	$2^4 * T_{WDT}$	(1/16/128/1024 + 2) * T_{WDT}
001	$2^6 * T_{WDT}$	(1/16/128/1024 + 2) * T_{WDT}
010	$2^8 * T_{WDT}$	(1/16/128/1024 + 2) * T_{WDT}
011	$2^{10} * T_{WDT}$	(1/16/128/1024 + 2) * T_{WDT}
100	$2^{12} * T_{WDT}$	(1/16/128/1024 + 2) * T_{WDT}
101	$2^{14} * T_{WDT}$	(1/16/128/1024 + 2) * T_{WDT}
110	$2^{16} * T_{WDT}$	(1/16/128/1024 + 2) * T_{WDT}
111	$2^{18} * T_{WDT}$	(1/16/128/1024 + 2) * T_{WDT}

Table 5-6 Watchdog Timer Time-out Interval Selection

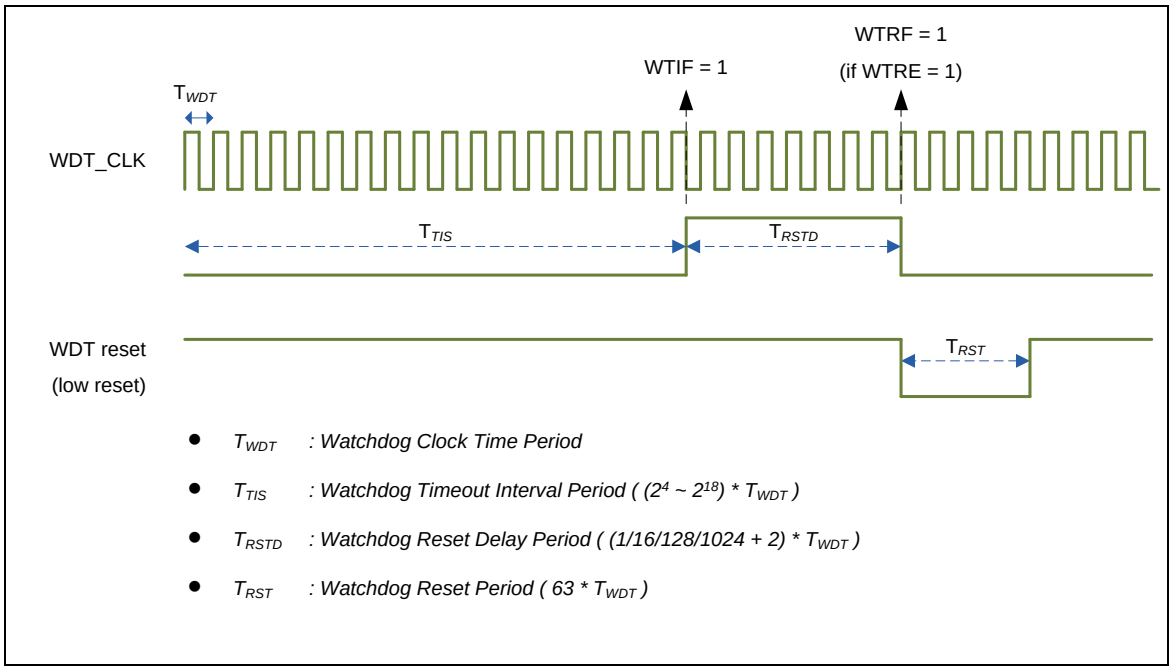


Figure 5-77 Watchdog Timer Time-out Interval and Reset Period Timing

5.12 Window Watchdog Timer (WWDT)

5.12.1 Overview

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

5.12.2 Features

- 6-bit down counter (WWDTVAL[5:0]) and 6-bit compare value (WWDTCR[21:16] – WINCMP value) to make the window period flexible
- Selectable maximum 11-bit WWDT clock prescale (WWDTCR[11:8] – PERIODSEL value) to make WWDT time-out interval variable

5.12.3 Block Diagram

The Window Watchdog Timer block diagram is shown as follows.

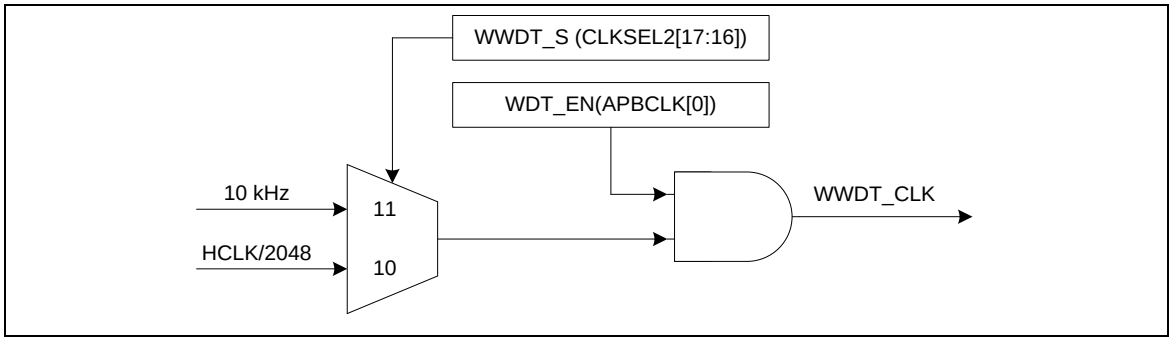


Figure 5-78 Window Watchdog Timer Clock Control

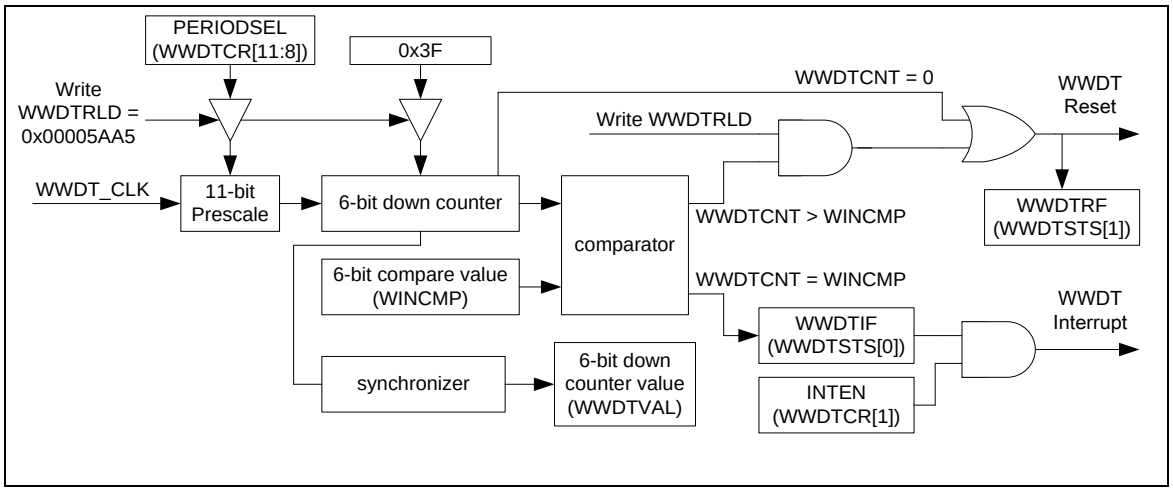


Figure 5-79 Window Watchdog Timer Block Diagram

5.12.4 Functional Description

The Window Watchdog Timer includes a 6-bit down counter with programmable prescale value to define different time-out intervals.

The clock source of 6-bit Window Watchdog Timer is based on system clock divide 2048 (HCLK/2048) or internal 10 kHz oscillator with a programmable maximum 11-bit prescale value. Also, the programmable 11-bit prescale value is controlled by PERIODSEL (WWDTCR[11:8] WWDT prescale period select) and the correlate of PERIODSEL and prescale value are listed in Table 5-7.

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

Table 5-7 Window Watchdog Timer Prescale Value Selection

The Window Watchdog Timer can be enabled only once by software setting WWDTEN (WWDTCR[0] WWDT enable) bit to 1 after chip power on or reset and the WWDT down counter will start counting from 0x3F and cannot be stopped by software unless chip has been reset again.

During down counting by the WWDT counter, the WWDTIF (WWDTSR[0] WWDT compare match interrupt flag) is set to 1 if the WWDT counter value is equal to WINCMP (WWDTCR[21:16] WWDT window compare register) value; if WWDTIE (WWDTCR[1] WWDT interrupt enable) is also set to 1 by software, the WWDT time-out interrupt signal is generated also while WWDTIF is set to 1 by hardware.

The WWDT time-out reset signal is generated when the WWDT counter value reaches to 0. Before WWDT counter down counting to 0, software can write 0x00005AA5 to WWDTRLD register to reload WWDT internal counter value to 0x3F to prevent WWDT time-out reset happen when current WWDT counter value (WWDTCVR value) is equal to or smaller than WINCMP

value. If current WWDT counter value (WWDTCVR value) is larger than WINCMP value and software writes 0x00005AA5 to the WWDTRLD register, WWDT reset signal will be generated to cause chip reset. Figure 5-80 shows the reset and reload behavior of WWDT.

To prevent program runs to disable Window Watchdog Timer counter counting unexpected, the control register WWDTCR can only be written once after chip is powered on or reset. Software cannot disable Window Watchdog Timer counter counting (WWDTEN), change time-out prescale period (PERIODSEL) or change window compare value (WINCMP) while WWDTEN bit has been enabled by software unless chip is reset.

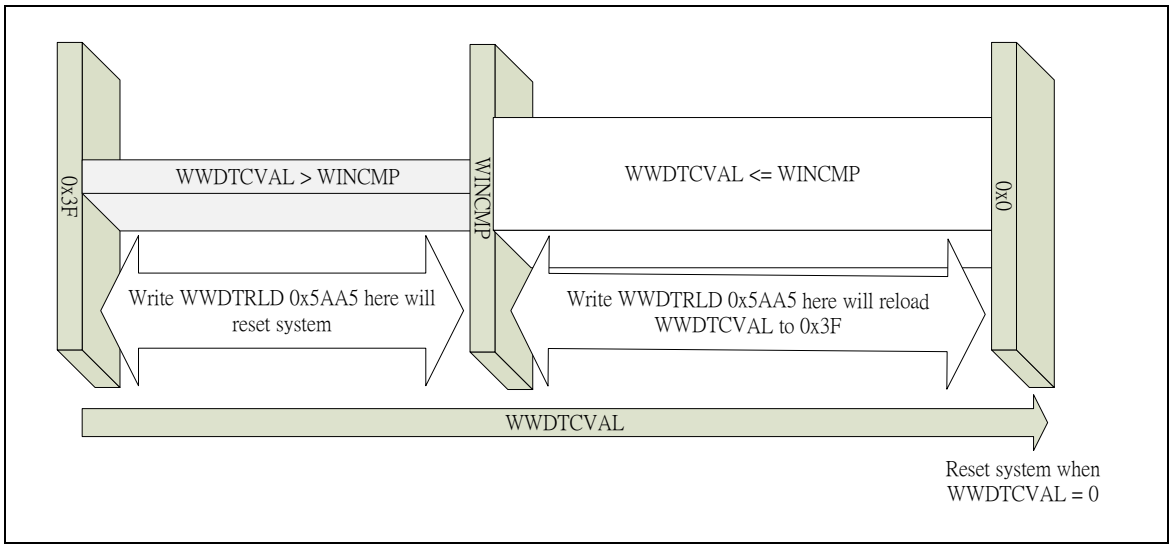


Figure 5-80 Window Watchdog Timer Reset and Reload Behavior

When software writes 0x00005AA5 to WWDTRLD register to reload WWDT counter value to 0x3F, it needs 3 WWDT clocks to sync reload command to actually perform reload action. It means if software set PERIODSEL (WWDTCR[11:8] WWDT prescale period select) to 0, the prescale value should be as 1, and the WINCMP (WWDTCR[21:16] WWDT window compare register) value must be larger than 2; otherwise, writing WWDTRLD by software to reload WWDT counter value to 0x3F is unavailable and WWDT time-out reset always happened. Table 5-8 shows the limitation of WINCMP.

Prescale Value	Valid WINCMP Value
1	0x3 ~ 0x3F
2	0x2 ~ 0x3F
Others	0x0 ~ 0x3F

Table 5-8 WINCMP Setting Limitation

5.13 UART Interface Controller (UART)

The NuMicro™ NUC200 series provides up to three channels of Universal Asynchronous Receiver/Transmitters (UART). UART0 supports High Speed UART and UART1~2 perform Normal Speed UART. Besides, only UART0 and UART1 support the flow control function.

5.13.1 Overview

The Universal Asynchronous Receiver/Transmitter (UART) performs a serial-to-parallel conversion on data received from the peripheral, and a parallel-to-serial conversion on data transmitted from the CPU. The UART controller also supports IrDA SIR, LIN master/slave mode and RS-485 mode functions. Each UART channel supports seven types of interrupts including:

- Transmitter FIFO empty interrupt (INT_THRE);
- Receiver threshold level reached interrupt (INT_RDA);
- Line status interrupt (parity error or frame error or break interrupt) (INT_RLS);
- Receiver buffer time-out interrupt (INT_TOUT);
- MODEM/Wake-up status interrupt (INT_MODEM);
- Buffer error interrupt (INT_BUF_ERR);
- LIN interrupt (INT_LIN).

Interrupts of UART0 and UART2 share the interrupt number 12 (vector number is 28); Interrupt number 13 (vector number is 29) only supports UART1 interrupt. Refer to the Nested Vectored Interrupt Controller chapter for System Interrupt Map.

The UART0 is built-in with a 64-byte transmitter FIFO (TX_FIFO) and a 64-byte receiver FIFO (RX_FIFO) that reduces the number of interrupts presented to the CPU. The UART1~2 are equipped with 16-byte transmitter FIFO (TX_FIFO) and 16-byte receiver FIFO (RX_FIFO). The CPU can read the status of the UART at any time during the operation. The reported status information includes the type and condition of the transfer operations being performed by the UART, as well as 4 error conditions (parity error, frame error, break interrupt and buffer error) probably occur while receiving data. The UART includes a programmable baud rate generator that is capable of dividing clock input by divisors to produce the serial clock that transmitter and receiver need. The baud rate equation is Baud Rate = UART_CLK / M * [BRD + 2], where M and BRD are defined in Baud Rate Divider Register (UA_BAUD). Table 5-9 lists the equations in the various conditions and Table 5-10 lists the UART baud rate setting table.

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

Table 5-9 UART Baud Rate Equation

System clock = internal 22.1184 MHz high speed oscillator						
Baud Rate	Mode 0		Mode 1		Mode 2	
	Parameter	Register	Parameter	Register	Parameter	Register

921600	x	x	A=0,B=11	0x2B00_0000	A=22	0x3000_0016
460800	A=1	0x0000_0001	A=1,B=15 A=2,B=11	0x2F00_0001 0x2B00_0002	A=46	0x3000_002E
230400	A=4	0x0000_0004	A=4,B=15 A=6,B=11	0x2F00_0004 0x2B00_0006	A=94	0x3000_005E
115200	A=10	0x0000_000A	A=10,B=15 A=14,B=11	0x2F00_000A 0x2B00_000E	A=190	0x3000_00BE
57600	A=22	0x0000_0016	A=22,B=15 A=30,B=11	0x2F00_0016 0x2B00_001E	A=382	0x3000_017E
38400	A=34	0x0000_0022	A=62,B=8 A=46,B=11 A=34,B=15	0x2800_003E 0x2B00_002E 0x2F00_0022	A=574	0x3000_023E
19200	A=70	0x0000_0046	A=126,B=8 A=94,B=11 A=70,B=15	0x2800_007E 0x2B00_005E 0x2F00_0046	A=1150	0x3000_047E
9600	A=142	0x0000_008E	A=254,B=8 A=190,B=11 A=142,B=15	0x2800_00FE 0x2B00_00BE 0x2F00_008E	A=2302	0x3000_08FE
4800	A=286	0x0000_011E	A=510,B=8 A=382,B=11 A=286,B=15	0x2800_01FE 0x2B00_017E 0x2F00_011E	A=4606	0x3000_11FE

Table 5-10 UART Baud Rate Setting Table

The UART0 and UART1 controllers support the auto-flow control function that uses two low-level signals, nCTS (clear-to-send) and nRTS (request-to-send), to control the flow of data transfer between the chip 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 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 controller detects nCTS is asserted from external device. If a valid asserted nCTS is not detected the UART controller will not send data out.

The UART controllers also provides Serial IrDA (SIR, Serial Infrared) function (User must set IrDA_EN (UA_FUN_SEL [1]) to enable IrDA function). The SIR specification defines a short-range infrared asynchronous serial transmission mode with 1 start bit, 8 data bits, and 1 stop bit. The maximum data rate supports up to 115.2 Kbps (half duplex). The IrDA SIR block contains an IrDA SIR Protocol encoder/decoder. The IrDA SIR Protocol encoder/decoder is half-duplex only. So it cannot transmit and receive data at the same time. The IrDA SIR physical layer specifies a minimum 10ms transfer delay between transmission and reception, and this delay feature must be implemented by software.

The alternate function of UART controllers is LIN (Local Interconnect Network) function. The LIN mode is selected by setting the UA_FUN_SEL[1:0] to ’01’. In LIN mode, 1 start bit and 8 data bits format with 1 stop bit are required in accordance with the LIN standard.

For NuMicro™ NUC200 Series, another alternate function of UART controllers is RS-485 9-bit mode, and direction control provided by nRTS pin or can program GPIO (PB.2 for UART0_nRTS and PB.6 for UART1_nRTS) to implement the function by software. The RS-485 mode is selected by setting the UA_FUN_SEL register to select RS-485 function. The RS-485 transceiver control is implemented using the nRTS control signal from an asynchronous serial port to enable the RS-485 transceiver. In RS-485 mode, many characteristics of the receiving and transmitting are same as UART.

5.13.2 Features

- Full duplex, asynchronous communications
- Separates receive / transmit 64/16/16 bytes (UART0/UART1/UART2) entry FIFO for data payloads
- Supports hardware auto flow control/flow control function (nCTS, nRTS) and programmable nRTS flow control trigger level (UART0 and UART1 support)
- Programmable receiver buffer trigger level
- Supports programmable baud-rate generator for each channel individually
- Supports nCTS wake-up function (UART0 and UART1 support)
- Supports 7-bit receiver buffer time-out detection function
- UART0/UART1 can through DMA channels to receive/transmit data
- Programmable transmitting data delay time between the last stop and the next start bit by setting UA_TOR [DLY] register
- Supports break error, frame error, parity error and receive / transmit buffer overflow detect function
- Fully programmable serial-interface characteristics
  - Programmable 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

5.13.3 Block Diagram

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

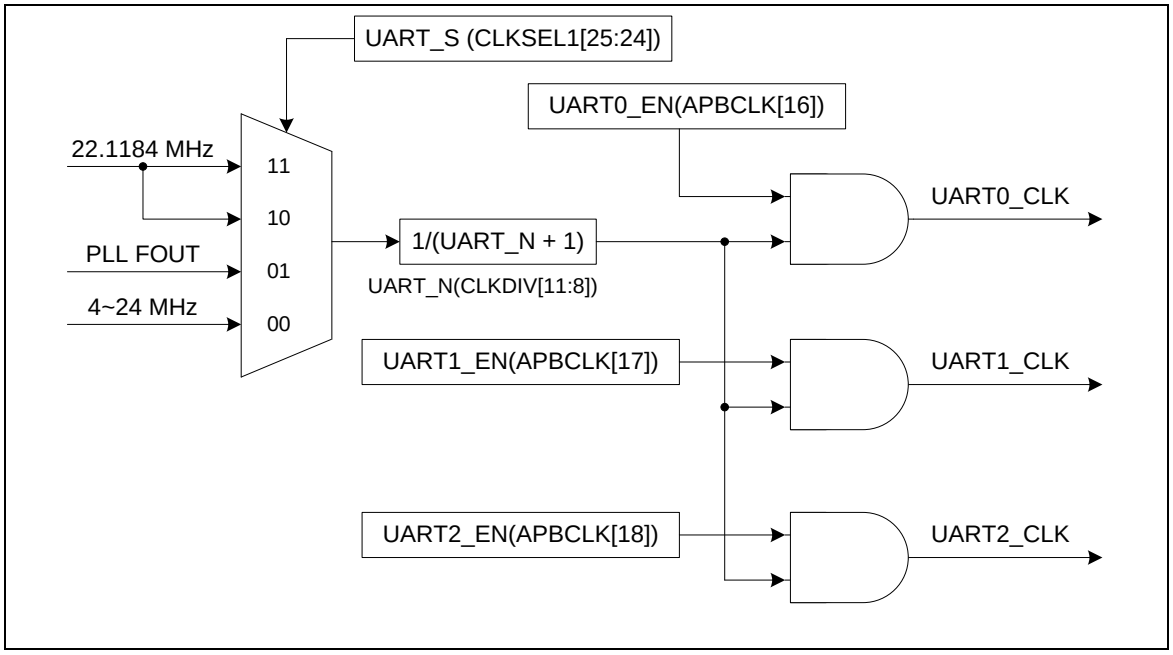


Figure 5-81 UART Clock Control Diagram

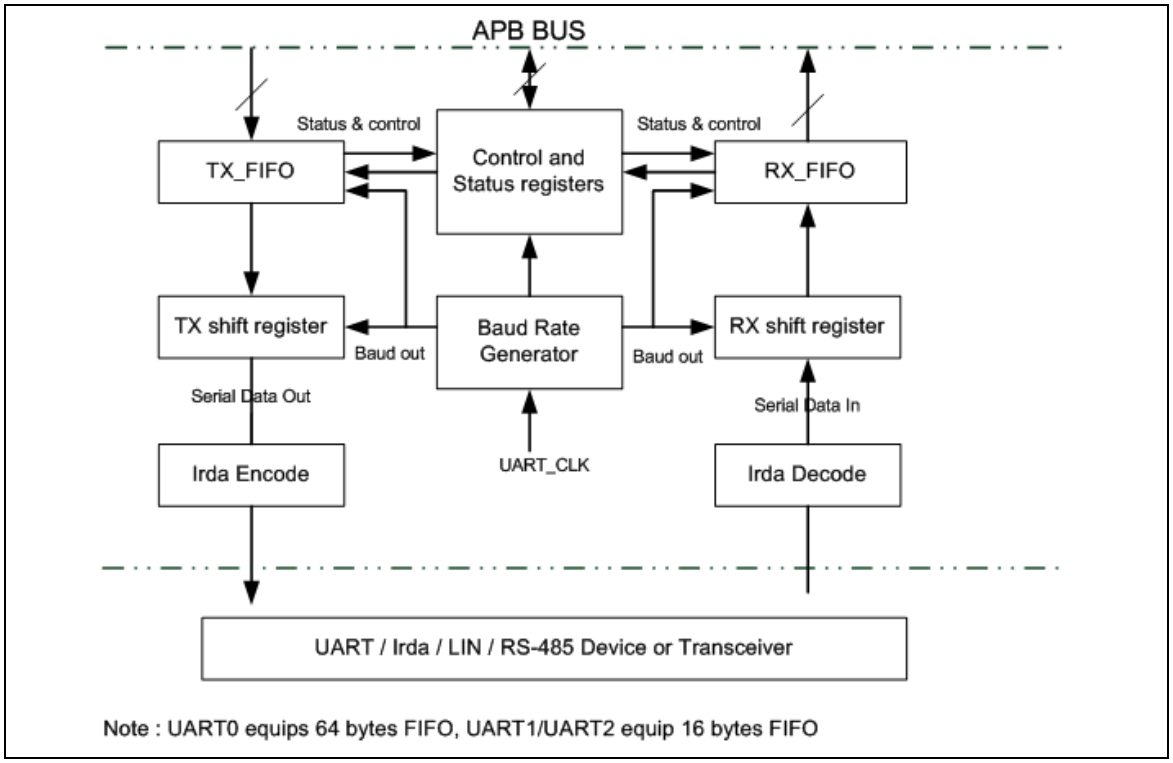


Figure 5-82 UART Block Diagram

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

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

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

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

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

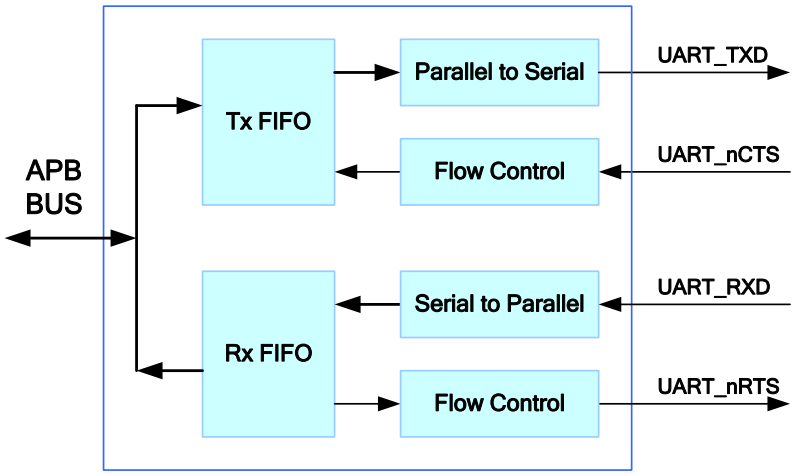
Baud Rate Generator
Divide the external clock by the divisor to get the desired baud rate. Refer to baud rate equation.

IrDA Encode
This block is IrDA encode control block.

IrDA Decode
This block is IrDA decode control block.

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

The following diagram demonstrates the auto-flow control block.



Note: It is only supported in UART0 and UART1.

Figure 5-83 Auto Flow Control Block Diagram

5.13.4 IrDA Mode

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

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

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

The following diagram demonstrates the IrDA control block.

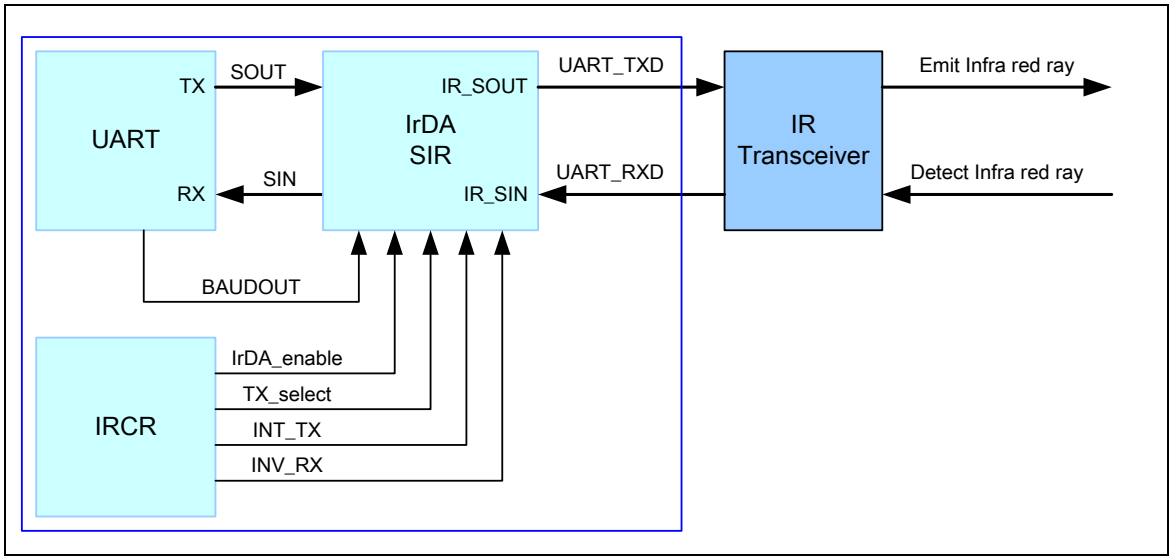


Figure 5-84 IrDA Block Diagram

5.13.4.1 *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.

5.13.4.2 *IrDA SIR Receive Decoder*

The IrDA SIR Receive Decoder demodulates the Return-to-Zero bit stream from the input detector and outputs the NRZ serial bits stream to the UART received data input. The decoder input is normally high in the idle state. (Because of this, IRCR bit 6 should be set as 1 by default)

A start bit is detected when the decoder input is LOW

5.13.4.3 *IrDA SIR Operation*

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

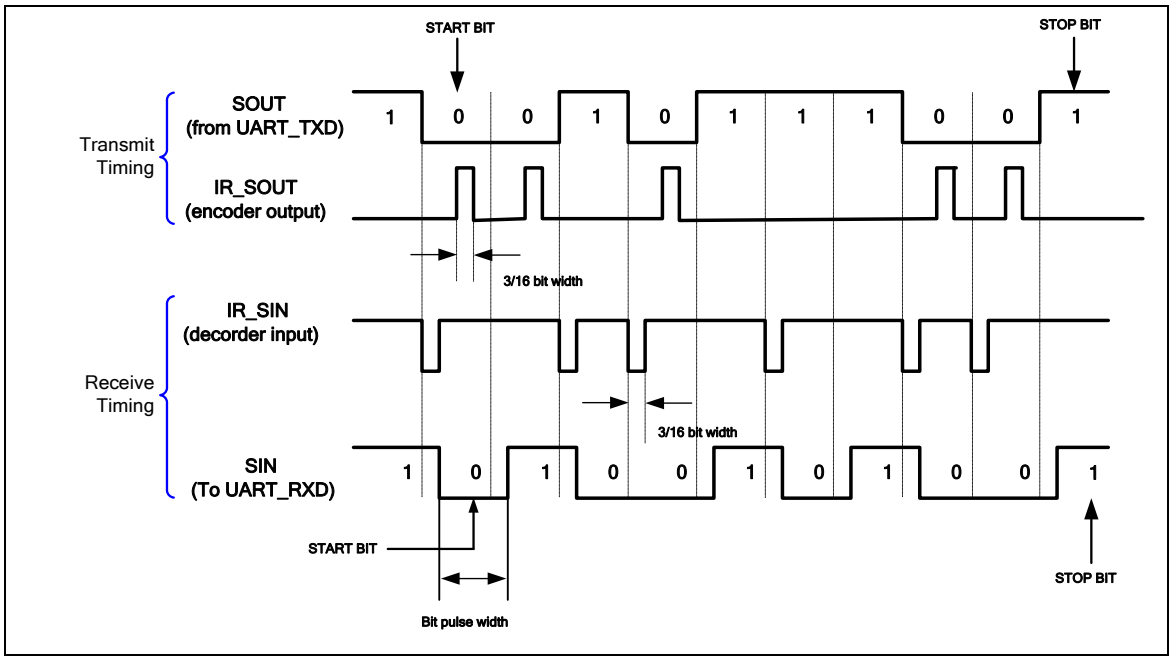


Figure 5-85 IrDA Timing Diagram

5.13.5 LIN (Local Interconnection Network) Mode

The UART supports LIN function, and LIN mode is selected by setting the UA_FUN_SEL[1:0] to '01'. The UART supports LIN break/delimiter generation and break/delimiter detection in LIN master mode, and supports header detection and automatic resynchronization in LIN Slave mode.

5.13.5.1 *Structure of LIN Frame*

According to the LIN protocol, all information transmitted is packed as frames; a frame consists of a header (provided by the master task) and a response (provided by a slave task), followed by a response (provided by a slave task). The header (provided by the master task) consists of a break field and a sync field followed by a frame identifier (frame ID). The frame identifier uniquely defines the purpose of the frame. The slave task is appointed for providing the response associated with the frame ID. The response consists of a data field and a checksum field. The following diagram is the structure of LIN Frame.

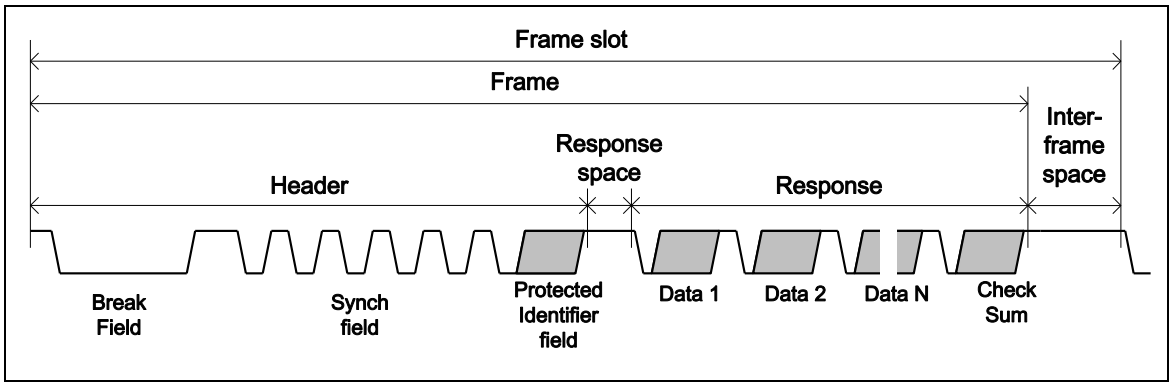


Figure 5-86 Structure of LIN Frame

5.13.5.2 *Structure of LIN Byte*

In LIN mode, each byte field is initiated by a START bit with value 0 (dominant), followed by 8 data bits (UA_LCR [WLS] = 2'b11) and no parity bit, LSB is first and ended by 1 stop bit (UA_LCR [NSB] = 1) with value 1 (recessive) in accordance with the LIN standard. The structure of Byte is shown as follows.

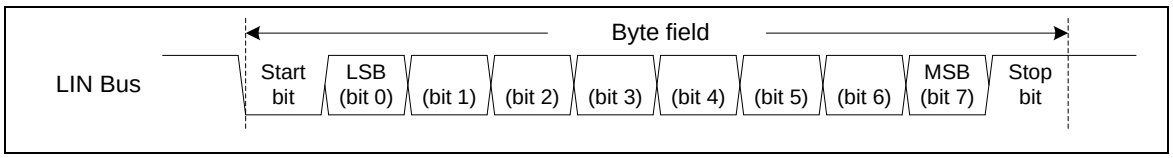


Figure 5-87 Structure of LIN Byte

5.13.5.3 *LIN Master Mode*

The UART controller supports 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 UA_LCR[WLS] = 11 and disable parity check by clearing UA_LCR[PBE] bit and configure the stop bit to 1 by clearing UA_LCR[NSB] 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 modes. The header selected mode can be “break field” or “break field and sync field” or “break field, sync field and frame ID field” by setting LIN_HEAD_SEL in UA_LIN_CTL register. If the selected header is “break field”, software must handle the following sequence to send a complete header to bus by filling sync data (0x55) and frame ID data to the UA_THR register. If the selected header is “break field and sync field”, software must handle the sequence to send a complete header to bus by filling the frame ID data to UA_THR register, and if the selected header is “break field, sync field and frame ID field”, hardware will control the header sending sequence automatically but software must filled frame ID data to UA_LIN_CTL [LIN_PID] register. When operating in header selected mode in which the selected header is “break field, sync field and frame ID field”, the frame ID parity bit can be calculated by software or hardware depending whether the UA_LIN_CTL [LIN_IDPEN] bit is set 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 UA_LIN_CTL [LIN_PID] first)

Table 5-11 LIN Header Selection in Master Mode

When operating in LIN data transmission, LIN bus transfer state can be monitored by hardware or software. User can enable hardware monitoring by setting UA_LIN_CTL [BIT_ERR_EN] to “1”, if the input pin (SIN) state is not equal to the output pin (SOUT) state in LIN transmitter state that hardware will generate an interrupt to CPU. Software can also monitor the LIN bus transfer state by checking the read back data in UA_RBR register. The following sequence is a program sequence example.

The procedure without software error monitoring in Master mode:

1. Fill Protected Identifier to [UA_LIN_CTL]LIN_PAD.
2. Select the hardware transmission header field including “break field + sync field + Protected identifier field” by setting UA_LIN_CTL [LIN_HEAD_SEL] = 10
3. Select the hardware transmission header field by setting UA_LIN_CTL [LIN_HEAD_SEL]).
4. Request header transmission by setting the LIN_SHD bit in the UA_LIN_CTL register.

5. Wait until UA_LIN_CTL[LIN_SHD] be cleared by hardware.
6. Wait until UA_FSR[TE_FLAG] set to “1” by hardware.

Note1: The default setting of break field + break/sync delimiter is 13 dominant bits (break field) and 1 recessive bit (break/sync delimiter). Software can change it by setting UA_LIN_CTL [LIN_BKFL] and UA_LIN_CTL [LIN_BS_LEN] to change the dominant bits.

Note2: The default setting of break/sync delimiter length is 1 bit time and the inter-byte spaces default setting is also 1 bit time. Software can change them by setting UA_LIN_CTL [LIN_BS_LEN] and UA_TOR [DLY].

Note3: If the header includes the “break field, sync field and frame ID field”, software must fill frame ID in UA_LIN_CTL [LIN_PID] register before trigger header transmission (setting the LIN_SHD bit in the UA_LIN_CTL register). The frame ID parity can be generated by software or hardware depends on UA_LIN_CTL [LIN_IDPEN]. If the parity generated by software (UA_LIN_CTL [LIN_IDPEN] = 0), software must fill 8 bit data (include 2 bit parity) in this field, and if the parity generated by hardware (UA_LIN_CTL [LIN_IDPEN] = 1), software fill ID0~ID5, hardware will calculi P0 and P1.

Procedure with software error monitoring in Master mode

1. Choose the hardware transmission header field only including “break field” by setting UA_LIN_CTL [LIN_HEAD_SEL] = 0x00.
2. Enable break detection function by setting LIN_BKDET_EN bit in UA_LIN_CTL.
3. Request break + break/sync delimiter transmission by setting the LIN_SHD bit in the UA_LIN_CTL register.
4. Wait until the LIN_BKDET_F flag in the UA_LIN_SR register be set to “1” by hardware..
5. Request sync field transmission by writing 0x55 into UA_THR register.
6. Wait until the RDA_IF flag in the UA_ISR register be 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 flag in the UA_ISR register be set to “1” by hardware and then read back the UA_RBR register.

LIN break and delimiter detection

When software enables the break detection function by setting LIN_BKDET_EN bit in UA_LIN_CTL register, the break detection circuit is activated. The break detection circuit is totally independent from the UART receiver.

When the break detection function is enabled, the circuit looks at the input SIN pin for a start signal. If circuit detected consecutive dominant is greater than 11 bits dominant followed by a recessive bit (delimiter), the UA_LIN_SR [LIN_BKDET_F] flag is set at the end of break field. If the UA_IER [LIN_IEN] bit =1, an interrupt will be generated. The behavior of the break detection and break flag are shown in the following figure.

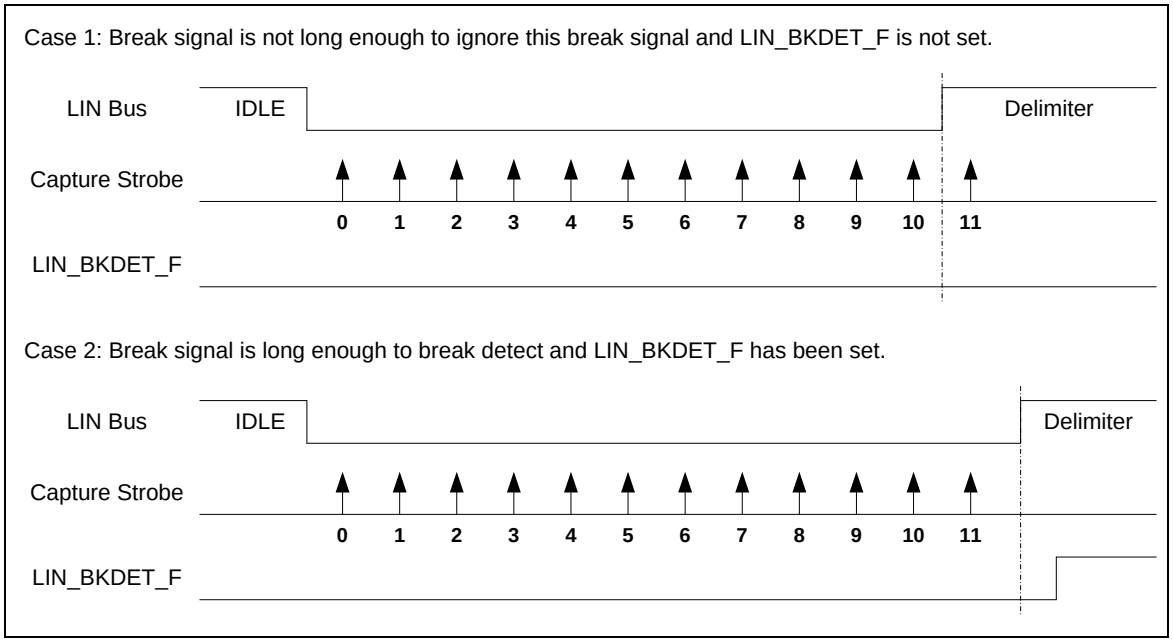


Figure 5-88 Break Detection in LIN Mode

LIN break and delimiter detection

The LIN master can transmit response (master is the publisher of the response) and receive response (master is the subscriber of the response). When the master is the publisher of the response, the master sends response by writing the UA_THR register. If the master is the subscriber of the response, the master will receive response from other slave node.

5.13.5.4 *LIN Slave Mode*

The UART controller supports LIN Slave mode by setting the LINS_EN bit in UA_LIN_CTL register. 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 UA_LCR[WLS] = 11and disable parity check by clearing UA_LCR[PBE] bit and configure the stop bit to 1 by clearing UA_LCR[NSB] bit.
3. Select LIN function mode by setting UA_FUN_SEL register.
4. Enable LIN slave mode by setting the LINS_EN bit in UA_LIN_CTL.

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 the slave header detection function by setting the LINS_HDET_EN bit in UA_LIN_CTL 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 flag in UA_LIN_SR register will be set (If the UA_IER [LIN_IEN] bit =1, an interrupt will be generated). User can enable the frame ID parity check function by setting LIN_IDPEN bit in UA_LIN_CTL register. If only received frame ID parity is not correct (break and sync filed are correct), the [UA_LIN_SR]LIN_IDPERR_F flag (If the UA_IER [LIN_IEN] bit =1, an interrupt will be generated) and UA_LIN_SR [LINS_HDET_F] both will be set. User can also put LIN in mute mode by setting LIN_MUTE_EN bit in UA_LIN_CTL register. This mode allows detection of headers only (break + sync + frame ID) and prevents the reception of any other characters. In order to avoid bit rate tolerance, the controller supports automatic resynchronization function to avoid clock deviation error, user can enable this feature by setting LINS_ARS_EN bit in UA_LIN_CTL register.

LIN response transmission

The LIN slave node can transmit response and receive response. When slave node is the publisher of the response, the slave node sends response by filling data to the UA_THR register. If the slave node is the subscriber of the response, the slave node receives data from LIN bus.

LIN header time-out error

The LIN slave controller contains a header time-out counter. If the entire header is not received within the maximum time limit of 57 bit times, the header error flag (UA_FSR [LIN_HERR_F]) will be set. The time-out counter is enabled at each break detect edge and stopped in the following conditions.

- A LIN frame ID field has been received.
- The header error flag asserts.
- Writing 1 to the LINS_SYNC_F bit in UA_LIN_SR register to re-search a new frame header.

Mute mode and LIN exit from mute mode condition

In Mute mode, a LIN slave node will not receive any data until specified condition occurred. It allows header detection only and prevents the reception of any other characters. User can enable Mute mode by setting the LIN_MUTE_EN bit in UA_LIN_CTL register and exiting from Mute mode condition can be selected by LIN_HEAD_SEL in UA_LIN_CTL register.

Note: It is recommended to set LIN slave node to Mute mode after checksum transmission.

The LIN slave controller exiting from Mute mode is described as follows: If [UA_LIN_CTL] LIN_HEAD_SEL is set to “break field”, when LIN slave controller detects a valid LIN break + delimiter, the controller will enable the receiver (exit from Mute mode) and subsequent data (sync data, frame ID data, response data) are received in RX-FIFO.

If [UA_LIN_CTL]LIN_HEAD_SEL is set to “break field and sync field”, when the LIN slave controller detects a valid LIN break + delimiter followed by a valid sync field without frame error, the controller will enable the receiver (exit from mute mode) and subsequent data(ID data, response data) are received in RX-FIFO. If [UA_LIN_CTL]LIN_HEAD_SEL is set to “break field, sync field and ID field”, when the LIN slave controller detects a valid LIN break + delimiter and valid sync field without frame error followed by ID data without frame error and received ID data matched UA_LIN_CTL [LIN_PID] value. The controller will enable the receiver (exit from mute mode) and subsequent data (response data) are received in RX-FIFO.

Slave mode non-automatic resynchronization

User can disable the automatic resynchronization function to fix the communication baud rate. When operating in Non-Automatic Resynchronization mode, software needs some initial process, and the initialization process flow of Non-Automatic Resynchronization mode is shown as follows:

1. Select the desired baud-rate by setting the UA_BAUD register.
2. Select LIN function mode by setting UA_FUN_SEL register.
3. Disable automatic resynchronization function by setting UA_LIN_CTL[LINS_ARS_EN] = 0.
4. Enable LIN slave mode by setting the LINS_EN bit in UA_LIN_CTL.

Slave mode with automatic resynchronization

User can enable the automatic resynchronization function by setting LINS_ARS_EN bit in UA_LIN_CTL register. 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 UA_LIN_CTL[LINS_ARS_EN] = 1.
4. Enable LIN slave mode by setting the LINS_EN bit in UA_LIN_CTL.

When the automatic resynchronization function is enabled, after each LIN break field, the time duration between five falling edges is sampled on engine clock and the result of this measurement is stored in an internal 13-bit register and the UA_BAUD register value will be automatically updated at the end of the fifth falling edge. If the measure timer (13-bit) overflows before five falling edges, then the header error flag (UA_LIN_SR [LIN_HERR_F]) will be set.

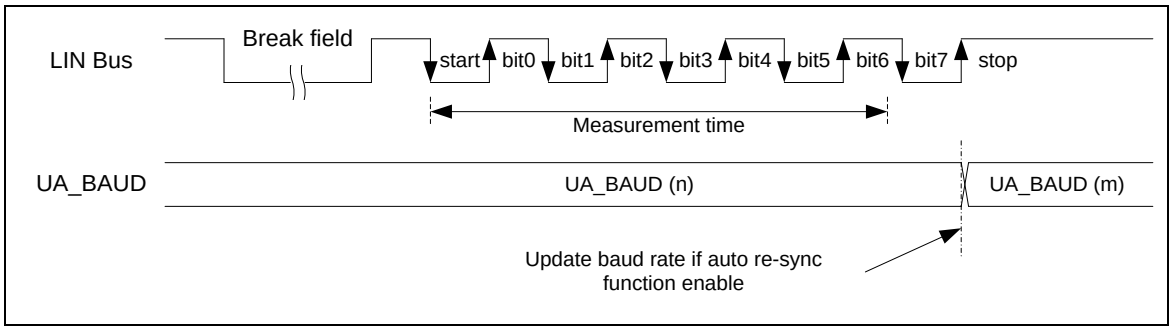


Figure 5-89 LIN Sync Field Measurement

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 engine clock 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 UA_LIN_CTL [LINS_DUM_EN] bit to enable auto reload initial baud rate value function. If the LINS_DUM_EN 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 bit will be cleared automatically. The behavior of LIN updated method as shown in the following figure.

Note1: It is recommended to set the LINS_DUM_EN bit before every checksum reception.

Note2: When a header error is detected, user must write 1 to LINS_SYNC_F bit in UA_LIN_SR register to re-search new frame header. When writing 1 to it, hardware will reload the initial baud-rate (TEMP_REG) and re-search new frame header.

Note3: When operating in Automatic Resynchronization mode, the baud rate setting must be mode2 (UA_BAUD [DIV_X_EN] and UA_BAUD [DIV_X_ONE] must be 1).

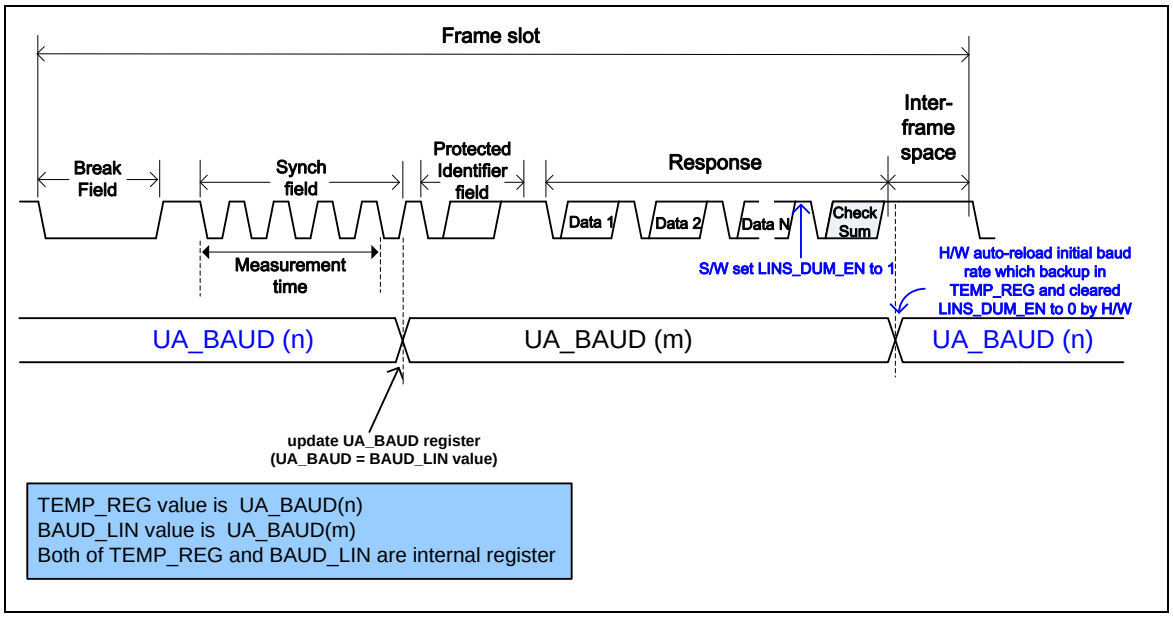


Figure 5-90 UA_BAUD Update Sequence in Automatic Resynchronization Mode when LINS_DUM_EN = 1

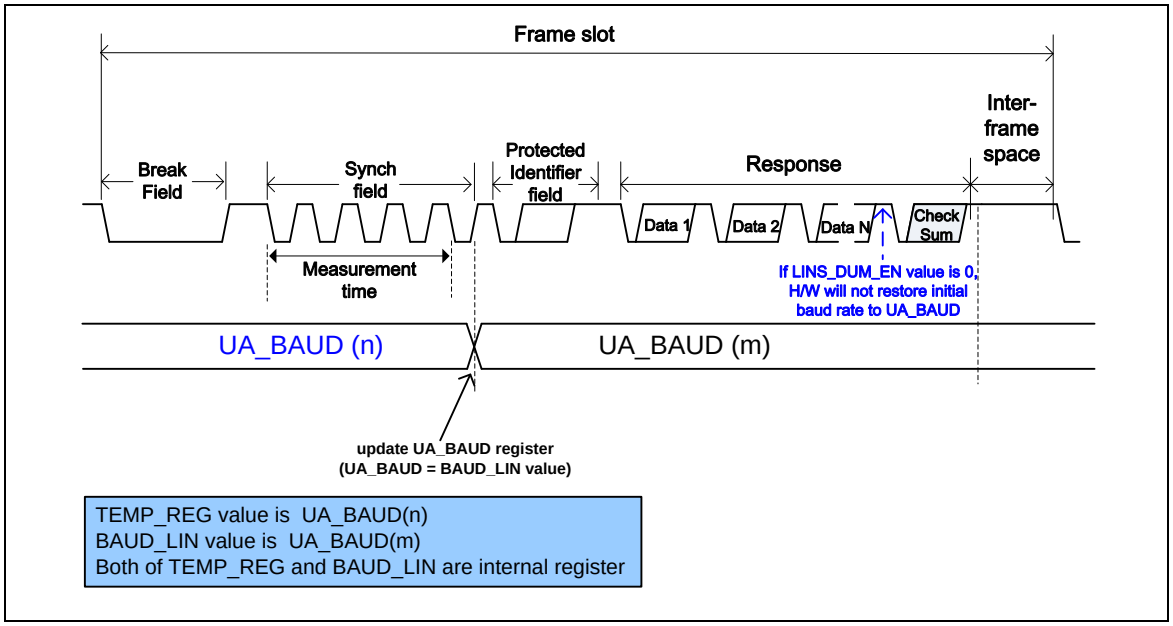


Figure 5-91 UA_BAUD Update Sequence in Automatic Resynchronization Mode when LINS_DUM_EN = 0

Deviation error on the sync field

When operating in Automatic Resynchronization mode, the controller will check the deviation error on the sync field. The deviation error is checked by comparing the current baud rate with the received sync field. Two checks are performed in parallel.

Check1: Based on measurement between the first falling edge and the last falling edge of the sync field.

- If the difference is more than 14.84%, the header error flag UA_LIN_SR[LINS_HERR_F]) will be set.
- If the difference is less than 14.06%, the header error flag UA_LIN_SR[LINS_HERR_F] will not be set.
- If the difference is between 14.84% and 14.06%, the header error flag UA_LIN_SR[LINS_HERR_F] may either set or not (depending on the data dephasing).

Check2: Based on measurement of time between each falling edge of the sync field.

- If the difference is more than 18.75%, the header error flag UA_LIN_SR[LINS_HERR_F] will be set.
- If the difference is less than 15.62%, the header error flag UA_LIN_SR[LINS_HERR_F] will not be set.
- If the difference is between 18.75% and 15.62%, the header error flag UA_LIN_SR[LINS_HERR_F] may either set or not (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 UA_LIN_CTL [LINS_DUM_EN] register (It is recommend setting the LINS_DUM_EN bit before every checksum reception)

LIN header error detection

In LIN Slave function mode, when user enables the header detection function by setting the LINS_HDET_EN bit in UA_LIN_CTL register, hardware will handle the header detect flow. If the header has an error, the LIN header error flag (UA_LIN_SR [LIN_HERR_F]) will be set and an interrupt is generated if the UA_IER [LIN_IEN] bit is set. When header error is detected, user must reset the detect circuit to re-search a new frame header by writing 1 to LINS_SYNC_F bit in UA_LIN_SR register to re-search a new frame header.

The LIN header error flag (UA_LIN_SR [LIN_HERR_F]) is set if one of the following conditions occurs:

- Break Delimiter is too short (less than 0.5 bit time).
- Frame error in sync field or Identifier field.
- The sync field data is not 0x55 (Non-Automatic Resynchronization mode).
- The sync field deviation error (With Automatic Resynchronization mode).
- The sync field measure time-out (With Automatic Resynchronization mode).
- LIN header reception time-out.

5.13.6 RS-485 Function Mode

The UART supports **RS-485 9-bit mode function**. The RS-485 mode is selected by setting the register UA_FUN_SEL[1:0]. The RS-485 transceiver control is implemented using the nRTS control signal from an asynchronous serial port. In RS-485 mode, many characteristics of the RX and TX are same as UART.

In RS-485 mode, the controller can configure it as an RS-485 addressable slave and the RS-485 master transmitter will identify an address character by setting the parity (9^{th} bit) to 1. For data characters, the parity is set to 0. Software can use the UA_LCR register to control the 9-th bit (when the PBE , EPE and SPE are set, the 9-th bit is transmitted 0 and when PBE and SPE are set and EPE is cleared, the 9-th bit is transmitted 1).

The controller 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 UA_ALT_CSR register, and can drive the transfer delay time between the last stop bit leaving the TX-FIFO and the de-assertion of by setting UA_TOR [DLY] register.

Note: When RS485 NMM or AAD mode is selected, the RS485 clock operating frequency should be less than or equal to half of PCLK clock operation frequency. Otherwise, RS485 cannot receive correct data.

RS-485 Normal Multidrop Operation Mode (NMM)

In RS-485 Normal Multidrop operation mode, first, software must decide which data before the address byte detected will be stored in RX-FIFO or not. If software wants to ignore any data before address byte detected, the flow is to set UA_FCR[RX_DIS] and then enable UA_ ALT_CSR [RS485_NMM] and the receiver will ignore any data until an address byte is detected (bit9 =1) and the address byte data will be stored in the RX-FIFO. If software wants to receive any data before address byte detected, the flow disables UA_FCR[RX_DIS] and then enables UA_ ALT_CSR[RS485_NMM] and the receiver will received any data.

If an address byte is detected (bit9 =1), it will generate an interrupt to CPU and UA_FCR[RX_DIS] can decide whether accepting the following data bytes are stored in the RX-FIFO. If software disables receiver by setting UA_FCR[RX_DIS] register, when a next address byte is detected, the controller will clear the UA_FCR[RX_DIS] bit and the address byte data will be stored in the RX-FIFO.

RS-485 Auto Address Detection Operation Mode (AAD)

In RS-485 Auto Address Detection Operation mode, the receiver will ignore any data until an address byte is detected (bit9 =1) and the address byte data matches the UA_ALT_CSR [ADDR_MATCH] value. The address byte data will be stored in the RX-FIFO. All the received byte data will be accepted and stored in the RX-FIFO until and address byte data not match the UA_ALT_CSR[ADDR_MATCH] value.

RS-485 Auto Direction Mode (AUD)

Another option function of RS-485 controllers is **RS-485 auto direction control function**. The RS-485 transceiver control is implemented 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 disable. User can set LEV_RTS in UA_MCR register to change the nRTS driving level.

Program Sequence Example:

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

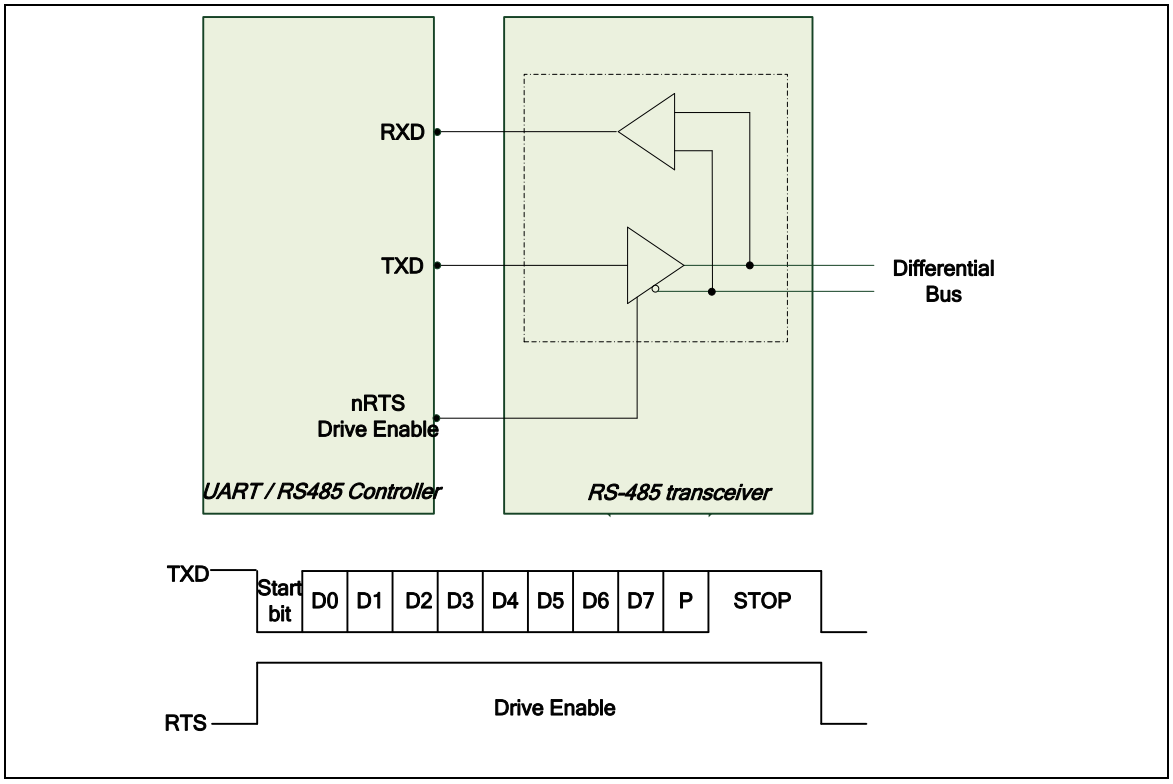


Figure 5-92 Structure of RS-485 Frame

5.14 PS/2 Device Controller (PS2D)

5.14.1 Overview

PS/2 device controller provides a basic timing control for PS/2 communication. All communication between the device and the host is managed through the PS2_CLK and PS2_DAT pins. Unlike PS/2 keyboard or mouse device controller, the receive/transmit code needs to be translated as meaningful code by firmware. The device controller generates the PS2_CLK signal after receiving a “Request to Send” state, but host has ultimate control over communication. Data of PS2_DAT line sent from the host to the device is read on the rising edge and sent from the device to the host is change after rising edge. A 16 bytes FIFO is used to reduce CPU intervention. Software can select 1 to 16 bytes for a continuous transmission.

5.14.2 Features

- Host communication inhibit and Request to Send state detection
- Reception frame error detection
- Programmable 1 to 16 bytes transmit buffer to reduce CPU intervention
- Double buffer for data reception
- Software override bus

5.14.3 Block Diagram

The PS/2 device controller consists of APB interface and timing control logic for PS2_DAT and PS2_CLK pins.

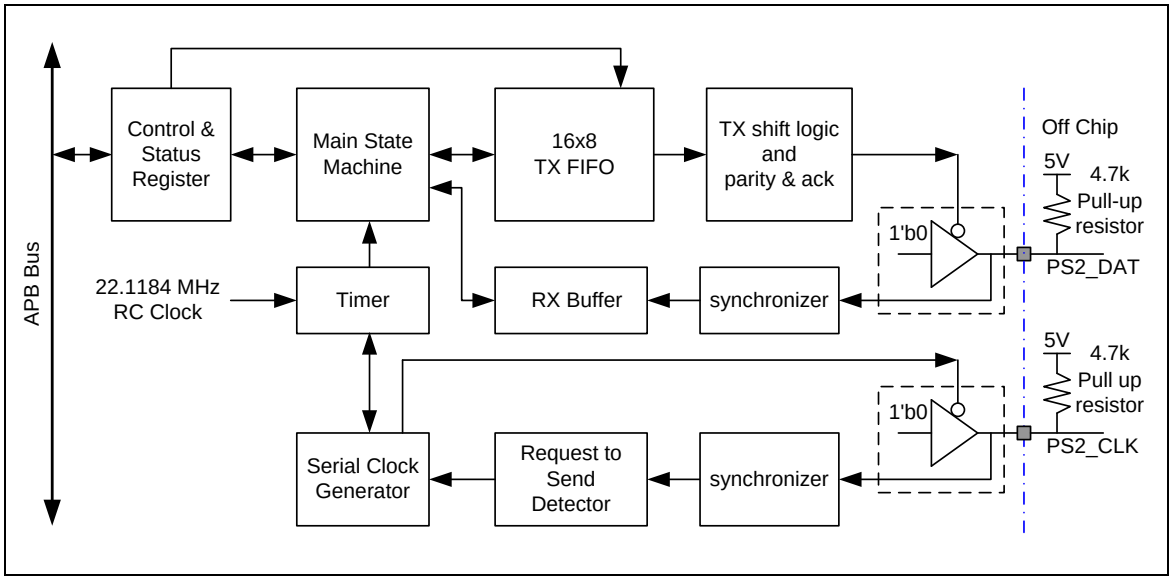


Figure 5-93 PS/2 Device Block Diagram

5.14.4 Functional Description

5.14.4.1 *Communication*

The PS/2 device implements a bidirectional synchronous serial protocol. The bus is "Idle" when both lines are high (open-collector). This is the only state where the device is allowed start to transmit PS2 data. The host has ultimate control over the bus and may inhibit communication at any time by pulling the PS2_CLK line low.

The PS2_CLK signal is generated by PS/2 device. If the host wants to send PS2 data, it must first inhibit communication from the device by pulling PS2_CLK low. The host then pulls PS2_DAT low and releases PS2_CLK. This is the "Request-to-Send" state and signals the device to start generating PS2_CLK pulses.

PS2_DAT	PS2_CLK	Bus State
High	High	Idle
High	Low	Communication Inhibit
Low	High	Host Request to Send

All data is transmitted one byte at a time and each byte is sent in a frame consisting of 11 or 12 bits. These bits are:

- 1 start bit, which is always 0
- 8 data bits, least significant bit first
- 1 parity bit (odd parity)
- 1 stop bit, which is always 1
- 1 acknowledge bit (host-to-device communication only)

The parity bit is set if there is an even number of 1's in the data bits and cleared to 0 if there is an odd number of 1's in the data bits. This is used for parity error detection that is the number of 1's in the data bits plus the parity bit and always adds up to an odd number then parity bit sets to 1. The device must check this bit and if incorrect it should respond as if it had received an invalid command.

The host may inhibit communication at any time by pulling the PS2_CLK line low for at least 100 us. If a transmission is inhibited before the 11th clock pulse, the device must abort the current transmission and prepare to resend the current data when host releases PS2_CLK. In order to reserve enough time for software to decode host command, the transmit logic is blocked by RXINT bit, software must clear the RXINT bit to start resend. Software can write CLRFIFO to 1 to reset FIFO pointer if needed.

Device-to-Host

The device uses a serial protocol with 11-bit frames. These bits are:

- 1 start bit, which is always 0
- 8 data bits, least significant bit first
- 1 parity bit (odd parity)
- 1 stop bit, which is always 1

The device writes a bit on the PS2_DAT line when PS2_CLK is high, and it is read by the host when PS2_CLK is low, which is illustrated in Figure 5-94.

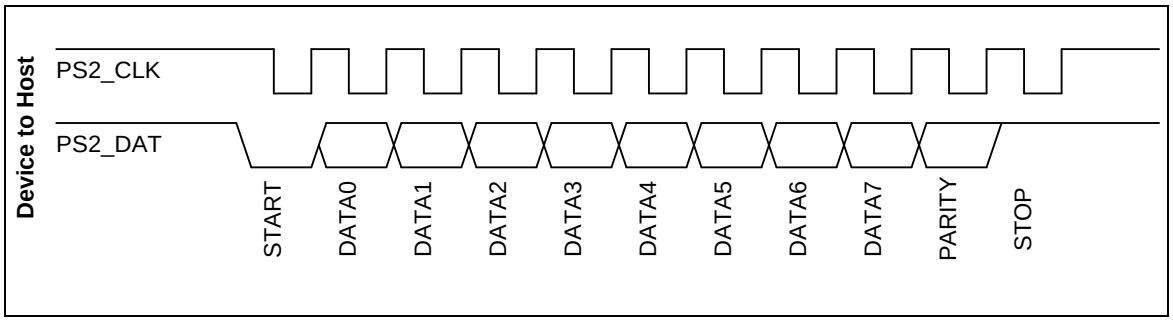


Figure 5-94 Data Format of Device-to-Host

Host-to-Device:

First, the PS/2 device always generates the PS2_CLK signal. If the host wants to send PS2 data, it must first put the PS2_CLK and PS2_DAT lines in a "Request-to-send" state as follows:

- Inhibit communication by pulling PS2_CLK low for at least 100 us
- Apply "Request-to-send" by pulling PS2_DAT low, then release PS2_CLK

The device should check for this state at intervals not to exceed 10 ms. When the device detects this state, it will begin generating PS2_CLK signals and PS2_CLK in eight PS2_DAT bits, one parity bit and one stop bit. The host changes the PS2_DAT line status only when the PS2_CLK line status is low, and PS2_DAT is read by the device when PS2_CLK is high.

After the stop bit is received, the device will acknowledge the received byte by bringing the PS2_DAT line low and generating one last PS2_CLK pulse. If the host does not release the PS2_DAT line after the 11th PS2_CLK pulse, the device will continue to generate PS2_CLK pulses until the PS2_DAT line is released.

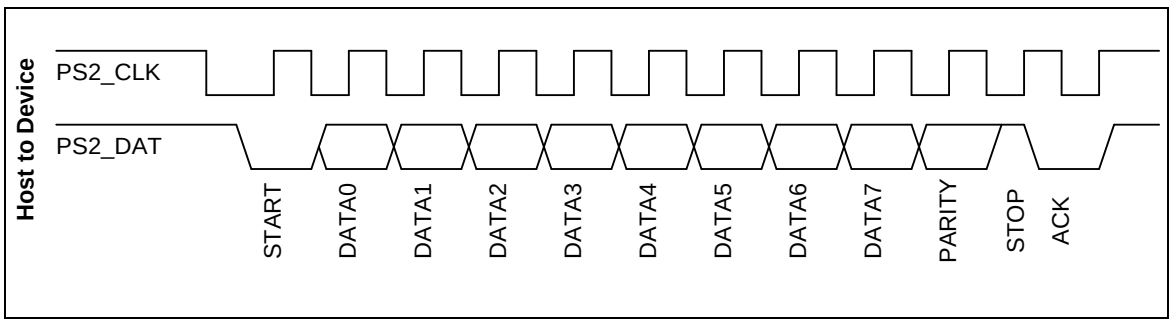


Figure 5-95 Data Format of Host-to-Device

The host and the detailed timing of the DATA and CLK for communication are shown below.

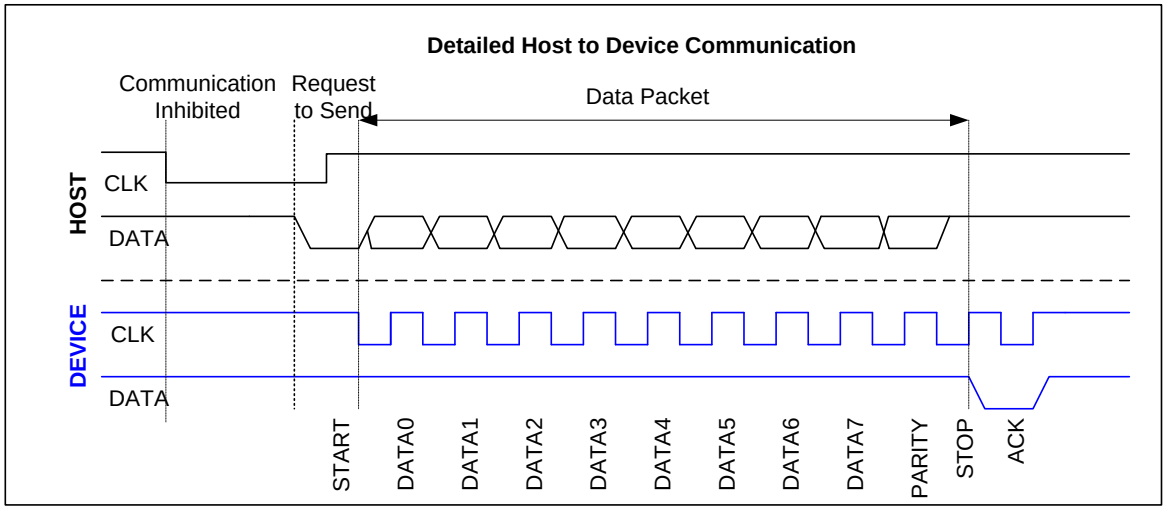


Figure 5-96 PS/2 Bit Data Format

5.14.4.2 *PS/2 Bus Timing Specification*

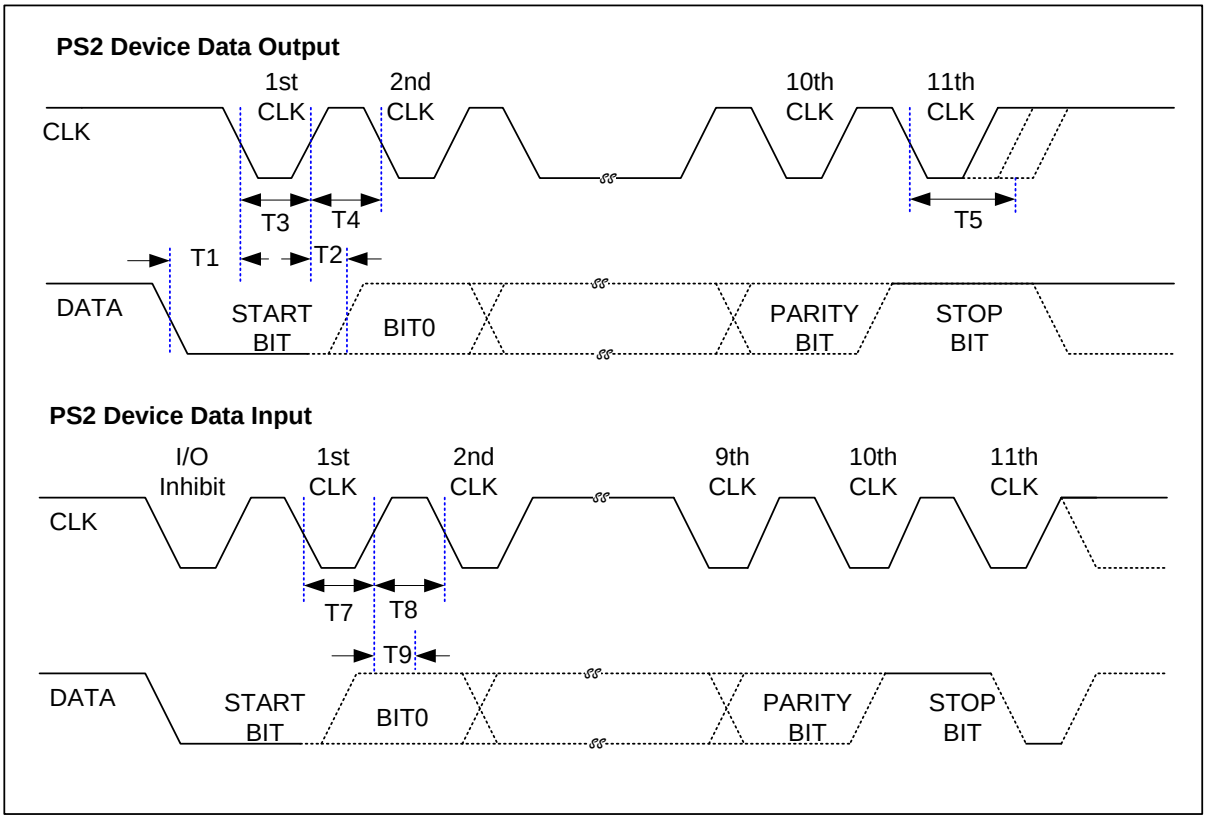


Figure 5-97 PS/2 Bus Timing

Symbol	Timing Parameter	Min.	Max.
T1	PS2_DAT transition to the falling edge of PS2_CLK	5us	25us
T2	Rising edge of PS2_CLK to PS2_DAT transition	5us	T4-5us
T3	Duration of PS2_CLK inactive	30us	50us
T4	Duration of PS2_CLK active	30us	50us
T5	Time to auxiliary device inhibit after 11^{th} clock to ensure auxiliary device does not start another transmission	>0	50us
T7	Duration of PS2_CLK inactive	30us	50us
T8	Duration of PS2_CLK active	30us	50us
T9	Time from inactive to active PS2_CLK transition, used to time auxiliary device sample PS2_DAT	5us	25us

5.14.4.3 *TX FIFO Operation*

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

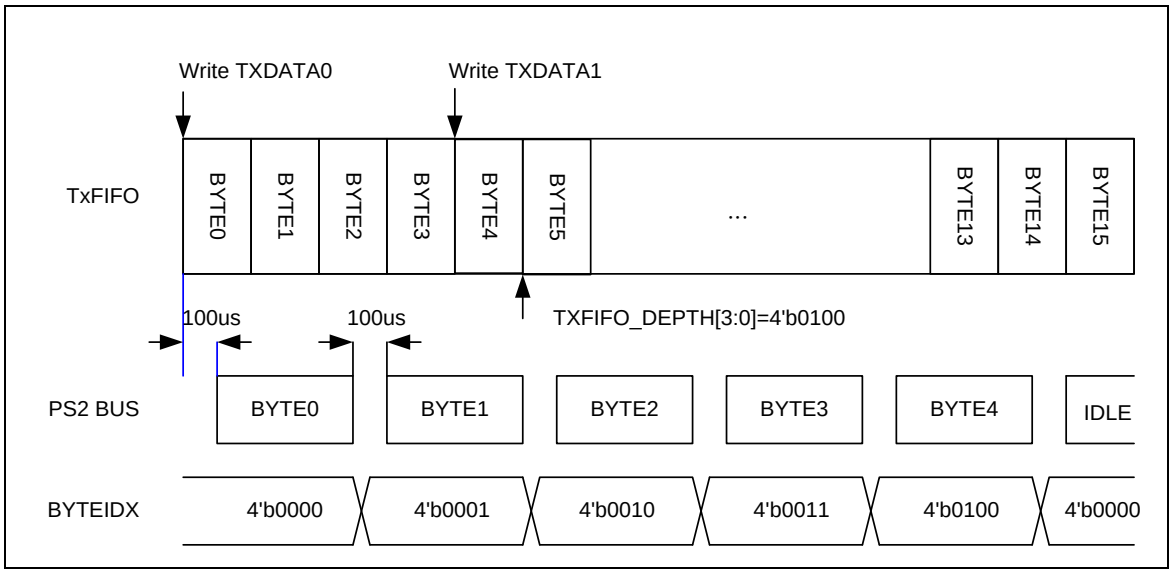


Figure 5-98 PS/2 Data Format

5.15 I²S Controller (I²S)

5.15.1 Overview

The I²S controller consists of I²S protocol to interface with external audio CODEC. Two 8-word deep FIFO for read path and write path respectively and is capable of handling 8-, 16-, 24- and 32-bit word sizes. PDMA controller handles the data movement between FIFO and memory.

5.15.2 Features

- Operated as either Master or Slave
- Capable of handling 8-, 16-, 24- and 32-bit word sizes
- Supports Mono and stereo audio data
- Supports I²S and MSB justified data format
- Provides two 8-word FIFO data buffers, one for transmitting and the other for receiving
- Generates interrupt requests when buffer levels cross a programmable boundary
- Two PDMA requests, one for transmitting and the other for receiving

5.15.3 Block Diagram

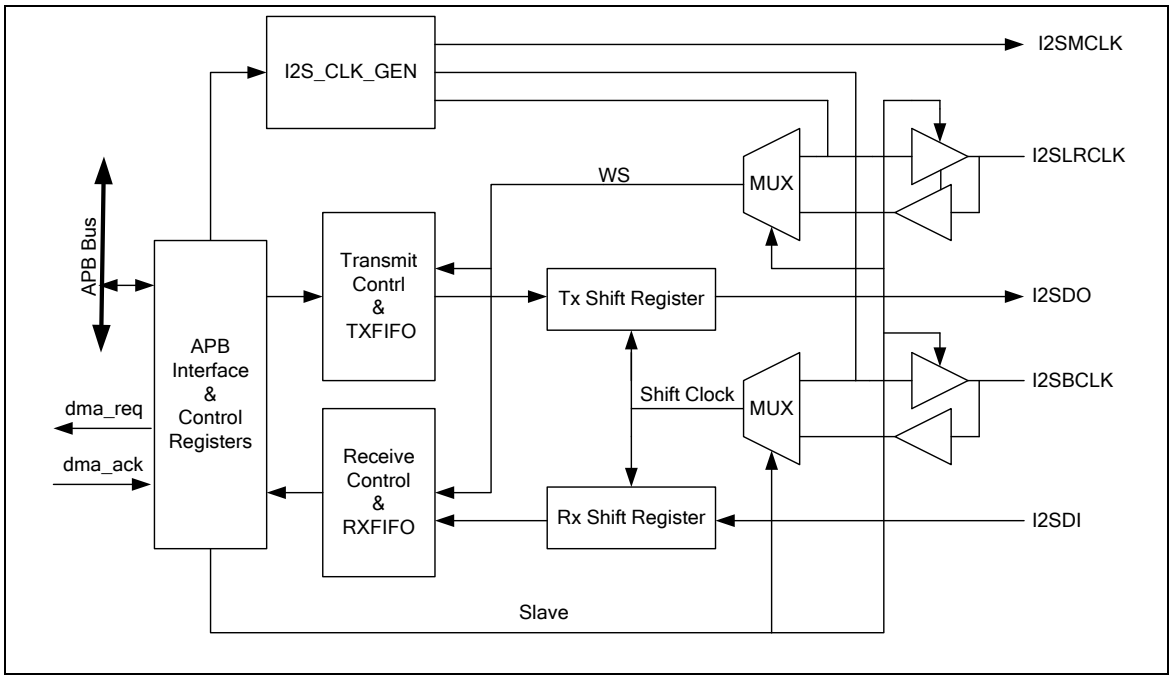


Figure 5-99 I²S Controller Block Diagram

5.15.4 Functional Description

5.15.4.1 *I²S Clock*

The I²S controller has four clock sources selected by I2S_S(CLKSEL2[1:0]). The I²S clock rate must be slower than or equal to system clock rate.

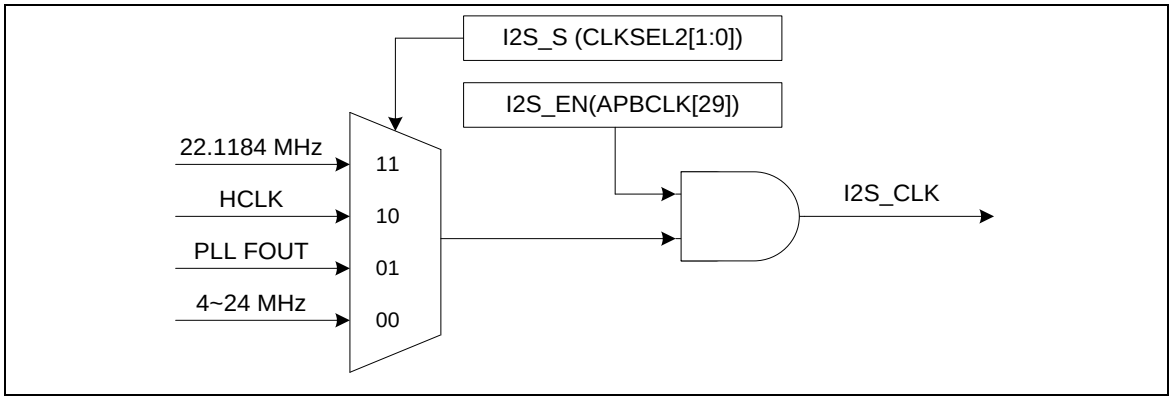


Figure 5-100 I²S Clock Control Diagram

5.15.4.2 *I²S Operation*

The I²S controller supports MSB justified and I²S data format. The I2SLRCLK signal indicates which audio channel is in transferring. The bit count of an audio channel is determined by WORDWIDTH setting. The transfer sequence is always first from the most significance bit, MSB. Data are read on rising clock edge and are driven on falling clock edge.

In I²S data format, the MSB is sent and latched on the second clock of an audio channel.

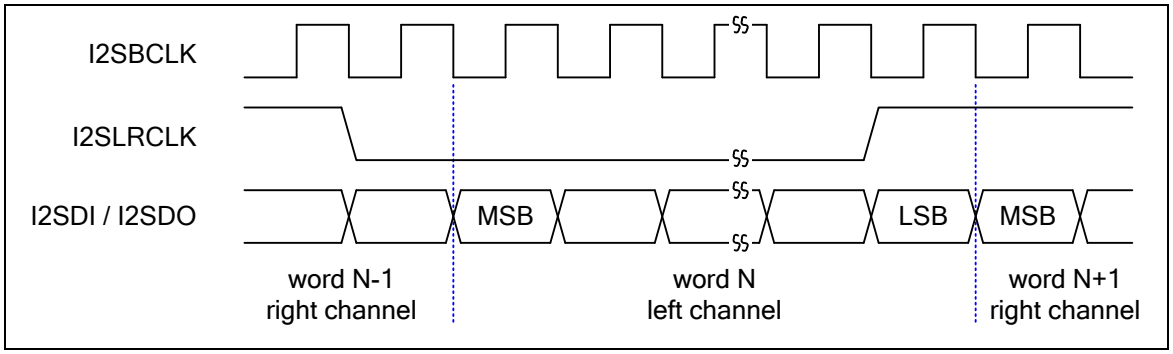


Figure 5-101 I²S Data Format Timing Diagram

In MSB justified data format, the MSB is sent and latched on the first clock of an audio channel.

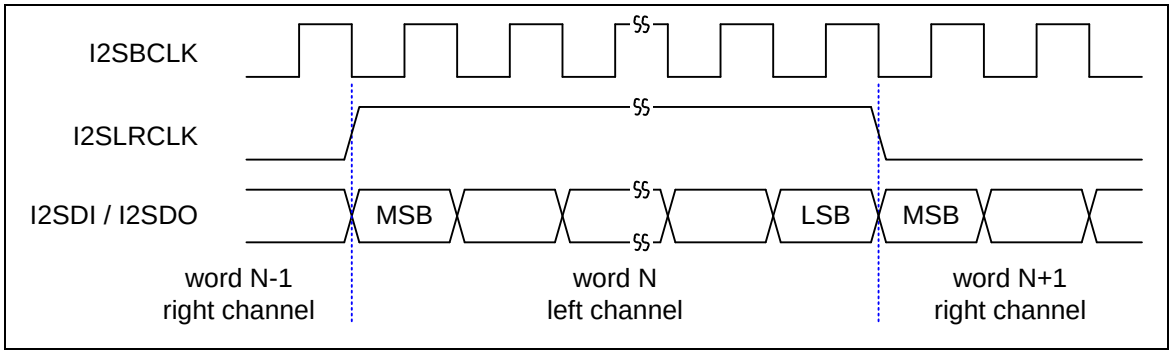


Figure 5-102 MSB Justified Data Format Timing Diagram

5.15.4.3 *I²S Interrupt sources*

The I²S controller supports left channel zero-cross interrupt, right channel zero-cross interrupt, transmit FIFO threshold level interrupt, transmit FIFO overflow interrupt and transmit FIFO underflow interrupt in transmit operation. In receive operation, it supports receive FIFO threshold level interrupt, receive FIFO overflow interrupt and receive FIFO underflow interrupt. When I²S interrupt occurs, user can check I2STXINT and I2SRXINT flags to recognize the interrupt sources.

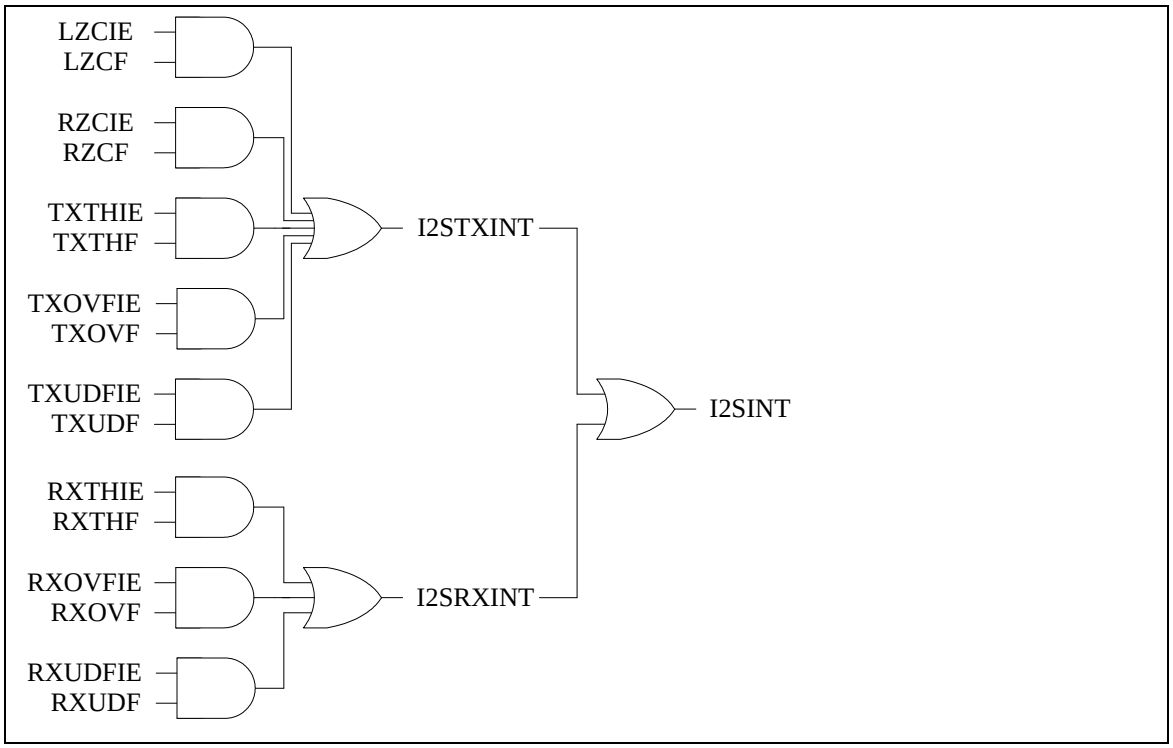


Figure 5-103 I²S Interrupts

5.15.4.4 *FIFO operation*

The word width of an audio channel can be 8, 16, 24 or 32 bits. The memory arrangements for various settings are shown below.

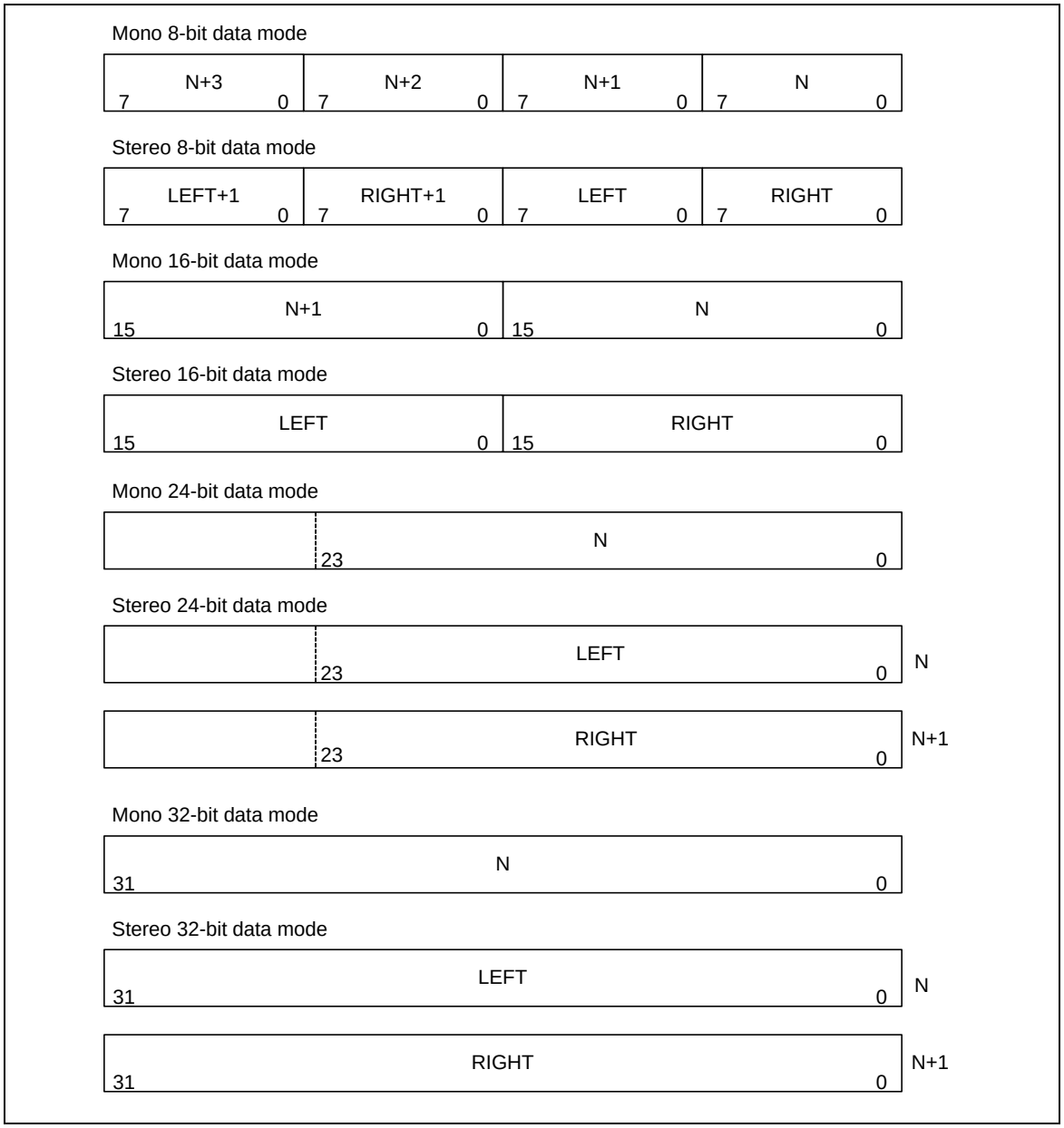


Figure 5-104 FIFO Contents for Various I²S Modes

5.16 Analog-to-Digital Converter (ADC)

5.16.1 Overview

The NuMicro™ NUC200 Series contains one 12-bit successive approximation analog-to-digital converters (SAR A/D converter) with 8 input channels. The A/D converter supports three operation modes: single, single-cycle scan and continuous scan mode. The A/D converter can be started by software, PWM Center-aligned trigger and external STADC pin.

5.16.2 Features

- Analog input voltage range: 0~V_{REF}
- 12-bit resolution and 10-bit accuracy is guaranteed
- Up to 8 single-end analog input channels or 4 differential analog input channels
- Up to 760 kSPS conversion rate as ADC clock frequency is 16 MHz (chip working at 5V)
- Three operating modes
  - Single mode: A/D conversion is performed one time on a specified channel
  - Single-cycle scan mode: A/D conversion is performed one cycle on all specified channels with the sequence from the smallest numbered channel to the largest numbered channel
  - Continuous scan mode: A/D converter continuously performs Single-cycle scan mode until software stops A/D conversion
- An A/D conversion can be started by:
  - Writing 1 to ADST bit through software
  - PWM Center-aligned trigger
  - External pin STADC
- Conversion results are held in data registers for each channel with valid and overrun indicators
- Conversion result can be compared with specify value and user can select whether to generate an interrupt when conversion result matches the compare register setting
- Channel 7 supports 3 input sources: external analog voltage, internal Band-gap voltage, and internal temperature sensor output

5.16.3 Block Diagram

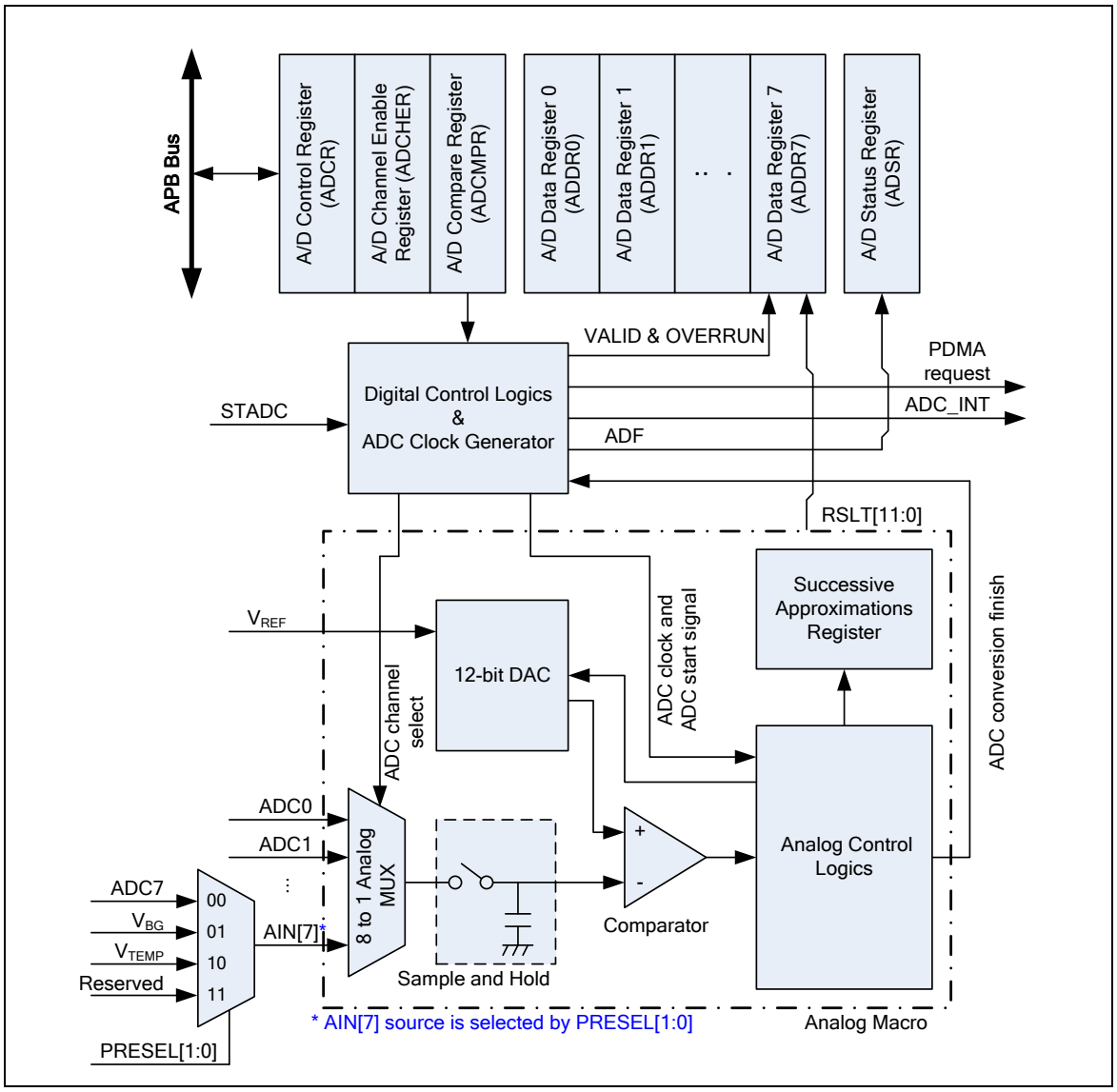


Figure 5-105 ADC Controller Block Diagram

5.16.4 Functional Description

The A/D converter operates by successive approximation with 12-bit resolution. The ADC has three operation modes: Single mode, Single-cycle Scan mode and Continuous Scan mode. When changing the operating mode or analog input channel, to prevent incorrect operation, software must clear ADST bit to 0 in ADCR register.

5.16.4.1 *ADC Clock Generator*

The maximum sampling rate is up to 760 kSPS. The ADC engine has four clock sources selected by 2-bit ADC_S (CLKSEL1[3:2]), the ADC clock frequency is divided by an 8-bit prescaler with the formula:

The ADC clock frequency = (ADC clock source frequency) / (ADC_N+1);

where the 8-bit ADC_N is located in register CLKDIV[23:16].

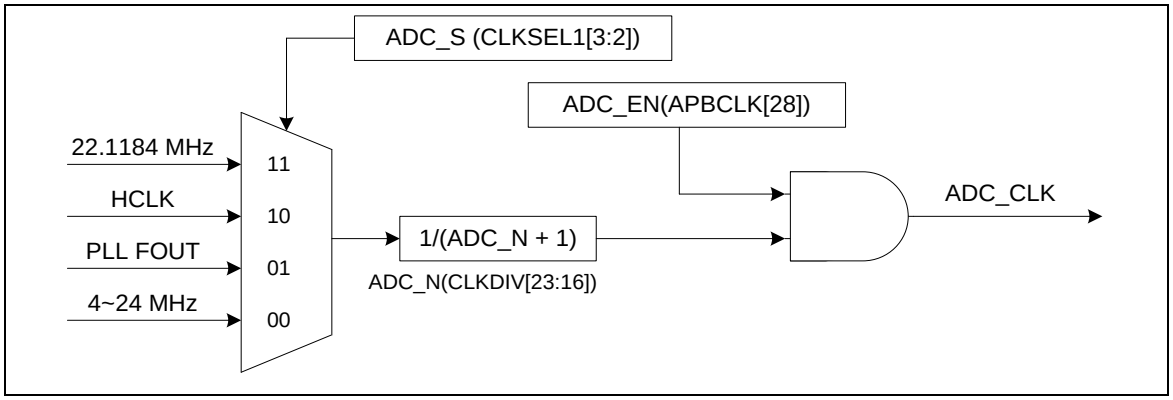


Figure 5-106 ADC Clock Control

5.16.4.2 *Single Mode*

In single mode, A/D conversion is performed only once on the specified single channel. The operations are as follows:

1. A/D conversion will be started when the ADST bit of ADCR is set to 1 by software.
2. When A/D conversion is finished, the result is stored in the A/D data register corresponding to the channel.
3. The ADF bit of ADSR register will be set to 1. If the ADIE bit of ADCR register is set to 1, the ADC interrupt will be asserted.
4. The ADST bit remains 1 during A/D conversion. When A/D conversion ends, the ADST bit is automatically cleared to 0 and the A/D converter enters idle state.

Note: If software enables more than one channel in single mode, the channel with the smallest number will be selected and the other enabled channels will be ignored.

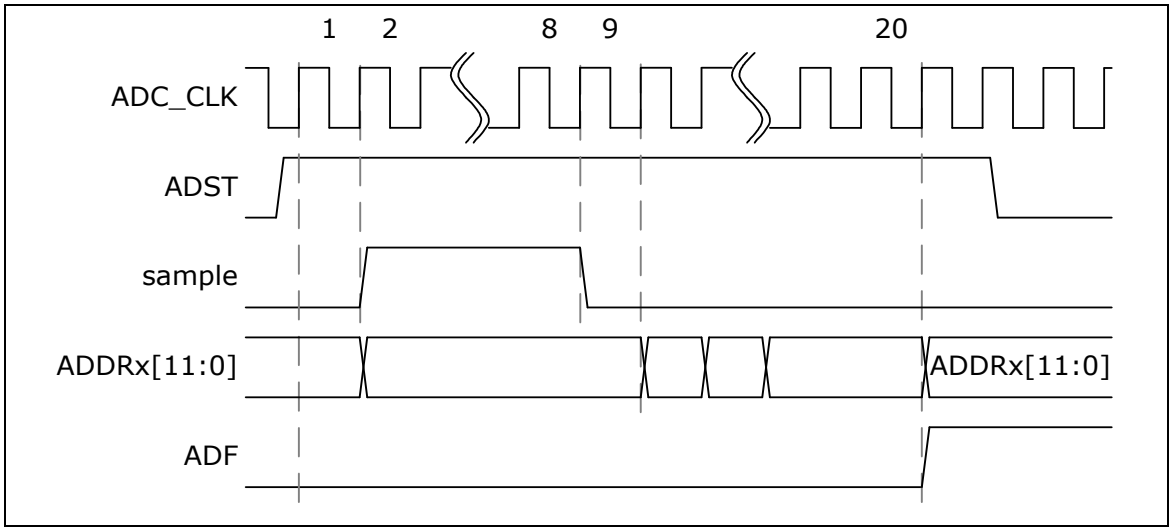


Figure 5-107 Single Mode Conversion Timing Diagram

5.16.4.3 *Single-Cycle Scan Mode*

In single-cycle scan mode, A/D conversion will sample and convert the specified channels once in the sequence from the smallest number enabled channel to the largest number enabled channel.

1. When the ADST bit of ADCR is set to 1 by software or external trigger input, A/D conversion starts on the channel with the smallest number.
2. When A/D conversion for each enabled channel is completed, the result is sequentially transferred to the A/D data register corresponding to each channel.
3. When the conversions of all the enabled channels are completed, the ADF bit in ADSR is set to 1. If the ADC interrupt function is enabled, the ADC interrupt occurs.
4. After A/D conversion ends, the ADST bit is automatically cleared to 0 and the A/D converter enters idle state. If ADST is cleared to 0 before all enabled ADC channels conversion done, ADC controller will finish current conversion and save the result to the ADDRx of the current conversion channel.

An example timing diagram for single-cycle scan on enabled channels (0, 2, 3 and 7) is shown below.

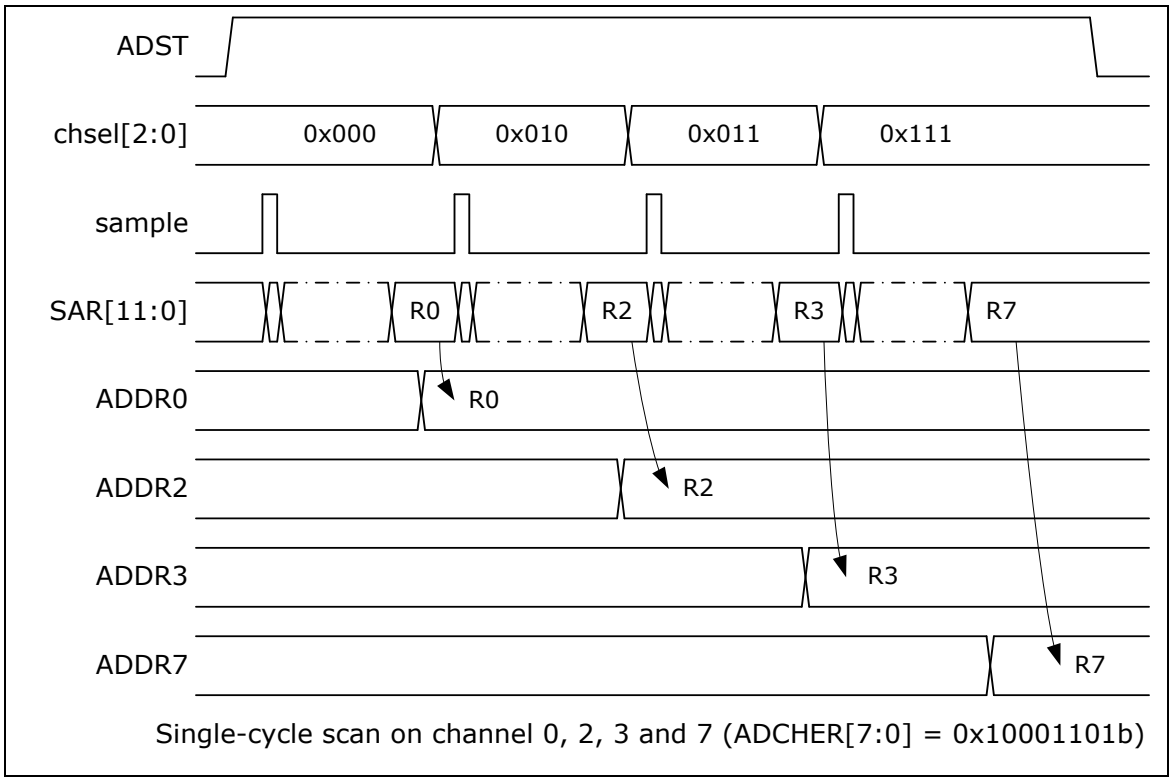


Figure 5-108 Single-Cycle Scan on Enabled Channels Timing Diagram

5.16.4.4 *Continuous Scan Mode*

In continuous scan mode, A/D conversion is performed sequentially on the specified channels that enabled by CHEN bits in ADCHER register (maximum 8 channels for ADC). The operations are as follows:

1. When the ADST bit in ADCR is set to 1 by software, A/D conversion starts on the channel with the smallest number.
2. When A/D conversion for each enabled channel is completed, the result of each enabled channel is stored in the A/D data register corresponding to each enabled channel.
3. When A/D converter completes the conversions of all enabled channels sequentially, the ADF bit (ADSR[0]) will be set to 1. If the ADC interrupt function is enabled, the ADC interrupt occurs. The conversion of the enabled channel with the smallest number will start again if software has not cleared the ADST bit.
4. As long as the ADST bit remains at 1, the step 2 ~ 3 will be repeated. When ADST is cleared to 0, ADC controller will stop conversion.

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

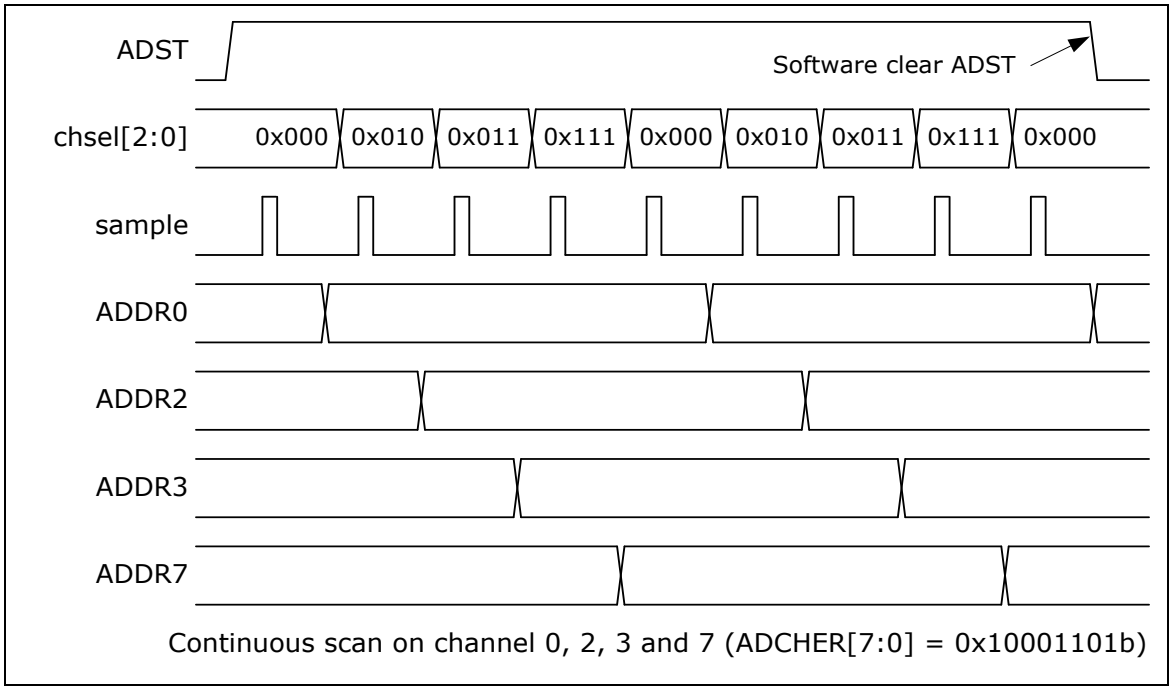


Figure 5-109 Continuous Scan on Enabled Channels Timing Diagram

5.16.4.5 *External trigger Input Sampling and A/D Conversion Time*

In single-cycle scan mode, A/D conversion can be triggered by external pin request. When the ADCR.TRGEN is set to high to enable ADC external trigger function, setting the TRGS[1:0] bits to 00b is to select external trigger input from the STADC pin. Software can set TRGCOND[1:0] to select trigger condition is falling/rising edge or low/high level. If level trigger condition is selected, the STADC pin must be kept at defined state at least 8 PCLKs. The ADST bit will be set to 1 at the 9th PCLK and start to conversion. Conversion is continuous if external trigger input is kept at active state in level trigger mode. It is stopped only when external condition trigger condition disappears. If edge trigger condition is selected, the high and low state must be kept at least 4 PLCKs. Pulse that is shorter than this specification will be ignored.

5.16.4.6 *PWM Center-aligned trigger*

In single-cycle scan mode, the PWM can be the trigger source of ADC by setting the TRGEN(ADCR[8]) to 1 and the TRGS(ADCR[5:4]) to 11b.

When PWM enables trigger ADC function, the PWM will generate a trigger signal to ADC when PWM counter is running to PWM center point.

5.16.4.7 *Conversion Result Monitor by Compare Function*

ADC controller provide two sets of compare register ADCMPR0 and ADCMPR1, to monitor maximum two specified channels conversion result from A/D conversion controller, refer to Figure 5-110. Software can select which channel to be monitored by set CMPCH(ADCMPRx[5:0]) and CMPCOND bit is used to check conversion result is less than specify value or greater than (equal to) value specified in CMPD[11:0]. When the conversion of the channel specified by CMPCH is completed, the comparing action will be triggered one time automatically. When the compare result meets the setting, compare match counter will increase 1, otherwise, the compare match counter will be cleared to 0. When counter value reach the setting of (CMPMATCNT+1) then CMPF bit will be set to 1, if CMPIE bit is set then an ADC_INT interrupt request is generated. Software can use it to monitor the external analog input pin voltage transition in scan mode without imposing a load on software. Detailed logics diagram is shown below.

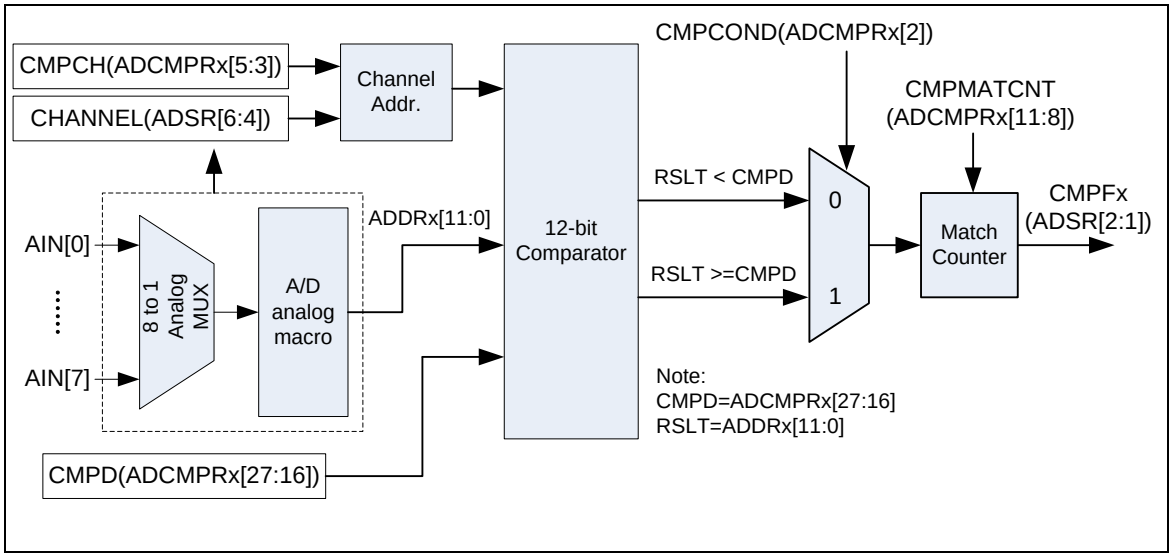


Figure 5-110 A/D Conversion Result Monitor Logics Diagram

5.16.4.8 *Interrupt Sources*

There are three interrupt sources of ADC interrupt. When an ADC operation mode finishes its conversion, the A/D conversion end flag, ADF, will be set to 1. The CMPF0 and CMPF1 are the compare flags of compare function. When the conversion result meets the settings of ADCMPR0/1, the corresponding flag will be set to 1. When one of the flags, ADF, CMPF0 and CMPF1, is set to 1 and the corresponding interrupt enable bit, ADIE of ADCR and CMPIE of ADCMPR0/1, is set to 1, the ADC interrupt will be asserted. Software can clear the flag to revoke the interrupt request.

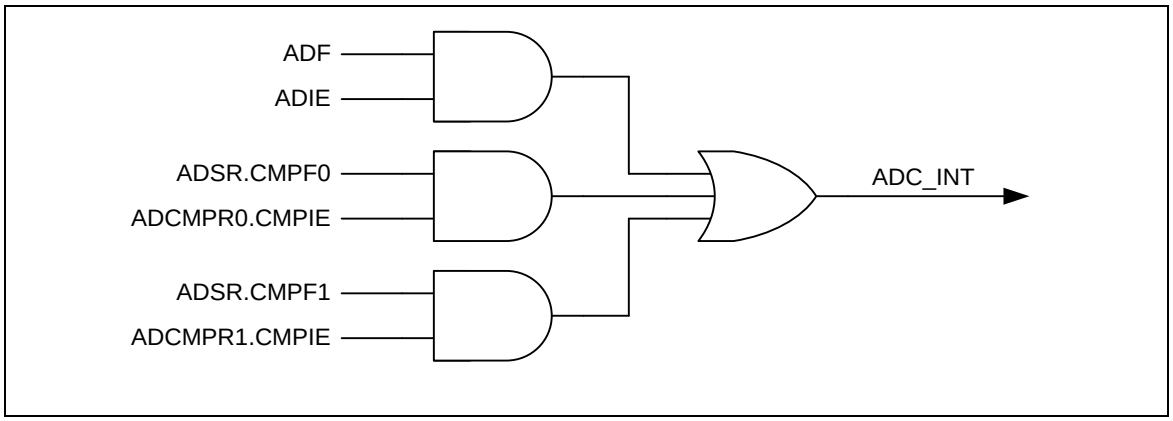


Figure 5-111 A/D Controller Interrupt

5.16.4.9 *Peripheral DMA Request*

When A/D conversion is finished, the conversion result will be loaded into ADDR register and VALID bit will be set to 1. If the PTEN bit of ADCR is set, ADC controller will generate a request to PDMA. User can use PDMA to transfer the conversion results to a user-specified memory space without CPU's intervention. The source address of PDMA operation is fixed at ADPDMA, no matter what channels was selected. When PDMA is transferring the conversion result, ADC will continue converting the next selected channel if the operation mode of ADC is single scan mode or continuous scan mode. User can monitor current PDMA transfer data through reading ADPDMA register. If ADC completes the conversion of a selected channel and the last conversion result of the same channel has not been transferred by PDMA, OVERRUN bit of the corresponding channel will be set and the last ADC conversion result will be overwritten by the new ADC conversion result. PDMA will transfer the latest data of selected channels to the user-specified destination address.

5.17 Analog Comparator (ACMP)

5.17.1 Overview

The NuMicro™ NUC200 Series contains two comparators which can be used in a number of different configurations. 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 5-112.

5.17.2 Features

- Analog input voltage range: 0~ V_{DDA}
- Supports Hysteresis function
- Supports optional internal reference voltage input at negative end for each comparator

5.17.3 Block Diagram

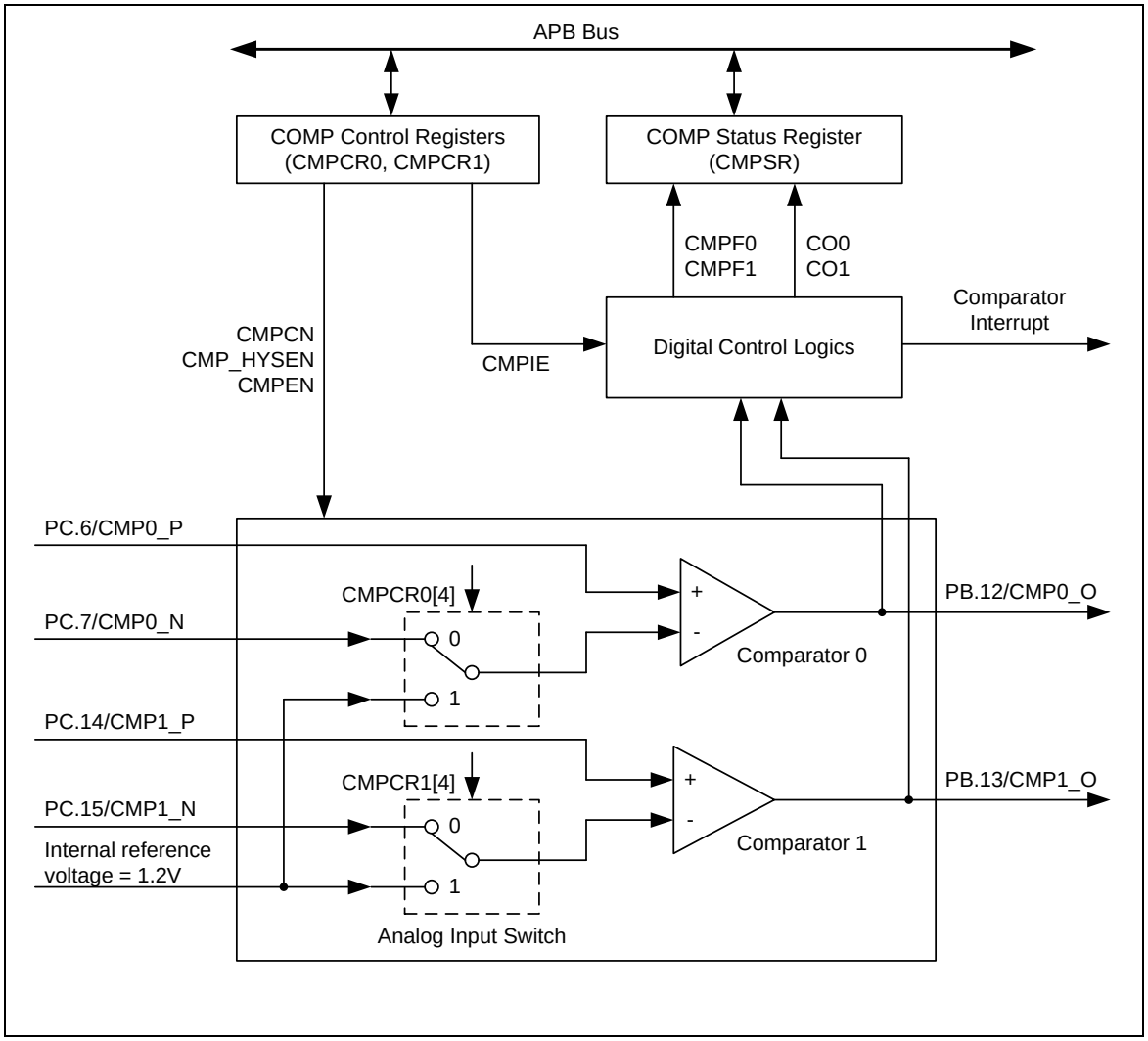


Figure 5-112 Analog Comparator Block Diagram

5.17.4 Functional Description

5.17.4.1 *Interrupt Sources*

The output of comparators are sampled by PCLK and reflected at CO1 and CO2 of CMPSR register. If CMP0IE/CMP1IE of CMP0CR/CMP1CR is set to 1, the comparator interrupt will be enabled. As the output state of comparator is changed, the comparator interrupt will be asserted and the corresponding flag, CMPF0 or CMPF1, will be set. Software can clear the flag to 0 by writing 1 to it.

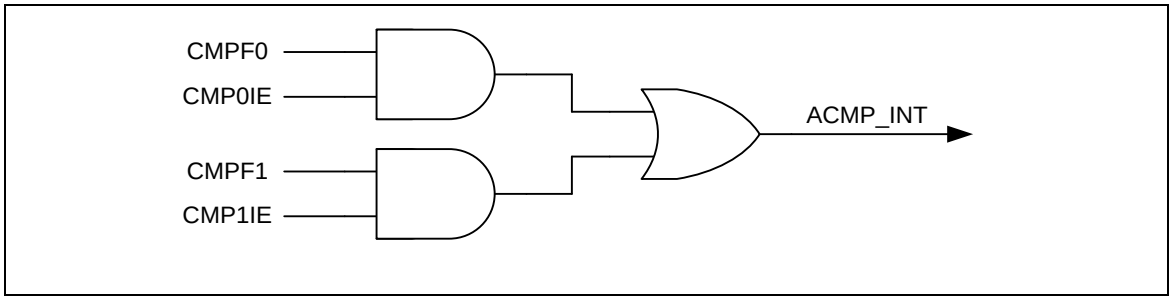


Figure 5-113 Comparator Controller Interrupt Sources

5.17.4.2 *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.

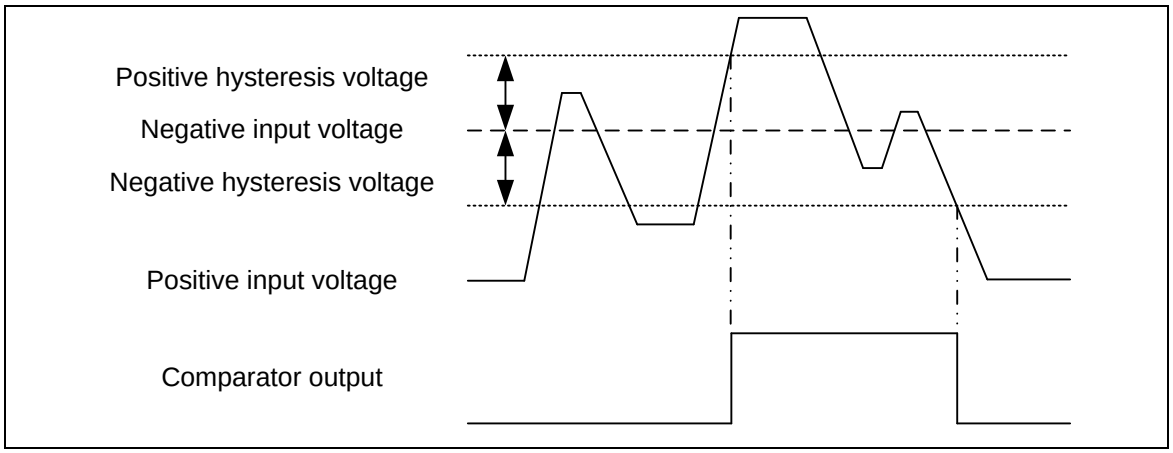


Figure 5-114 Comparator Hysteresis Function

5.18 PDMA Controller (PDMA)

5.18.1 Overview

The NuMicro™ NUC200 series DMA contains nine-channel peripheral direct memory access (PDMA) controller and a cyclic redundancy check (CRC) generator.

The PDMA that transfers data to and from memory or transfer data to and from APB devices. For PDMA channel (PDMA CH0~CH8), there is one-word buffer as transfer buffer between the Peripherals APB devices and Memory. Software can stop the PDMA operation by disable PDMA PDMA_CSRx[PDMACEN]. The CPU can recognize the completion of a PDMA operation by software polling or when it receives an internal PDMA interrupt. The PDMA controller can increase source or destination address or fixed them as well.

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

5.18.2 Features

- Supports nine PDMA channels and one CRC channel. Each PDMA channel can support a unidirectional transfer
- AMBA AHB master/slave interface compatible, for data transfer and register read/write
- Hardware round robin priority scheme. DMA channel 0 has the highest priority and channel 8 has the lowest priority
- PDMA operation
  - □ Peripheral-to-memory, memory-to-peripheral, and memory-to-memory transfer
  - □ Supports word/half-word/byte transfer data width from/to peripheral
  - □ Supports address direction: increment, fixed.
- Cyclic Redundancy Check (CRC)
  - □ Supports four common polynomials CRC-CCITT, CRC-8, CRC-16, and CRC-32
    - CRC-CCITT: $X^{16} + X^{12} + X^5 + 1$
    - CRC-8: $X^8 + X^2 + X + 1$
    - CRC-16: $X^{16} + X^{15} + X^2 + 1$
    - CRC-32: $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$
  - □ Supports programmable CRC seed value.
  - □ Supports programmable order reverse setting for input data and CRC checksum.
  - □ Supports programmable 1's complement setting for input data and CRC checksum.
  - □ Supports CPU PIO mode or DMA transfer mode.
  - □ Supports the follows write data length in CPU PIO mode
    - 8-bit write mode (byte): 1-AHB clock cycle operation.
    - 16-bit write mode (half-word): 2-AHB clock cycle operation.
  - □ - 32-bit write mode (word): 4-AHB clock cycle operation.
  - □ Supports byte alignment transfer data length and word alignment transfer source address in CRC DMA mode.

5.18.3 Block Diagram

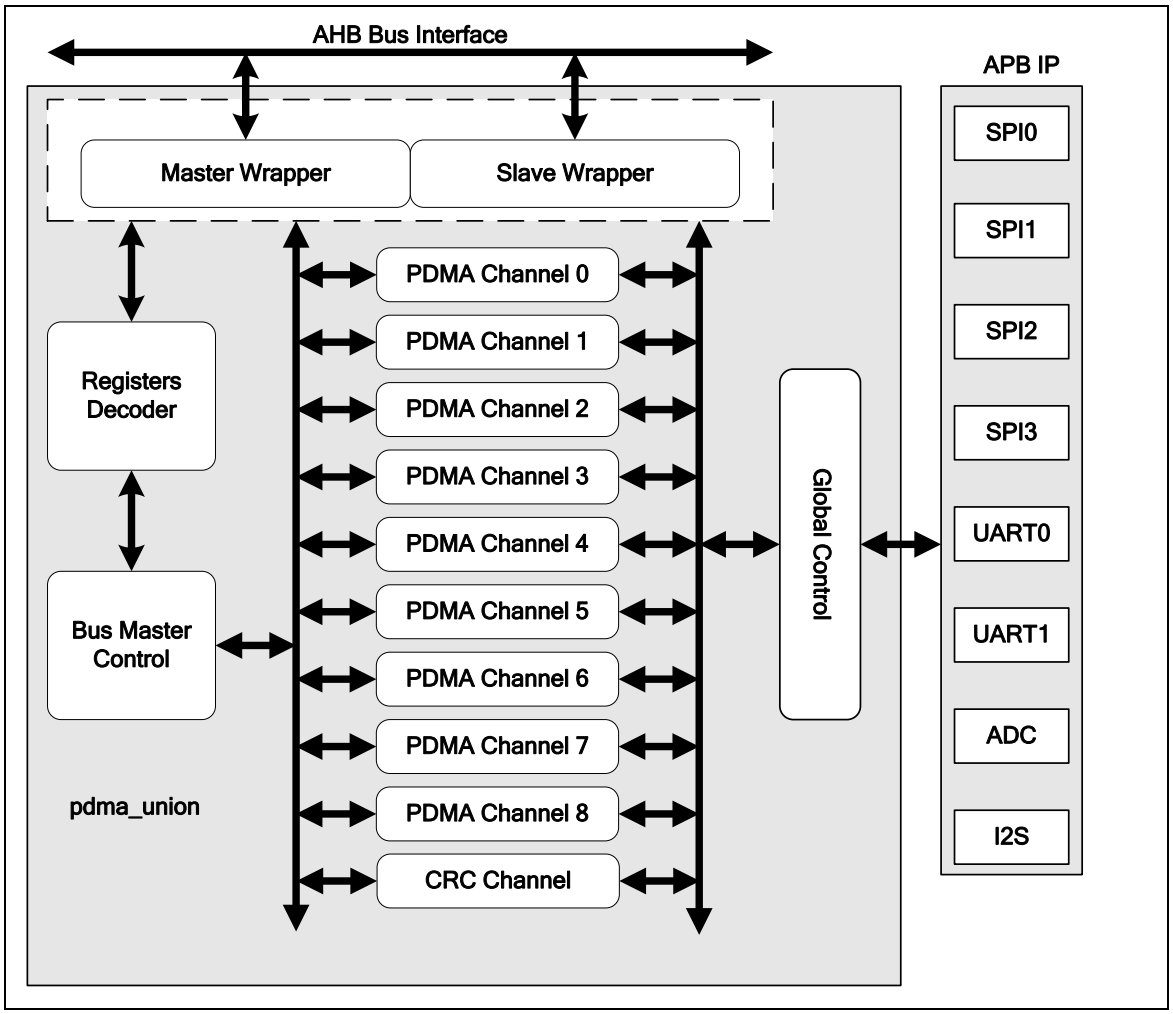


Figure 5-115 DMA Controller Block Diagram

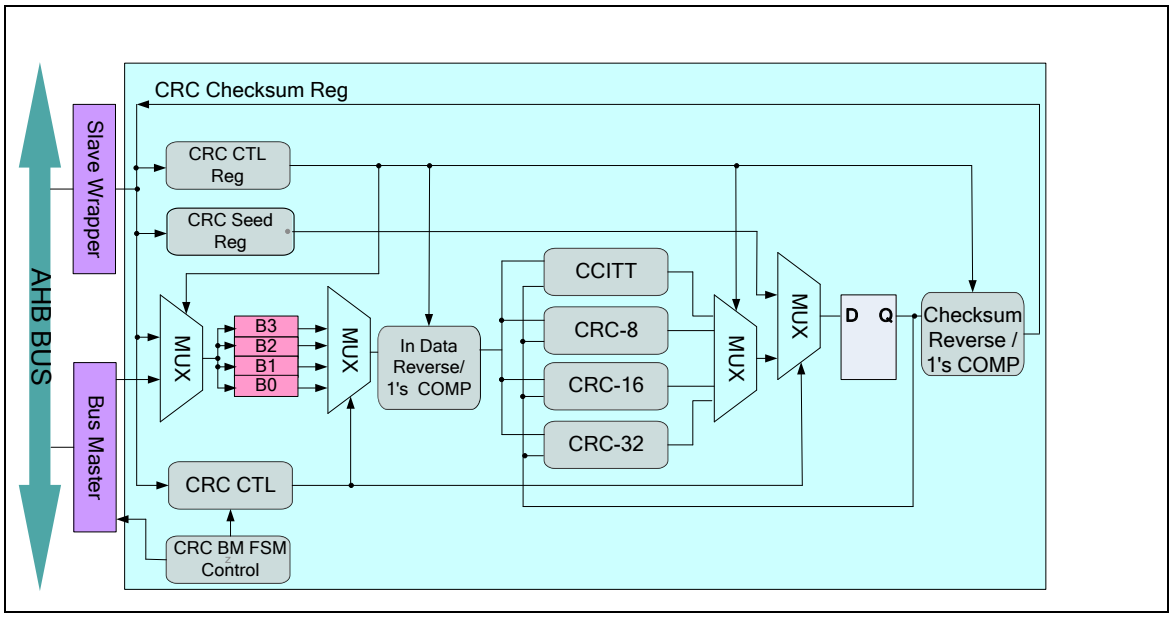


Figure 5-116 CRC Generator Block Diagram

5.18.4 Functional Description

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

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

- **PDMA**

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

Every PDMA channel behavior is not pre-defined, users must configure the channel service settings of PDMA_PDSSR0, PDMA_PDSSR1 and PDMA_PDSSR2 registers before starting the related PDMA channel.

Software must enable PDMA channel by setting PDMA_SCRx[PDMACEN] bit and then write a valid source address to the PDMA_SARx register, a destination address to the PDMA_DARx register, and a transfer count to the PDMA_BCRx register. Next, trigger the PDMA_CSRx [TRIG_EN]. PDMA will continue the transfer until PDMA_CBCRx counts down to 0. The following sequence is a program sequence example.

- Enable PDMA engine clock by setting PDMA_EN bit in AHBCLK register.
- Configure the channel service setting by setting PDMA_PDSSR0/ PDMA_PDSSR1/ PDMA_PDSSR2 register.
- Configure PDMA_CSRx register:
  - Enable PDMA channel(PDMACE)
  - Set source/destination address direction(SAD_SEL / DAD_SEL)
  - Configure PDMA mode selection(MODE_SEL)
  - Configure peripherial transfer width selection(APB_TWS).
- Configure source /destination address by setting PDMA_SARx/PDMA_DARx registers.
- Configure PDMA_transfer byte count by setting PDMA_BCRx register.
- Enable PDMA block transfer done interrupt by setting BLKD_IE[PDMA_IERx]. (optional)
- Enable PDMA NVIC by setting NVIC_ISER register bit 26 to “1”. (optional)
- Enable PDMA read/write transfer by setting TRIG_EN bit in PDMA_CSRx register.
- If PDMA block transfer done interrupt is generated, write “1” to PDMA_ISRx[BKLD_IF] by software to clear interrupt flag.
- Enable PDMA read/write transfer by setting the TRIG_EN bit in PDMA_CSRx register for the next block transfer.

If an error occurs during the PDMA operation, the channel stops unless software clears the error condition and sets the PDMA_CSRx [SW_RST] to reset the PDMA channel and set PDMA_CSRx [PDMACEN] and [TRIG_EN] bits field to start again.

In PDMA (Peripheral-to-Memory or Memory-to-Peripheral) mode, DMA can transfer data between the Peripherals APB IP (e.g. UART, SPI, ADC) and Memory.

- **CRC**

The DMA controller contains a cyclic redundancy check (CRC) generator that can perform CRC calculation with programmable polynomial settings. The operation polynomial includes CRC-CCITT, CRC-8, CRC-16 and CRC-32; Software can choose the CRC operation polynomial mode by setting CRC_MODE[1:0] (CRC_CTL[31:30] CRC Polynomial Mode).

The CRC engine supports CPU PIO mode if CRCCEN bit (CRC_CLT [0] CRC Channel Enable) is 1, TRIG_EN bit (CRC_CTL [23] CRC DMA Trigger Enable) is 0 and DMA transfer mode if CRCCEN bit (CRC_CLT [0] CRC Channel Enable) is 1, TRIG_EN bit (CRC_CTL [23] CRC DMA Trigger Enable) is 1. The following sequence is a program sequence example.

Procedure when operating in CPU PIO mode:

- Enable CRC engine by setting CRCCEN bit (CRC_CLT [0] CRC Channel Enable) to 1.
- Set the transfer data format WDATA_RVS (CRC_CTL [24] Write Data Order Reverse), CHECKSUM_RVS (CRC_CTL [25] Checksum Reverse), WDATA_COM (CRC_CTL [26] Write Data 1's Complement), CHECKSUM_COM (CRC_CTL [27] Checksum 1's Complement), initial seed value in CRC_SEED (CRC_SEED [31:0] CRC Seed Register) and select write data length by setting CPU_WDLEN [1:0] (CRC_CTL [29:28] CPU Write Data Length).
- Set the CRC_RST bit (CRC_CTL [1] CRC Engine Reset) to 1 to load the initial seed value to CRC circuit but others contents of CRT_CTL register will not be cleared. This bit will be cleared automatically.
- Write data to CRC_WDATA (CRC_WDATA [31:0] CRC Write Data Register) to perform CRC calculation.
- Then, get the CRC checksum results by reading the CRC_CHECKSUM (CRC_CHECKSUM [31:0] CRC Checksum Register).

Procedure when operating in CRC DMA mode:

- Enable CRC engine by setting CRCCEN bit (CRC_CLT [0] CRC Channel Enable) to 1.
- Set the transfer data format WDATA_RVS (CRC_CTL [24] Write Data Order Reverse), CHECKSUM_RVS (CRC_CTL [25] Checksum Reverse), WDATA_COM (CRC_CTL [26] Write Data 1's Complement), CHECKSUM_COM (CRC_CTL [27] Checksum 1's Complement) and initial seed value in CRC_SEED (CRC_SEED [31:0] CRC Seed Register).
- Specify a valid source address (word alignment) and transfer counts by setting CRC_DMASAR (CRC_DMASAR[31:0] CRC DMA Transfer Source Address Register) and CRC_DMABCR (CRC_DMABCR [15:0] CRC DMA Transfer Byte Count Register).
- Set TRIG_EN bit (CRC_CTL [23] CRC DMA Trigger Enable) to 1 to perform CRC calculation.
- Wait CRC DMA transfer and check if CRC DMA transfer is done by the CRC_BLKD_IF bit (CRC DMA Block Transfer Done Interrupt Flag), and then get the CRC checksum results by reading the CRC_CHECKSUM (CRC_CHECKSUM [31:0] CRC Checksum Register).

5.19 Smart Card Host Interface (SC)

5.19.1 Overview

The Smart Card Interface controller (SC controller) is based on ISO/IEC 7816-3 standard and fully compliant with PC/SC Specifications. It also provides status of card insertion/removal.

5.19.2 Features

- ISO7816-3 T=0, T=1 compliant
- EMV2000 compliant
- Supports up to three ISO7816-3 ports
- Separates receive/ transmit 4 byte entry buffer for data payloads
- Programmable transmission clock frequency
- Programmable receiver buffer trigger level
- Programmable guard time selection (11 ETU ~ 266 ETU)
- One 24-bit and two 8-bit time-out counters for Answer to Request (ATR) and waiting times processing
- Supports auto inverse convention function
- Supports transmitter and receiver error retry and error retry number limitation function
- Supports hardware activation sequence process
- Supports hardware warm reset sequence process
- Supports hardware deactivation sequence process
- Supports hardware auto deactivation sequence when detecting the card removal

5.19.3 Block Diagram

The SC clock control and block diagram are shown as follows. The SC controller is completely asynchronous design with two clock domains, PCLK and engine clock. Note that PCLK frequency should be higher than the frequency of engine clock.

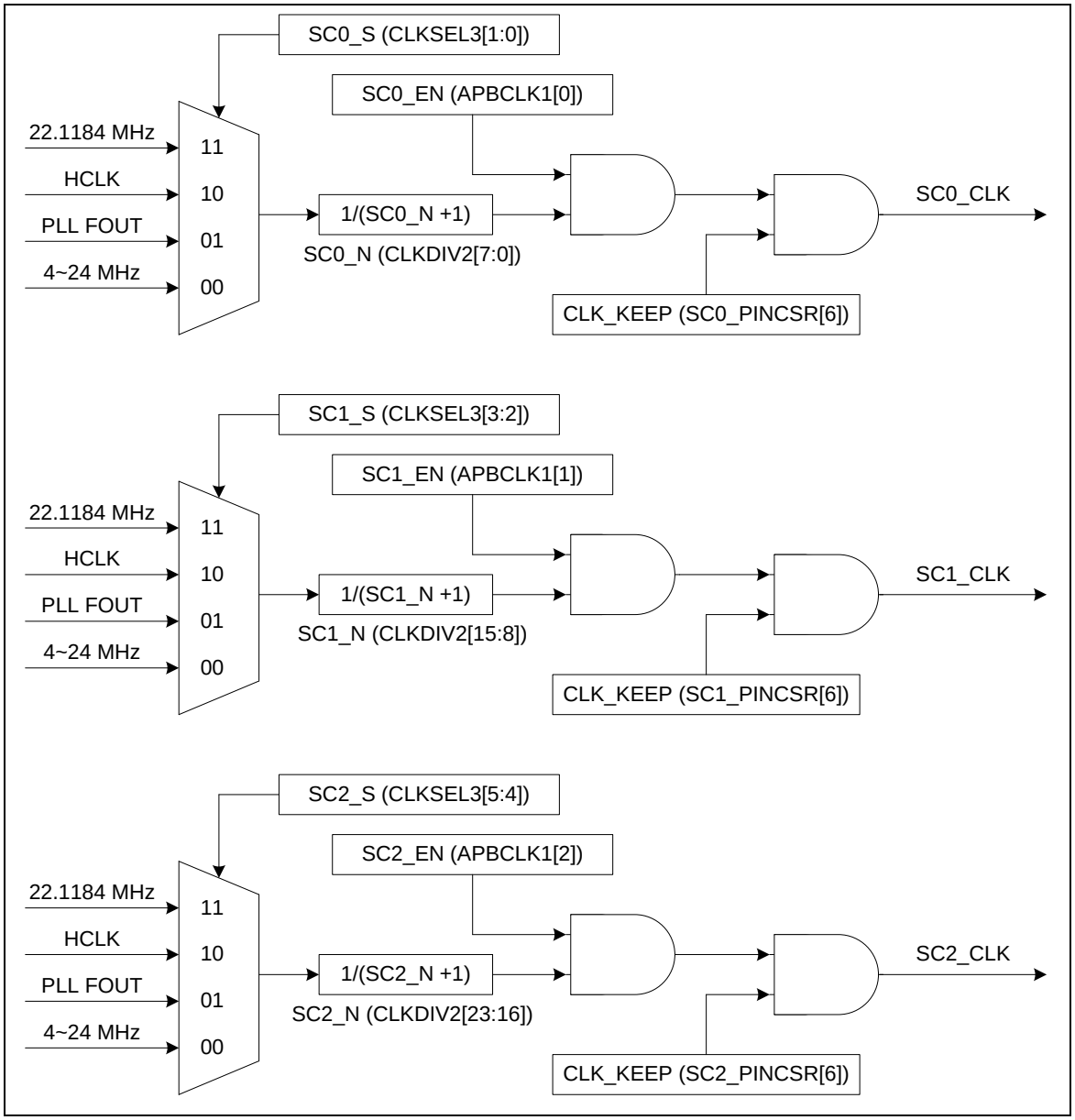


Figure 5-117 SC Clock Control Diagram (8-bit Prescale Counter in Clock Controller)

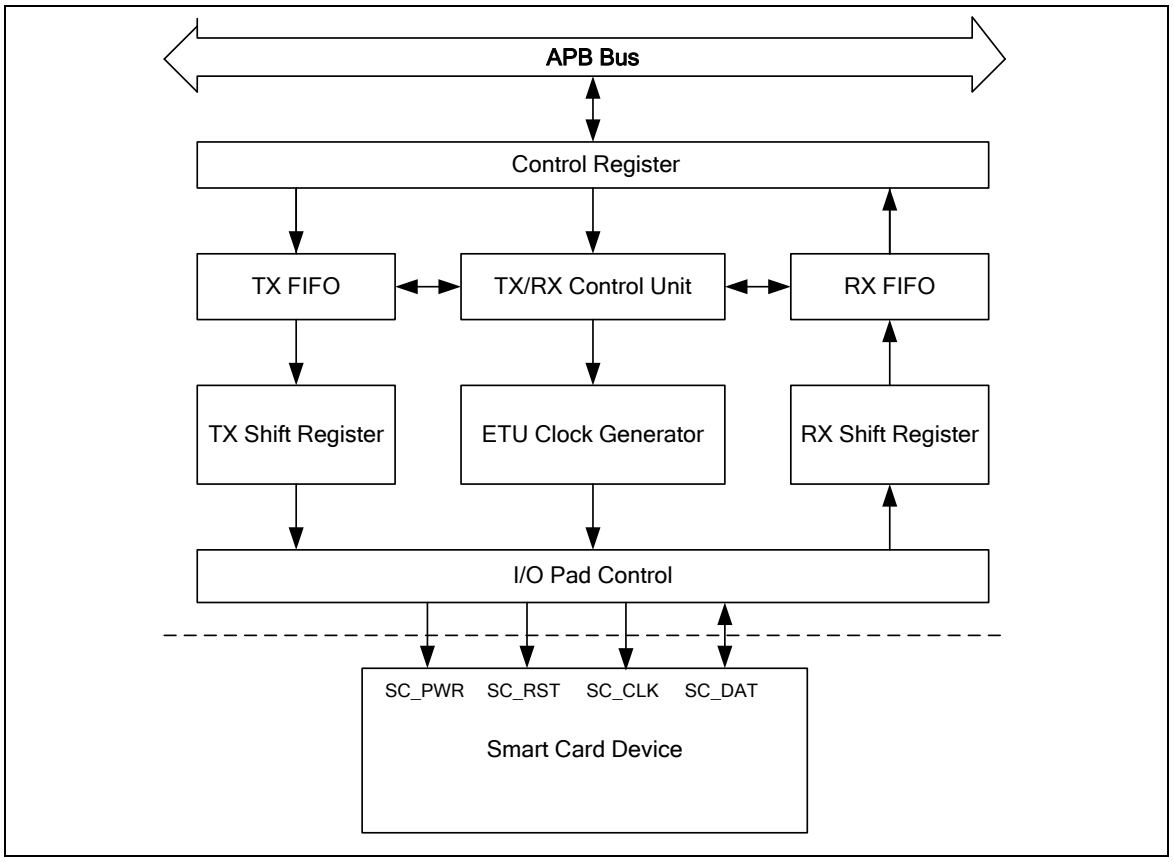


Figure 5-118 SC Controller Block Diagram

5.19.4 Functional Description

Basically, the smart card interface acts as a half-duplex asynchronous communication port and its data format is composed of ten consecutive bits, which is shown follows.

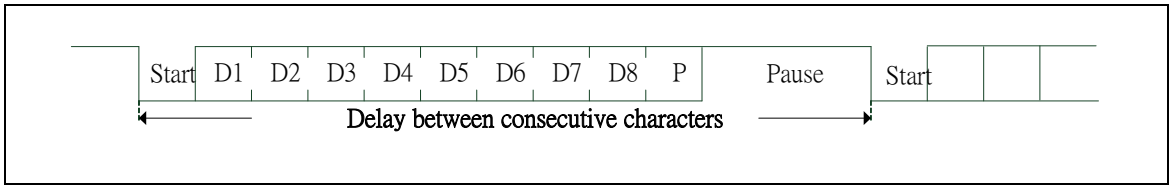


Figure 5-119 SC Data Character

5.19.4.1 *Activation, Warm Reset and Deactivation Sequence*

The Smart Card Interface controller supports hardware activation, warm reset and deactivation sequence.

Activation

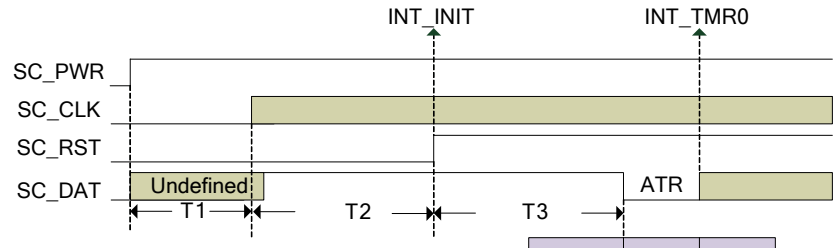
The activation sequence is shown as follows.

- Set SC_RST to low
- Set SC_PWR at high level and SC_DAT at high level (reception mode) at the same time.
- Enable SC_CLK clock
- De-assert SC_RST to high

The activation sequence can be controlled by software or hardware. If software wants to control it, software can control the SC_PINCSR and SC_TMRx register to process the activation sequence or setting SC_ALTCTL[ACT_EN] register, and then the interface will perform hardware activation sequence.

Following is activation control sequence in hardware activation mode:

- Set activation timing by setting SC_ALTCTL[INIT_SEL].
- TMR0 can be selected when SC_CTL[TMR_SEL] is 01, 10 or 11.
- Set operation mode SC_TMR0[MODE] to 0011 and give an Answer to Request value by setting SC_TMR0[CNT] register.
- When hardware de-asserts SC_RST to high, hardware will generator an initial end interrupt to CPU at the same time (if SC_IER[INIT_IE] = 1)
- If the TMR0 decreases the counter to “1” (start from SC_RST) and the card does not respond ATR before that time, hardware will generate interrupt INT_TMR0 to CPU.



Time	Comment
T1	SC_PWR to SC_CLK Start
T2	SC_CLK Start to SC_RST Assert
T3	SC_CLK Start to ART Appear

INIT_SEL	T1	T2
00	85	489
01	133	537
10	165	569
11	165	42060

Unit : SC Clock

Note : These value are measured by chip IO pin, the real value will depended on system design

Figure 5-120 SC Activation Sequence

Warm Reset

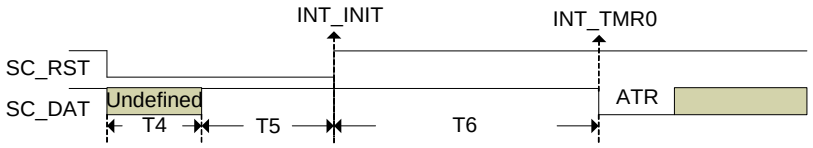
The warm reset sequence is shown as follows.

- Set SC_RST to low and set SC_DAT to high at the same time.
- Set SC_RST to high.

The warm reset sequence can be controlled by software or hardware. If software wants to control it, software can control SC_PINCSR and SC_TMRx register to process the warm reset sequence or set SC_ALTCTL[WARST_EN] register, and then the interface will perform hardware warm reset sequence.

Following is warm reset control sequence in hardware warm reset mode

- Set warm reset timing by setting SC_ALTCTL[INIT_SEL].
- Select TMR0 by setting SC_CTL[TMR_SEL] register (TMR_SEL can be 01, 10, or 11).
- Set operation mode SC_TMR0[MODE] to 0011 and give an Answer to Request value by setting SC_TMR0[CNT] register.
- Set TMR0_SEN and WARST_EN to start counting by SC_ALTCTL register.
- When hardware de-asserts SM_RST to high, hardware will generate an initial end interrupt to CPU at the same time (if SC_IER[INIT_IE] = “1”)
- If the TMR0 decrease the counter to “1” (start from SC_RST) and the card does not response ATR before that time, hardware will generate interrupt INT_TMR0 to CPU.



Time	Comment
T4	SC_RST to SC_DAT Reception Mode
T5	SC_DAT Reception Mode to SC_RST Assert
T6	SC_CLK Start to ATR Appear

INIT_SEL	T4	T5
00	81	483
01	129	531
10	161	563
11	161	42106

Unit:SC Clock

Note: This value is measured by chip IO pin, the real value will depend on system design.

Figure 5-121 SC Warm Reset Sequence

Deactivation

The deactivation sequence is shown as follows

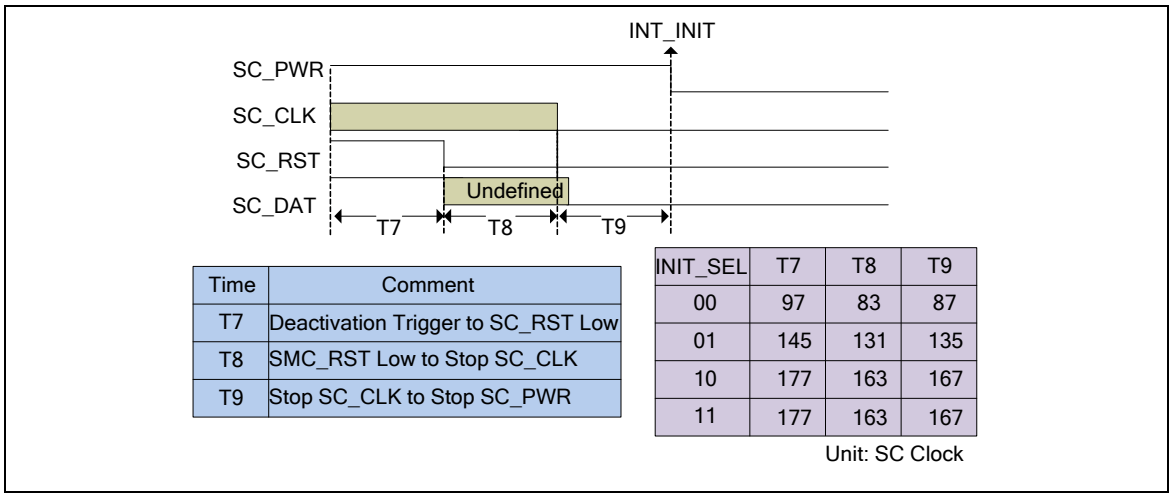
- Set SC_RST to low.
- Stop SC_CLK.
- Set SC_DAT to state low.
- Deactivated SC_PWR.

The deactivation sequence can be controlled by software or hardware. If software wants to control it, software can control SC_PINCSR and SC_TMR0 register to process the deactivation sequence or set SC_ALTCTL[DACT_EN] register, and then the interface will perform hardware deactivation sequence.

The SC controller also supports auto deactivation sequence when the card removal detection is set (SC_PINCSR[ADAC_CDEN]).

Following is deactivation control sequence in hardware deactivation mode:

- Set deactivation timing by setting SC_ALTCTL[INIT_SEL].
- Set DACT_EN to start counting by SC_ALTCTL register.
- When hardware de-asserts SC_PWR to low, the controller will generate an interrupt INT_INIT to CPU at the same time (if SC_IER[INIT_IE] = 1)



Time	Comment
T7	Deactivation Trigger to SC_RST Low
T8	SMC_RST Low to Stop SC_CLK
T9	Stop SC_CLK to Stop SC_PWR

INIT_SEL	T7	T8	T9
00	97	83	87
01	145	131	135
10	177	163	167
11	177	163	167

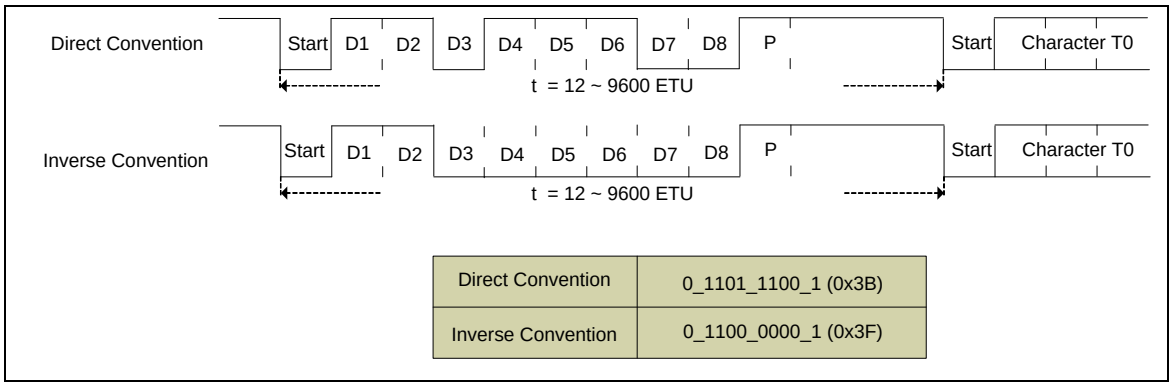
Unit: SC Clock

Figure 5-122 SC Deactivation Sequence

5.19.4.2 *Initial Character TS*

According to ISO7816-3, the initial character TS of answer to request (ATR) has two possible patterns (as shown in the following figure). If the TS pattern is 0_1100_0000_1, it is inverse convention. When decoded by inverse convention, the conveyed byte is equal to '3F'. If the TS pattern is 0_1101_1100_1, it is direct convention. When decoded by direct convention, the conveyed byte is equal to '3B'. Software can set SC_CTL[AUTO_CON_EN] and then the operating convention will be decided by hardware. Software can also set the SC_CTL[CON_SEL] register (set to 00 or 11) to change the operating convention after SC received TS of answer to request (ATR).

If software enables auto convention function by setting SC_CTL[AUTO_CON_EN] register, the setting step must be done before Answer to Request state and the first data must be 0x3B or 0x3F. After hardware received first data and stored it at buffer, hardware will decide the convention and change the SC_CTL[CON_SEL] register automatically. If the first data is neither 0x3B nor 0x3F, hardware will generate an auto-convention error interrupt INT_ACON_ERR (if SC_IER[ACON_ERR_IE] = 1) to CPU.



Direct Convention	0_1101_1100_1 (0x3B)
Inverse Convention	0_1100_0000_1 (0x3F)

Figure 5-123 Initial Character TS

5.19.4.3 *Error signal and character repetition*

According to ISO7816-3 T=0 mode description, as shown in following, if the receiver receives a wrong parity bit, it will pull the SC_DAT to low one to two bit period to inform the transmitter parity error. Then the transmitter will retransmit the character. The SC interface controller supports hardware error detection function in receiver and supports hardware re-transmit function in transmitter. Software can enable re-transmit function by setting SC_CTL[TX_ERETRY_EN]. Software can also define the retry (re-transmit) number limitation in SC_CTL[TX_ERETRY] register. The re-transmit number is up to TX_ERETRY +1 and if the re-transmit number is equal to TX_ERETRY +1, TX_OVER_REERR flag will be set by hardware and if SC_IER[TERR_IE] = 1, SC controller will generate a transfer error interrupt to CPU. Software can also define the received retry number limitation in SC_CTL[RX_ERETRY] register. The receiver retry number is up to RX_ERETRY +1, if the number of received errors by receiver is equal to RX_ERETRY +1, receiver will receive this error data to buffer and RX_OVER_REERR flag will be set by hardware and if SC_IER[TERR_IE] = 1, SC controller will generate a transfer error interrupt to CPU.

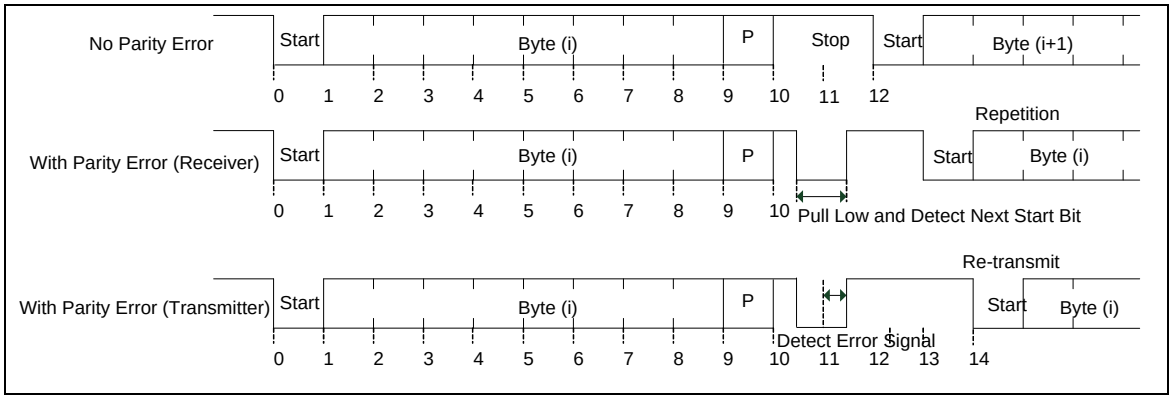


Figure 5-124 SC Error Signal

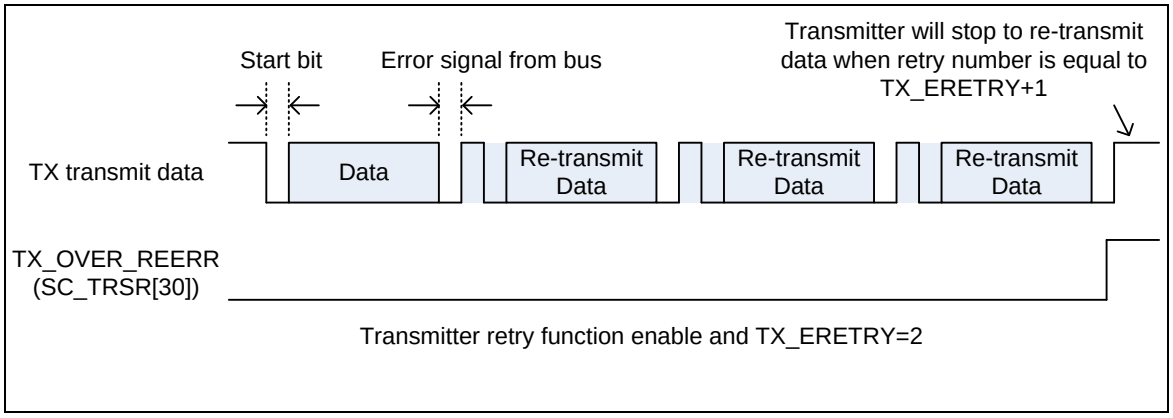


Figure 5-125 SC Transmitter Retry Number and Retry Over Flag

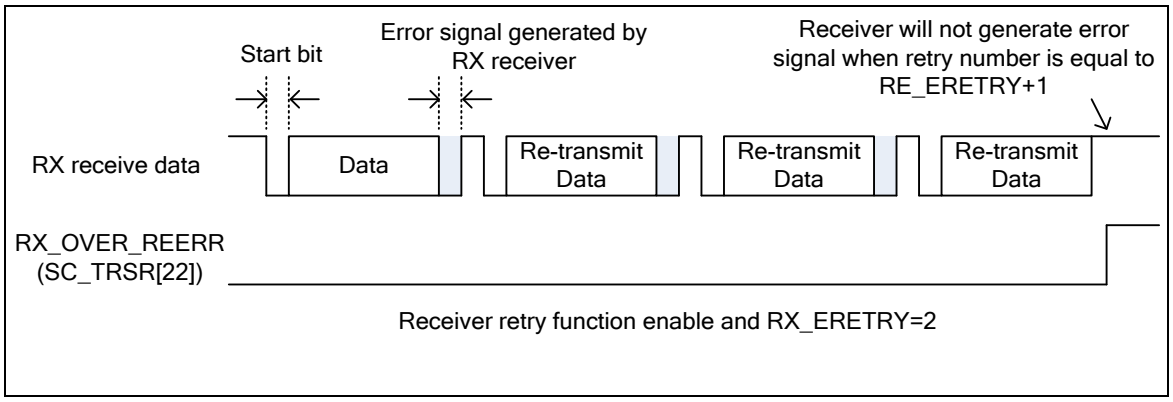


Figure 5-126 SC Receiver Retry Number and Retry Over Flag

5.19.4.4 *Internal time-out counter*

The smart card interface includes a 24-bit time-out counter and two 8-bit time-out counters. These counters help the controller in processing different real-time interval (ATR, WWT, BWT, etc.). Each counter can be set to start counting once the trigger enable bit has been written or a START bit has been detected.

The following is the programming flow:

- Enable counter by setting SC_CTL[TMR_SEL].
- Select operation mode SC_TMRx[MODE] and give a count value SC_TMRx[CNT] by setting SC_TMRx register.
- Set SC_ALTCTL[TMRx_SEN] to start counting.

5.20 FLASH MEMORY CONTROLLER (FMC)

5.20.1 Overview

The NuMicro™ NUC200 Series has 128/64/32K bytes on-chip embedded Flash for application program memory (APROM) that can be updated through ISP procedure. The In-System-Programming (ISP) function enables user to update program memory when chip is soldered on PCB. After chip is powered on, Cortex™-M0 CPU fetches code from APROM or LDROM decided by boot select (CBS) in Config0. By the way, the NuMicro™ NUC200 Series also provides additional DATA Flash for user to store some application dependent data. For 128K bytes APROM device, the data flash is shared with original 128K program memory and its start address is configurable in Config1. For 64K/32K bytes APROM device, the data flash is fixed at 4K.

5.20.2 Features

- Runs up to 50 MHz with zero wait state for continuous address read access
- All embedded flash memory supports 512 bytes page erase
- 128/64/32 KB application program memory (APROM)
- 4 KB In-System-Programming (ISP) loader program memory (LDROM)
- 4KB data flash for 64/32 KB APROM device
- Configurable data flash size for 128KB APROM device
- Configurable or fixed 4 KB data flash with 512 bytes page erase unit
- Supports In-Application-Programming (IAP) to switch code between APROM and LDROM without reset
- In-System-Programming (ISP) to update on-chip Flash

5.20.3 Block Diagram

The flash memory controller 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:

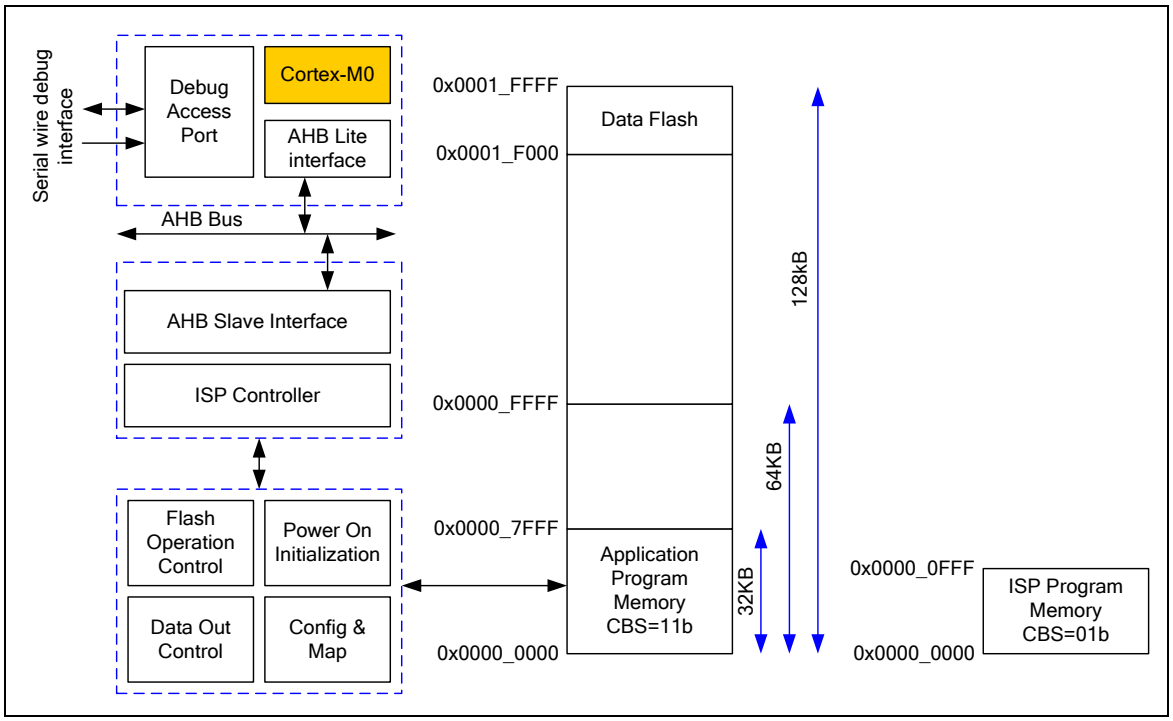


Figure 5-127 Flash Memory Control Block Diagram

5.20.4 Functional Description

5.20.4.1 *Flash Memory Organization*

The NuMicro™ NUC200 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 user 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. The size of each erase unit is 512 bytes. For 128 KB APROM device, the data flash and application program share the same 128 KB memory, if DFEN (Data Flash Enable) bit in Config0 is enabled, the data flash base address is defined by DFBADR 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™ Family, 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 ~ 0xFFF. However, if user want to read LDROM by ISP, they still need to read the address of LDROM as 0x0010_0000 ~ 0x0010_0FFF.

Table 5-12 and Figure 5-128 show the address mapping information of APROM, LDROM, data flash and user configuration for 32/64 and 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	4 KB		0x0010_0000	0x0010_0FFF

User Configuration	32/64/128 KB	2 words	0x0030_0000	0x0030_0004

Table 5-12 Memory Address Map

The Flash memory organization is shown as Figure 5-128:

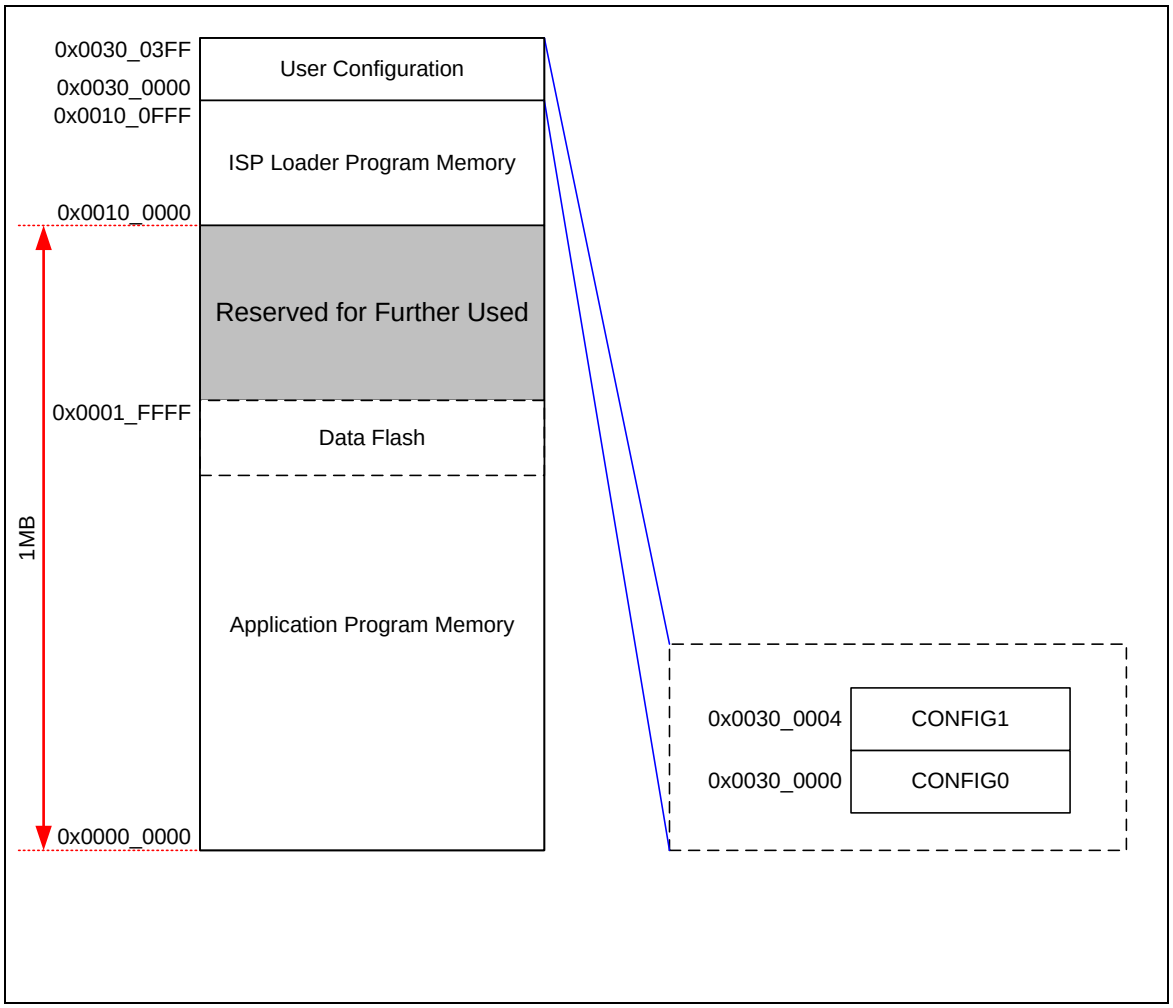


Figure 5-128 Flash Memory Organization

5.20.4.2 *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	CKF	CGPFMFP	CFOSC		
23	22	21	20	19	18	17	16
CBODEN	CBOV		CBORST	Reserved			
15	14	13	12	11	10	9	8
Reserved					CIOINI	Reserved	
7	6	5	4	3	2	1	0
CBS		Reserved				LOCK	DFEN

Config	Address = 0x0030_0000	
Bits	Descriptions	
[31]	**CWDTEN**	**Watchdog Enable** 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 Enable** 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]	**Reserved**	Reserved
[28]	**CKF**	**XT1 Clock Filter Enable** 0 = XT1 clock filter Disabled. 1 = XT1 clock filter Enabled.
[27]	**CGPFMFP**	**GPF Multi-Function Select** 0 = XT1_IN and XT1_OUT pin is configured as GPIO function. 1 = XT1_IN and XT1_OUT pin is used as external 4~24MHz crystal oscillator pin. **Note:** XT1_IN, XT1_OUT multi-function can only be changed by CGPFMFP.

<table>
<tr><td>[26:24]</td><td>CFOSC</td><td>CPU Clock Source Selection After Reset<table><tr><th>FOSC[2:0]</th><th>Clock Source</th></tr><tr><td>000</td><td>External 4~24 MHz high speed crystal oscillator clock</td></tr><tr><td>111</td><td>Internal RC 22.1184 MHz high speed oscillator clock</td></tr><tr><td>Others</td><td>Reserved</td></tr></table>The value of CFOSC will be load to CLKSEL0.HCLK_S[2:0] in system register after any reset occurs.</td></tr>
<tr><td>[23]</td><td>CBODEN</td><td>Brown-out Detector Enable
0= Brown-out detect Enabled after powered on.
1= Brown-out detect Disabled after powered on.</td></tr>
<tr><td>[22:21]</td><td>CBOV</td><td>Brown-out Voltage Selection<table><tr><th>CBOV</th><th>Brown-out Voltage</th></tr><tr><td>11</td><td>4.4 V</td></tr><tr><td>10</td><td>3.7 V</td></tr><tr><td>01</td><td>2.7 V</td></tr><tr><td>00</td><td>2.2 V</td></tr></table></td></tr>
<tr><td>[20]</td><td>CBORST</td><td>Brown-out Reset Enable
0 = Brown-out reset Enabled after powered on.
1 = Brown-out reset Disabled after powered on.</td></tr>
<tr><td>[19:11]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[10]</td><td>CIOINI</td><td>IO Initial State Select
0 = All GPIO default to be input tri-state mode after powered on.
1 = All GPIO default to be Quasi-bidirectional mode after chip is powered on .</td></tr>
<tr><td>[9:8]</td><td>Reserved</td><td>Reserved</td></tr>
</table>

<table>
<tr><td>[7:6]</td><td>CBS</td><td>Chip Boot Selection
<table>
<tr><th>CBS[1:0]</th><th>Boot Selection</th></tr>
<tr><td>11</td><td>APROM without IAP function.
Chip booting from APROM and program executing range only including APROM. LDROM cannot be access by program directly, except by through ISP.
APROM is write-protected in this mode.</td></tr>
<tr><td>01</td><td>LDROM without IAP function.
Chip booting from LDROM, program executing range only including LDROM; APROM cannot be access by program directly, except by through ISP.
LDROM is write-protected in this mode.</td></tr>
<tr><td>10</td><td>APROM with IAP function.
Chip booting from APROM, program executing range including LDROM and APROM
LDROM address is mapping to 0x0010_0000~0x0010_0FFF
The address 0x0000_0000 ~ 0x0000_01FF can be re-mapping to any other page within executing range though ISP command.</td></tr>
<tr><td>00</td><td>LDROM with IAP function.
Chip booting from LDROM, program executing range including LDROM and most of APROM (all but except first 512 bytes, because the first 512 bytes is mapped from LDROM)
LDROM address is mapping to 0x0010_0000 ~ 0x0010_0FFF, and also the first 512 bytes of LDROM is mapping to the address 0x0000_0000 ~ 0x0000_01FF.
The address 0x0000_0000 ~ 0x0000_01FF can be re-mapping to any other page within executing range though ISP command.</td></tr>
</table></td></tr>
<tr><td>[5:2]</td><td>Reserved</td><td>Reserved</td></tr>
<tr><td>[1]</td><td>LOCK</td><td>Security Lock
0 = Flash data is locked
1 = Flash data is not locked
When flash data is locked, only device ID, Config0 and Config1 can be read by writer and ICP through serial debug interface. Others data is locked as 0xFFFFFFFF. ISP can read data anywhere regardless of LOCK bit value.
User need to erase whole chip by ICP/Writer tool or erase user configuration by ISP to unlock.</td></tr>
<tr><td>[0]</td><td>DFEN</td><td>Data Flash Enable
0 = Data flash Enabled.
1 = Data flash Disabled.
Note: This bit only for 128 KB APROM Device</td></tr>
</table>

Config1 (Address = 0x0030_0004)

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

Config	Address = 0x0030_0004	
Bits	Descriptions	
[31:20]	**Reserved**	Reserved (It is mandatory to program 0x00 to these Reserved bits)
[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.

5.20.4.3 *Boot Selection*

The NuMicro™ NUC200 Series provides In-System-Programming (ISP) feature to enable user to update program memory by a stand-alone ISP firmware. A dedicated 4 KB program memory (LDROM) is used to store ISP firmware. User can select to start program fetch from APROM or LDROM by CBS[1] in Config0.

In addition to setting boot from APROM or LDROM, CBS in Config0 is also used to control system memory map after booting. When CBS[0] = 1 and set CBS[1] = 1 to boot from APROM, the application in APROM will not be able to access LDROM by memory read. In other words, when CBS[0] = 1 and CBS[1] = 0 are set to boot from LDROM, the software executed in LDROM will not be able to access APROM by memory read. Figure 5-129 shows the memory map when booting from APROM and LDROM.

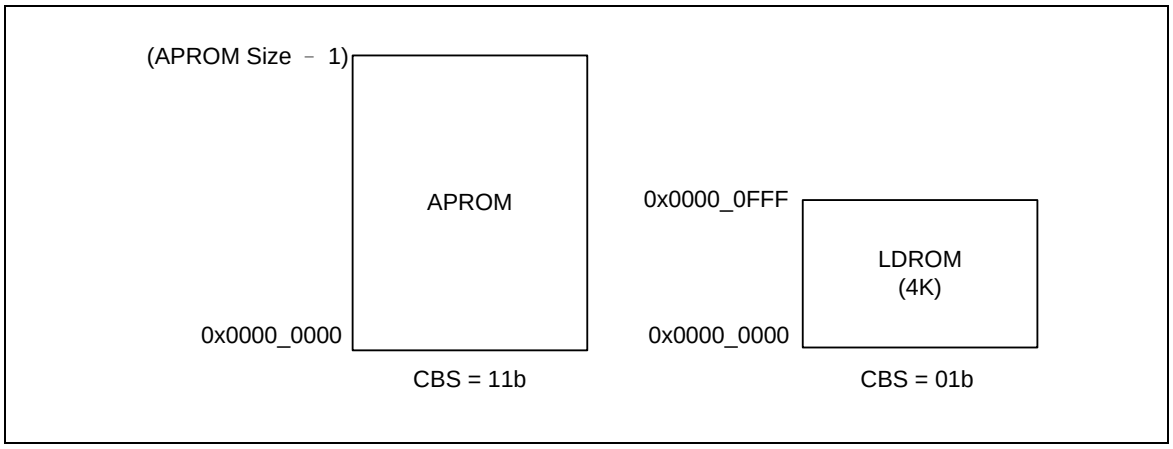


Figure 5-129 Program Executing Range for Booting from APROM and LDROM

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

5.20.4.4 *In-Application-Programming (IAP)*

The NuMicro™ NUC200 Series 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 re-booting chip and setting the chip boot selection bits in Config0 (CBS[1:0]) as 10b or 00b.

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 4 KB LDROM is mapped to 0x0010_0000~ 0x0010_0FFF.

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 4 KB LDROM is mapped to 0x0010_0000~0x0010_0FFF.

Please refer to Figure 5-130 for the address map while IAP is activating.

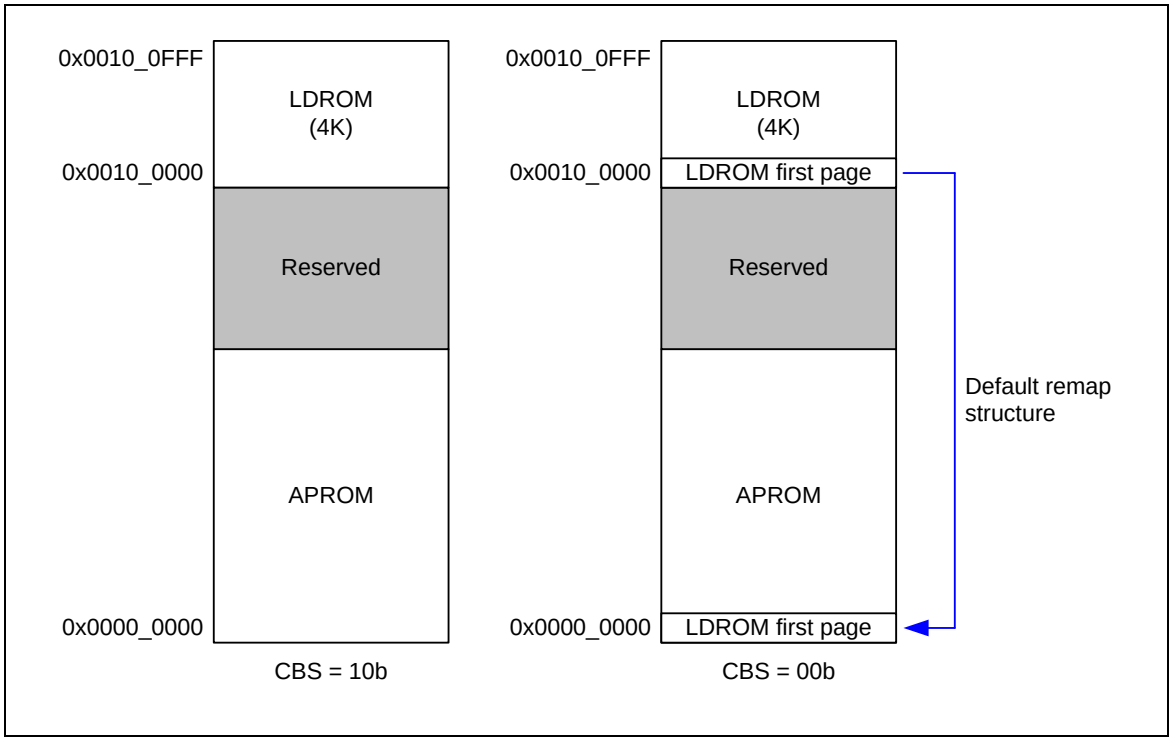


Figure 5-130 Executable Range of Code with IAP Function Enabled

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 field in the ISPSTA register.

5.20.4.5 *In-System-Programming (ISP)*

The NuMicro™ NUC200 Series supports ISP mode which allows a device to be reprogrammed under software control and avoids system fail risk when download or programming fail. Furthermore, the capability to update the application firmware makes a wide range of applications possible.

ISP provides the ability to update system firmware on board. Various peripheral interfaces let ISP loader in LDROM to receive new program code easily. The most common method to perform ISP is via UART along with the ISP loader in LDROM. General speaking, PC transfers the new APROM code through serial port. Then ISP loader receives it and re-programs into APROM through ISP commands.

5.20.4.6 *ISP Procedure*

The NuMicro™ NUC200 Series supports booting from APROM or LDROM initially defined by user configuration. The change of user configuration needs to reboot system to make it take effect. If user wants to switch between APROM or LDROM mode without changing user configuration, he needs to control BS bit of ISPCON control register, then reset CPU by IPRSTC1 control register. The boot switching flow by BS bit is shown in the following figure.

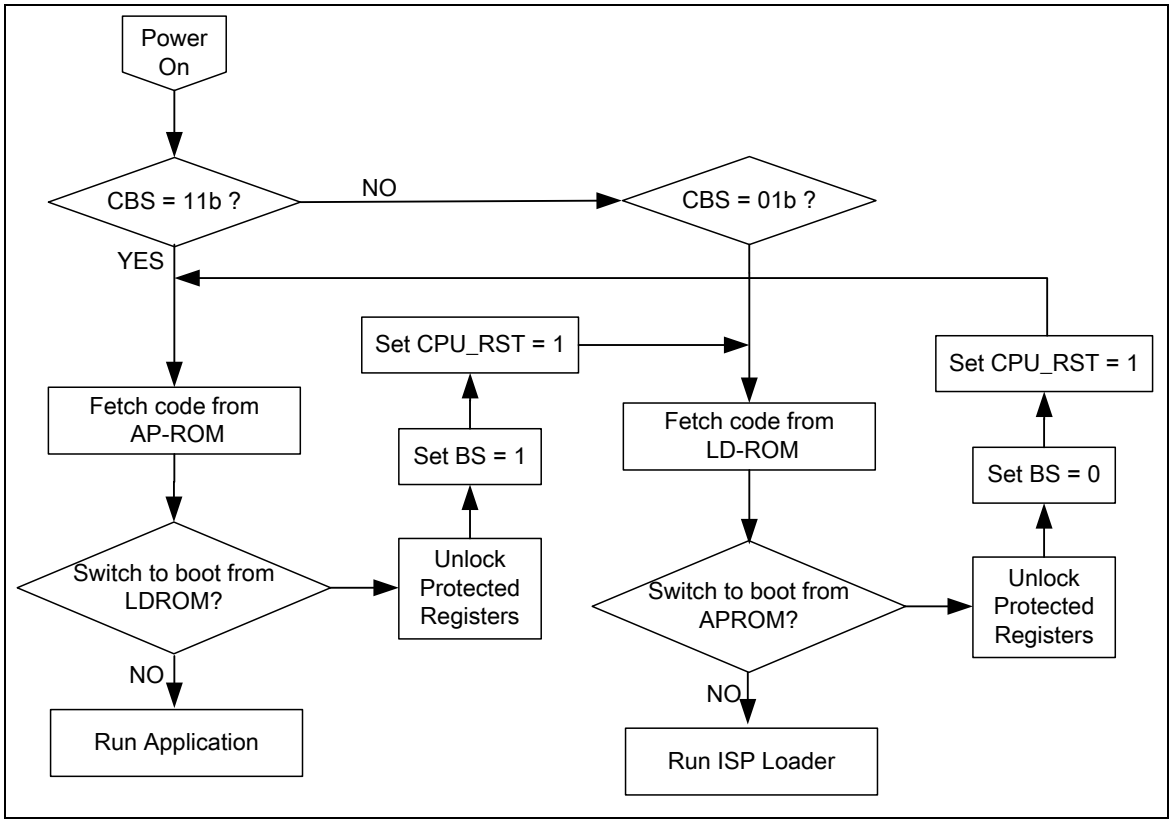


Figure 5-131 Example Flow of Boot Selection by BS Bit

Updating APROM by software in LDROM or updating LDROM by software in APROM can avoid a

system failure when update fails.

The ISP controller supports to read, erase and program embedded flash memory. Several control bits of ISP controller are write-protected, thus it is necessary to unlock before we can set them. To unlock the protected register bits, software needs to write 0x59, 0x16 and 0x88 sequentially to REGWRPROT. If register is unlocked successfully, the value of REGWRPROT will be 1. The unlock sequence must not be interrupted by other access; otherwise it may fail to unlock.

After unlocking the protected register bits, user needs to set the ISPCON control register to decide to update LDROM, User Configuration, APROM and enable ISP controller.

Once the ISPCON register is set properly, user can set ISPCMD for erase, read or programming. Set ISPADR for target flash memory based on flash memory origination. ISPDAT can be used to set the data to program or used to return the read data according to ISPCMD.

Finally, set ISPGO bit of ISPTRG control register to perform the relative ISP function. The ISPGO bit is self-cleared when ISP function has been done. To make sure ISP function has been finished before CPU goes ahead, ISB instruction is used right after ISPGO setting.

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

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

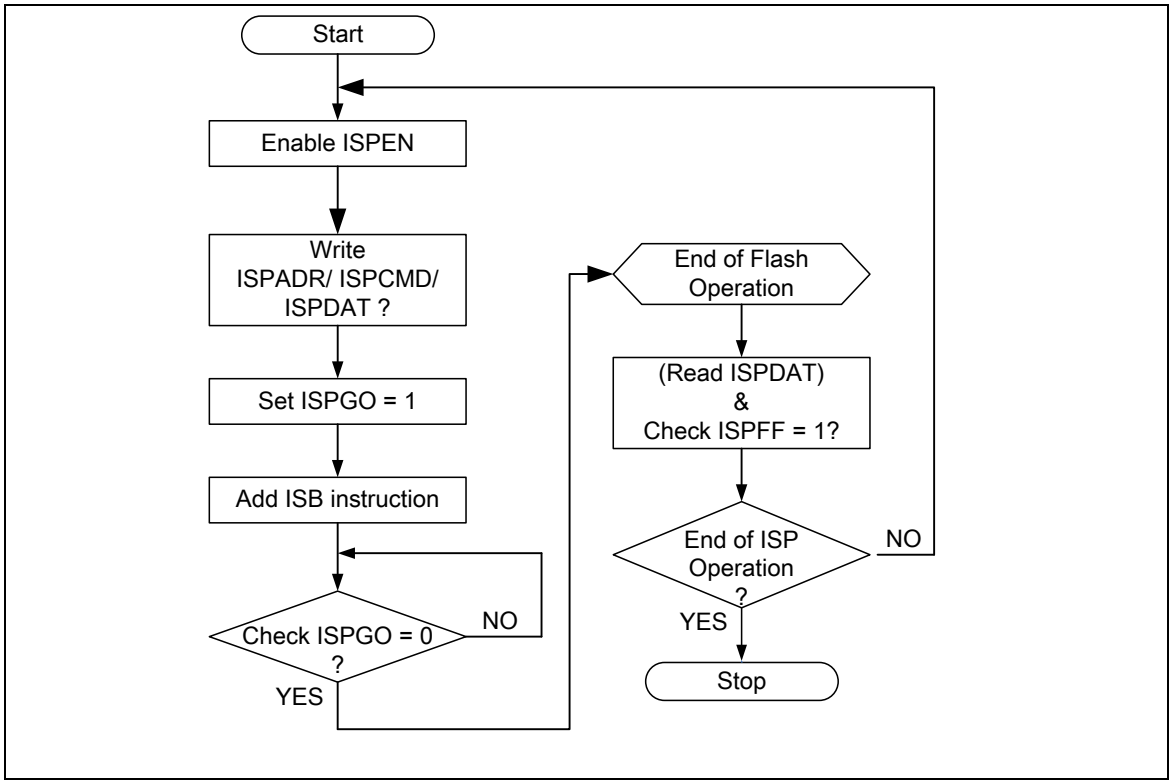


Figure 5-132 ISP Flow Example

ISP Command	ISPCMD	ISPADR	ISPDAT
FLASH Page Erase	0x22	Valid address of flash memory origination. It must be 512 bytes page alignment.	N/A
FLASH Program	0x21	Valid address of flash memory origination	Programming Data
FLASH Read	0x00	Valid address of flash memory origination	Return Data
Read Unique ID	0x04	0x0000_0000	Unique ID Word 0
		0x0000_0004	Unique ID Word 1
		0x0000_0008	Unique ID Word 2
Vector Page Re-Map	0x2E	Page in APROM or LDROM It must be 512 bytes page alignment	N/A

Table 5-13 ISP Command List

6 ELECTRICAL CHARACTERISTICS

6.1 Absolute Maximum Ratings

SYMBOL	PARAMETER	MIN.	MAX	UNIT
DC Power Supply	V_{DD}–V_{SS}	-0.3	+7.0	V
Input Voltage	V_{IN}	V_{SS}-0.3	V_{DD}+0.3	V
Oscillator Frequency	$1/t_{CLCL}$	4	24	MHz
Operating Temperature	TA	-40	+85	°C
Storage Temperature	TST	-55	+150	°C
Maximum Current into V_{DD}		-	120	mA
Maximum Current out of V_{SS}			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.

6.2 DC Electrical Characteristics

(V_{DD}-V_{SS}=5.5 V, T_A = 25°C, F_{OSC} = 50 MHz unless otherwise specified.)

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Operation Voltage	V_{DD}	2.5		5.5	V	V_{DD} = 2.5V ~ 5.5V up to 50 MHz
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}		V_{DD}		V	When system used analog function, please refer to chapter 6.4 for corresponding analog operating voltage
Operating Current Normal Run Mode at 50 MHz	I_{DD1}		34		mA	V_{DD} = 5.5V, All IP and PLL enabled, XTAL = 12 MHz
	I_{DD2}		15		mA	V_{DD} = 5.5V, All IP disabled and PLL enabled, XTAL = 12 MHz
	I_{DD3}		32		mA	V_{DD} = 3.3V, All IP and PLL enabled, XTAL = 12 MHz
	I_{DD4}		14		mA	V_{DD} = 3.3V, All IP disabled and PLL enabled, XTAL = 12 MHz
Operating Current Normal Run Mode at 12 MHz	I_{DD5}		8.5		mA	V_{DD} = 5.5V, All IP enabled and PLL disabled, XTAL = 12 MHz
	I_{DD6}		3.6		mA	V_{DD} = 5.5V, All IP and PLL disabled, XTAL = 12 MHz
	I_{DD7}		7.5		mA	V_{DD} = 3.3V, All IP enabled and PLL disabled, XTAL = 12 MHz
	I_{DD8}		2.6		mA	V_{DD} = 3.3V, All IP and PLL disabled, XTAL = 12 MHz

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Operating Current Normal Run Mode at 4 MHz	I_{DD9}		3.6		mA	V_{DD} = 5.5V, All IP enabled and PLL disabled, XTAL = 4 MHz
	I_{DD10}		2		mA	V_{DD} = 5.5V, All IP and PLL disabled, XTAL = 4 MHz
	I_{DD11}		2.8		mA	V_{DD} = 3.3V, All IP enabled and PLL disabled, XTAL = 4 MHz
	I_{DD12}		1.2		mA	V_{DD} = 3.3V, All IP and PLL disabled, XTAL = 4 MHz
Operating Current Normal Run Mode at 32.768 kHz	I_{DD13}		141		μA	V_{DD} = 5.5V, All IP enabled and PLL disabled, XTAL = 32.768 kHz
	I_{DD14}		129		μA	V_{DD} = 5.5V, All IP and PLL disabled, XTAL = 32.768 kHz
	I_{DD15}		138		μA	V_{DD} = 3.3V, All IP enabled and PLL disabled, XTAL = 32.768 kHz
	I_{DD16}		125		μA	V_{DD} = 3.3V, All IP and PLL disabled, XTAL = 32.768 kHz
Operating Current Normal Run Mode at 10 kHz	I_{DD17}		125		μA	V_{DD} = 5.5V, All IP enabled and PLL disabled, LIRC10 kHz enabled
	I_{DD18}		120		μA	V_{DD} = 5.5V, All IP and PLL disabled, LIRC10 kHz enabled
	I_{DD19}		125		μA	V_{DD} = 3.3V, All IP enabled and PLL disabled, LIRC10 kHz enabled

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
	I_{DD20}		120		μA	V_{DD} = 3.3V, All IP and PLL disabled, LIRC10kHz enabled
Operating Current Idle Mode at 50 MHz	I_{IDLE1}		28		mA	V_{DD} = 5.5V, All IP and PLL enabled, XTAL = 12 MHz
	I_{IDLE2}		10		mA	V_{DD} = 5.5V, All IP disabled and PLL enabled, XTAL = 12 MHz
	I_{IDLE3}		27		mA	V_{DD} = 3.3V, All IP and PLL enabled, XTAL = 12 MHz
	I_{IDLE4}		9		mA	V_{DD} = 3.3V All IP disabled and PLL enabled, XTAL = 12 MHz
Operating Current Idle Mode at 12 MHz	I_{IDLE5}		7.5		mA	V_{DD} = 5.5V, All IP enabled and PLL disabled, XTAL = 12 MHz
	I_{IDLE6}		2.4		mA	V_{DD} = 5.5V, All IP and PLL disabled, XTAL = 12 MHz
	I_{IDLE7}		6.5		mA	V_{DD} = 3.3V, All IP enabled and PLL enabled, XTAL = 12 MHz
	I_{IDLE8}		1.5		mA	V_{DD} = 3.3V, All IP and PLL disabled, XTAL = 12 MHz
Operating Current Idle Mode at 4 MHz	I_{IDLE9}		3.3		mA	V_{DD} = 5.5V, All IP enabled and PLL disabled, XTAL = 4 MHz
	I_{IDLE10}		1.7		mA	V_{DD} = 5.5V All IP and PLL disabled, XTAL = 4 MHz
	I_{IDLE11}		2.4		mA	V_{DD} = 3.3V, All IP enabled and PLL disabled, XTAL = 4 MHz

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
	I_{IDLE12}		0.8		mA	V_{DD} = 3.3V, All IP and PLL disabled, XTAL = 4 MHz
Operating Current Idle Mode at 32.768 kHz	I_{IDLE13}		133		μA	V_{DD} = 5.5V, All IP enabled and PLL disabled, XTAL = 32.768 kHz
	I_{IDLE14}		120		μA	V_{DD} = 5.5V, All IP and PLL disabled, XTAL = 32.768 kHz
	I_{IDLE15}		133		μA	V_{DD} = 3.3V, All IP enabled and PLL disabled, XTAL = 32.768 kHz
	I_{IDLE16}		120		μA	V_{DD} = 3.3V, All IP and PLL disabled, XTAL = 32.768 kHz
Operating Current Idle Mode at 10 kHz	I_{IDLE13}		122		μA	V_{DD} = 5.5V, All IP enabled and PLL disabled, LIRC10 kHz enabled
	I_{IDLE14}		118		μA	V_{DD} = 5.5V, All IP and PLL disabled, LIRC10 kHz enabled
	I_{IDLE15}		122		μA	V_{DD} = 3.3V, All IP enabled and PLL disabled, LIRC10 kHz enabled
	I_{IDLE16}		118		μA	V_{DD} = 3.3V All IP and PLL disabled, LIRC10 kHz enabled
Standby Current Power-down Mode (Deep Sleep Mode)	I_{PWD1}		15		μA	V_{DD} = 5.5V, RTC disabled, When BOD function disabled
	I_{PWD2}		15		μA	V_{DD} = 3.3V, RTC disabled, When BOD function disabled
	I_{PWD3}		17		μA	V_{DD} = 5.5V, RTC enabled , When BOD function disabled
	I_{PWD4}		17		μA	V_{DD} = 3.3V, RTC enabled , When BOD function disabled
Input Current PA, PB, PC, PD, PE, PF (Quasi-bidirectional mode)	I_{IN1}		-50	-60	μA	V_{DD} = 5.5V, V_{IN} = 0V or V_{IN}=V_{DD}

PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Input Current at nRESET[1]	I_{IN2}	-55	-45	-30	μA	V_{DD} = 3.3V, V_{IN} = 0.45V
Input Leakage Current PA, PB, PC, PD, PE, PF	I_{LK}	-2	-	+2	μA	V_{DD} = 5.5V, 0<V_{IN}<V_{DD}
Logic 1 to 0 Transition Current PA~PF (Quasi-bidirectional mode)	I_{TL} [3]	-650	-	-200	μA	V_{DD} = 5.5V, V_{IN}<2.0V
Input Low Voltage PA, PB, PC, PD, PE, PF (TTL input)	V_{IL1}	-0.3	-	0.8	V	V_{DD} = 4.5V
		-0.3	-	0.6		V_{DD} = 2.5V
Input High Voltage PA, PB, PC, PD, PE, PF (TTL input)	V_{IH1}	2.0	-	V_{DD} +0.2	V	V_{DD} = 5.5V
		1.5	-	V_{DD} +0.2		V_{DD} =3.0V
Input Low Voltage PA, PB, PC, PD, PE, PF (Schmitt input)	V_{IL2}	-0.3	-	0.3V_{DD}	V	
Input High Voltage PA, PB, PC, PD, PE, PF (Schmitt input)	V_{IH2}	0.7V_{DD}	-	V_{DD} +0.2	V	
Hysteresis voltage of PA, PB, PC, PD,PE, PF (Schmitt input)	V_{HY}		0.2V_{DD}		V	
Input Low Voltage XT1_IN[*2]	V_{IL3}	0	-	0.8	V	V_{DD} = 4.5V
		0	-	0.4		V_{DD} = 3.0V
Input High Voltage XT1_IN[*2]	V_{IH3}	3.5	-	V_{DD} +0.2	V	V_{DD} = 5.5V
		2.4	-	V_{DD} +0.2		V_{DD} = 3.0V
Input Low Voltage X32_IN[*2]	V_{IL4}	0	-	0.4	v	
Input High Voltage X32_IN[*2]	V_{IH4}	1.2		1.8	V	
Negative going threshold (Schmitt input), nRESET	V_{ILS}	-0.5	-	0.2V_{DD} -0.2	V	
Positive going threshold (Schmitt input), nRESET	V_{IHS}	0.7V_{DD}	-	V_{DD} +0.5	V	
Source Current PA, PB, PC, PD, PE, PF (Quasi-bidirectional Mode)	I_{SR11}	-300	-370	-450	μA	V_{DD} = 4.5V, V_S = 2.4V
	I_{SR12}	-50	-70	-90	μA	V_{DD} = 2.7V, V_S = 2.2V
	I_{SR12}	-40	-60	-80	μA	V_{DD} = 2.5V, V_S = 2.0V
Source Current PA, PB, PC, PD, PE, PF (Push-pull Mode)	I_{SR21}	-24	-28	-32	mA	V_{DD} = 4.5V, V_S = 2.4V
	I_{SR22}	-4	-6	-8	mA	V_{DD} = 2.7V, V_S = 2.2V
	I_{SR22}	-3	-5	-7	mA	V_{DD} = 2.5V, V_S = 2.0V

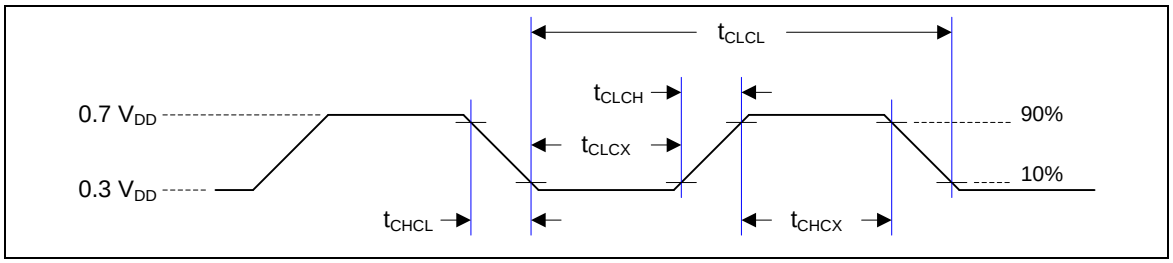
PARAMETER	SYM.	SPECIFICATION				TEST CONDITIONS
		MIN.	TYP.	MAX.	UNIT	
Sink Current PA, PB, PC, PD, PE, PF (Quasi-bidirectional and Push-pull Mode)	I_{SK1}	10	16	20	mA	V_{DD} = 4.5V, V_S = 0.45V
	I_{SK1}	7	10	13	mA	V_{DD} = 2.7V, V_S = 0.45V
	I_{SK1}	6	9	12	mA	V_{DD} = 2.5V, V_S = 0.45V
Brown-out Voltage with BOD_VL [1:0] = 00b	$V_{BO2.2}$	2.1	2.2	2.3	V	
Brown-out Voltage with BOD_VL [1:0] = 01b	$V_{BO2.7}$	2.6	2.7	2.8	V	
Brown-out voltage with BOD_VL [1:0] = 10b	$V_{BO3.8}$	3.5	3.7	3.9	V	
Brown-out Voltage with BOD_VL [1:0] = 11b	$V_{BO4.5}$	4.2	4.4	4.6	V	
Hysteresis range of BOD voltage	V_{BH}	30	-	150	mV	V_{DD} = 2.5V~5.5V

Note:

1. nRESET pin is a Schmitt trigger input.
2. Crystal Input is a CMOS input.
3. Pins of PA, PB, PC, PD, PE and PF can source a transition current when they are being externally driven from 1 to 0. In the condition of V_{DD} = 5.5 V, the transition current reaches its maximum value when V_{IN} approximates to 2 V.

6.3 AC Electrical Characteristics

6.3.1 External 4~24 MHz High Speed Oscillator



Note: Duty cycle is 50%.

SYMBOL	PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
t_{CHCX}	Clock High Time		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

6.3.2 External 4~24 MHz High Speed 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

6.3.2.1 *Typical Crystal Application Circuits*

CRYSTAL	C1	C2	R
4 MHz ~ 24 MHz	10~20pF	10~20pF	without

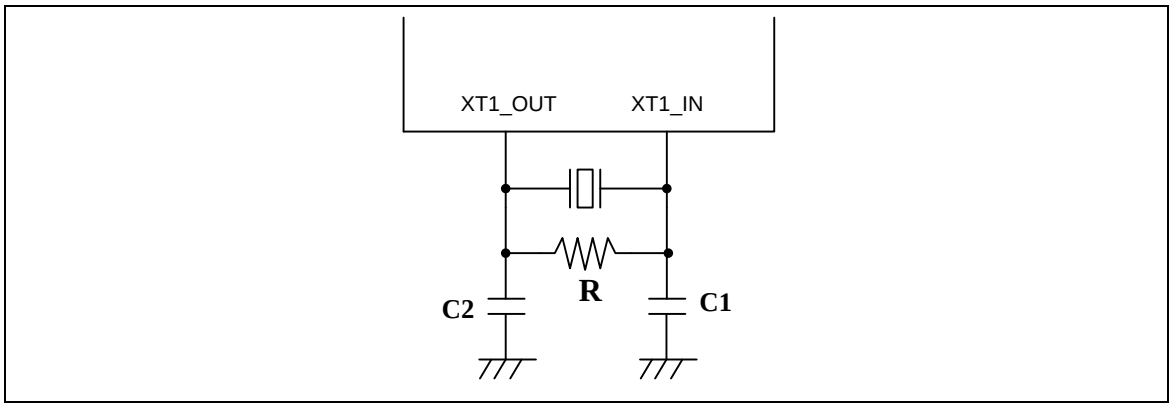


Figure 6-1 Typical Crystal Application Circuit

6.3.3 External 32.768 kHz Low Speed Crystal Oscillator

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Operation Voltage V_{DD}	-	2.5	-	5.5	V
Operation Temperature	-	-40	-	85	℃
Operation Current	32.768KHz at V_{DD}=5V		1.5		μA
Clock Frequency	External crystal	-	32.768	-	kHz

6.3.4 Internal 22.1184 MHz High Speed Oscillator

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Operation Voltage V_{DD}	-	2.5	-	5.5	V
Center Frequency	-	-	22.1184	-	MHz
Calibrated Internal Oscillator Frequency	+25℃; V_{DD} =5 V	-1	-	+1	%
	-40℃~+85℃; V_{DD}=2.5 V~5.5 V	-3	-	+3	%
Operation Current	V_{DD} =5 V	-	500	-	uA

6.3.5 Internal 10 kHz Low Speed Oscillator

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Operation Voltage V_{DD}	-	2.5	-	5.5	V
Center Frequency	-	-	10	-	kHz
Calibrated Internal Oscillator Frequency	+25℃; V_{DD} =5 V	-30	-	+30	%

	-40℃~+85℃; V_{DD}=2.5 V~5.5 V	-50	-	+50	%

6.4 Analog Characteristics

6.4.1 12-bit SARADC Specification

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT
-	Resolution	-	-	12	Bit
DNL	Differential nonlinearity error	-	-1~2	-1~4	LSB
INL	Integral nonlinearity error	-	±2	±4	LSB
EO	Offset error	-	±1	10	LSB
EG	Gain error (Transfer gain)	-	1	1.005	-
-	Monotonic	Guaranteed			
F_{ADC}	ADC clock frequency (AV_{DD} = 5V/3V)	-	-	16/8	MHz
F_S	Sample rate	-	-	760	kSPS
V_{DDA}	Supply voltage	3	-	5.5	V
I_{DD}	Supply current (Avg.)	-	0.5	-	mA
I_{DDA}		-	1.5	-	mA
V_{REF}	Reference voltage	3	-	V_{DDA}	V
I_{REF}	Reference current (Avg.)	-	1	-	mA
V_{IN}	Input voltage	0	-	V_{REF}	V

6.4.2 LDO and Power Management Specification

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	85	℃	
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.

6.4.3 Low Voltage Reset Specification

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Operation Voltage	-	0	-	5.5	V
Quiescent Current	V_{DD}=5.5 V	-	1	5	μA
Operation Temperature	-	-40	25	85	℃
Threshold Voltage	Temperature=25℃	1.7	2.0	2.3	V
	Temperature=-40℃	-	2.4	-	V
	Temperature=85℃	-	1.6	-	V
Hysteresis	-	0	0	0	V

6.4.4 Brown-out Detector Specification

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Operation Voltage	-	0	-	5.5	V
Operation Temperature	-	-40	25	85	℃
Quiescent Current	AV_{DD}=5.5 V	-	-	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

6.4.5 Power-on Reset Specification

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Operation Temperature	-	-40	25	85	℃
Reset Voltage	V+	-	2	-	V
Quiescent Current	Vin > reset voltage	-	1	-	nA

6.4.6 Temperature Sensor Specification

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Operation Voltage[1]		2.5	-	5.5	V
Operation Temperature		-40	-	85	℃
Current Consumption		6.4	-	10.5	μA
Gain			-1.76		mV/℃
Offset Voltage	Temp=0 ℃		720		mV

Note: Internal operation voltage comes from internal LDO.

6.4.7 Comparator Specification

PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
Operation Voltage AV_{DD}	-	2.5		5.5	V
Operation Temperature	-	-40	25	85	℃
Operation Current	V_{DD}=3.0 V	-	20	40	μA
Input Offset Voltage	-	-	5	15	mV
Output Swing	-	0.1	-	V_{DDA}-0.1	V
Input Common Mode Range	-	0.1	-	V_{DDA}-1.2	V
DC Gain	-	-	70	-	dB
Propagation Delay	VCM = 1.2 V and VDIFF = 0.1 V	-	200	-	ns
Comparison Voltage	20 mV at VCM=1 V 50 mV at VCM=0.1 V 50 mV at VCM=V_{DD}-1.2 10 mV for non-hysteresis	10	20	-	mV
Hysteresis	VCM=0.4 V ~ V_{DD}-1.2 V	-	±10	-	mV
Wake-up Time	CINP = 1.3 V CINN = 1.2 V	-	-	2	μs

6.4.8 USB PHY Specification

6.4.8.1 *USB DC Electrical Characteristics*

SYMBOL	PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
V_{IH}	Input High (driven)		2.0			V
V_{IL}	Input Low				0.8	V
V_{DI}	Differential Input Sensitivity	|PADP-PADM|	0.2			V
V_{CM}	Differential Common-mode Range	Includes V_{DI} range	0.8		2.5	V
V_{SE}	Single-ended Receiver Threshold		0.8		2.0	V
	Receiver Hysteresis			200		mV
V_{OL}	Output Low (driven)		0		0.3	V
V_{OH}	Output High (driven)		2.8		3.6	V
V_{CRS}	Output Signal Cross Voltage		1.3		2.0	V
R_{PU}	Pull-up Resistor		1.425		1.575	kΩ
V_{TRM}	Termination Voltage for Upstream Port Pull-up (RPU)		3.0		3.6	V
Z_{DRV}	Driver Output Resistance	Steady state drive*		10		Ω
C_{IN}	Transceiver Capacitance	Pin to GND			20	pF

*Driver output resistance doesn't include series resistor resistance.

6.4.8.2 *USB Full-Speed Driver Electrical Characteristics*

SYMBOL	PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
T_{FR}	Rise Time	C_L=50p	4		20	ns
T_{FF}	Fall Time	C_L=50p	4		20	ns
T_{FRFF}	Rise and Fall Time Matching	$T_{FRFF}=T_{FR}/T_{FF}$	90		111.11	%

6.4.8.3 *USB Power Dissipation*

SYMBOL	PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
I_{VBUS}	USB_VBUS Current (Steady State)	Standby		50		µA

6.4.8.4 *USB LDO Specification*

SYMBOL	PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
USB_VBUS	USB_VBUS Pin Input Voltage		4.0	5.0	5.5	V
USB_VDD33_CAP	LDO Output Voltage		3.0	3.3	3.6	V
C_{bp}	External Bypass Capacitor			1.0	-	uF

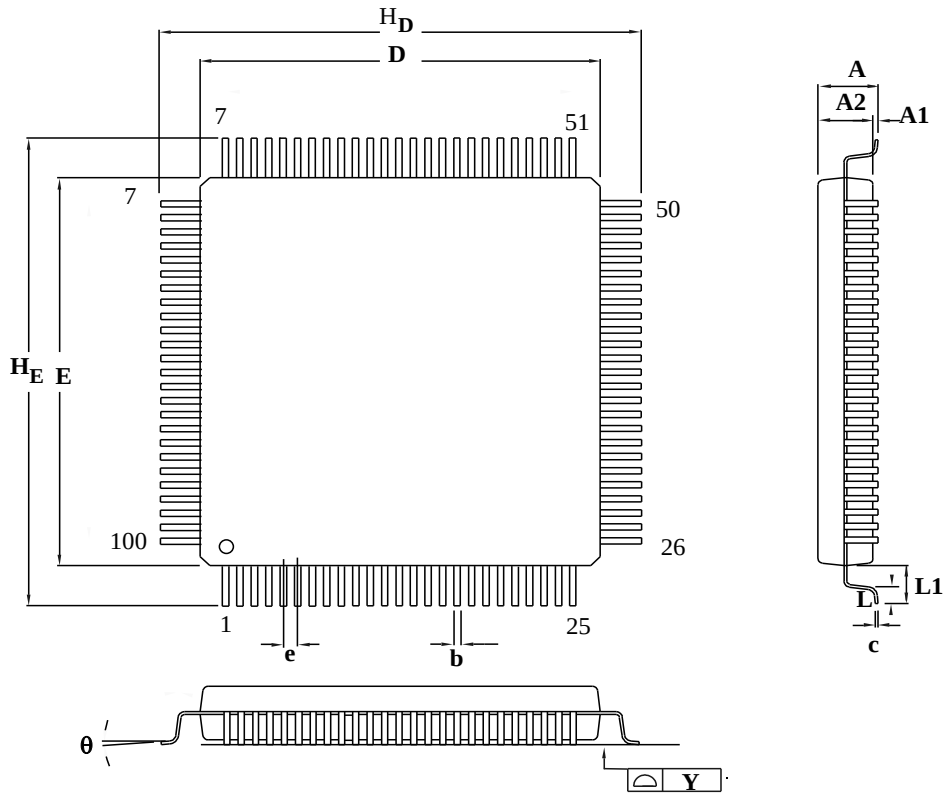
6.5 Flash DC Electrical Characteristics

SYMBOL	PARAMETER	CONDITION	MIN.	TYP.	MAX.	UNIT
V_{DD}	Supply Voltage		1.62	1.8	1.98	V[2]
N_{ENDUR}	Endurance		100000			cycles[1]
T_{RET}	Data Retention	At 85℃	10			year
T_{ERASE}	Page Erase Time			2		ms
T_{MER}	Mass Erase Time			10		ms
T_{PROG}	Program Time			20		μs
I_{DD1}	Read Current		-	0.15	0.5	mA/MHz
I_{DD2}	Program/Erase Current				7	mA
I_{PD}	Power Down Current		-	1	20	μA

1. Number of program/erase cycles.
2. V_{DD} is source from chip LDO output voltage.
3. This table is guaranteed by design, not test in production.

7 PACKAGE DIMENSIONS

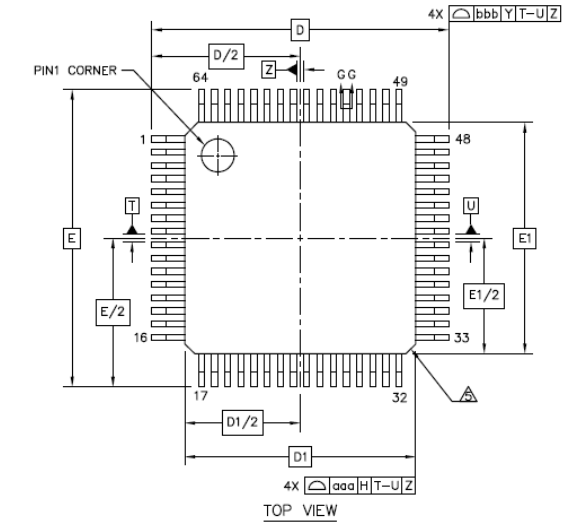
7.1 100-pin LQFP (14x14x1.4 mm footprint 2.0 mm)



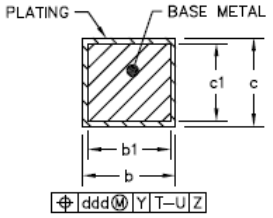
Controlling Dimension : Millimeters

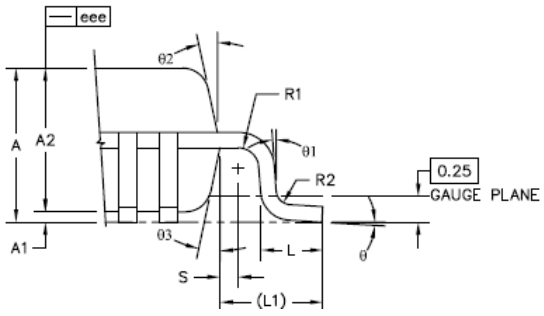
Symbol	Dimension in inch			Dimension in mm		
	Min	Nom	Max	Min	Nom	Max
A	—	—	0.063	—	—	1.60
A1	0.002	—	—	0.05	—	—
A2	0.053	0.055	0.057	1.35	1.40	1.45
b	0.007	0.009	0.011	0.17	0.22	0.27
c	0.004	0.006	0.008	0.10	0.15	0.20
D	0.547	0.551	0.556	13.90	14.00	14.10
E	0.547	0.551	0.556	13.90	14.00	14.10
e	—	0.020	—	—	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°

7.2 64-pin LQFP (7x7x1.4 mm footprint 2.0 mm)

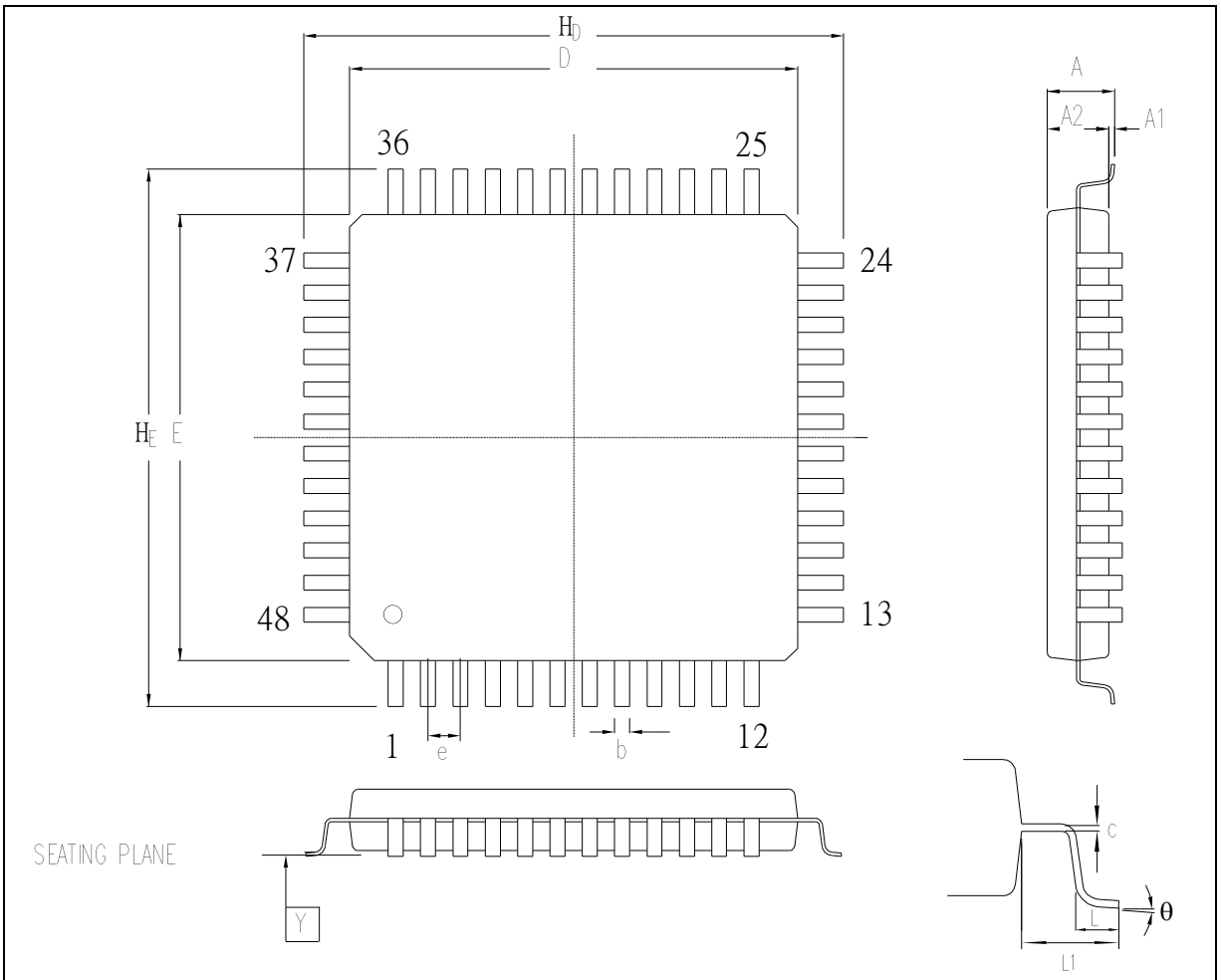


		SYMBOL	MIN	NOM	MAX
TOTAL THICKNESS		A	---	---	1.6
STAND OFF		A1	0.05	---	0.15
MOLD THICKNESS		A2	1.35	1.4	1.45
LEAD WIDTH(PLATING)		b	0.13	0.18	0.23
LEAD WIDTH		b1	0.13	0.16	0.19
L/F THICKNESS(PLATING)		c	0.09	---	0.2
L/F THICKNESS		c1	0.09	---	0.16
BODY SIZE	X	D	9 BSC		
	Y	E	9 BSC		
	X	D1	7 BSC		
	Y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





7.3 48-pin LQFP (7x7x1.4 mm footprint 2.0 mm)



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

8 REVISION HISTORY

REVISION	DATE	DESCRIPTION
V1.00	Dec 07, 2012	1. First version

Important Notice

Nuvoton Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, "Insecure Usage".

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer's risk, and in the event that third parties lay claims to Nuvoton as a result of customer's Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Nuvoton.