ARM® Cortex®-M4

32-bit Microcontroller

# NuMicro® Family
# NUC505 Series
# Technical Reference Manual

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

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

*All data and specifications are subject to change without notice.*

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

www.nuvoton.com

*TABLE OF CONTENTS*

## List of Figures

## List of Tables

# 1 GENERAL DESCRIPTION

The NuMicro® NUC505 series 32-bit microcontrollers are embedded with ARM® Cortex®-M4F core for consumer and industrial applications which need high computing power and rich communication interfaces.

The ARM® Cortex®-M4F core within NuMicro® NUC505 series can run up to 100 MHz and support DSP extensions and Floating Point Unit (FPU) function. The NuMicro® NUC505 series supports 128 Kbytes embedded SRAM with zero-wait state and 512 KB/ 2 Mbytes embedded SPI Flash memory, and is equipped with plenty of high performance peripheral devices, such as 24-bit Audio CODEC, USB2.0 High-speed Device, USB2.0 Full-speed Host, and other peripheral.

The NuMicro® NUC505 series is suitable for a wide range of applications such as:

- Audio and Wireless Audio Applications
- Thermal printerDid not find any incorrect format
- GPS Tracker / VTDR (Vehicle Travelling Data Recorder)
- Others high performance or data intensive computing applications

## Key Features:

- Core
  - ARM® Cortex®-M4F core running up to 100 MHz (with DSP and FPU)
- Memory
  - 128 KB of SRAM with zero-wait state
  - 512 KB/ 2 MB of SPI Flash
- Security for code protection
  - 128-bit key for code protection against pirating
  - Up to 15 times programming the key
- Clock Control
  - 12 MHz crystal oscillator input
  - Up to two PLLs for system clock and Audio
- Up to 12 Communication interfaces
  - USB 2.0 HS Device interface
  - Up to two USB 2.0 FS Host interfaces
  - Up to three UARTs
  - Up to three SPIs
  - Up to two I²C interfaces (up to 1 MHz)
  - SD Host
- GPIO
  - Supports up to 25/35/52 GPIOs for QFN88/LQFP64/LQFP48 respectively
- Timer
  - Supports four sets of 32-bit timers
  - Supports two watchdog timers (Independent and Window)
- RTC
  - Supports external power pin $V_{BAT}$
  - 32 bytes spare registers
  - Internal 32.768 kHz RC with calibration
- $I^2S$
  - Supports Master or Slave mode operation
  - Supports PCM mode A, PCM mode B, $I^2S$ and MSB justified data format
  - Supports DMA mode
- Audio CODEC
  - Embedded Stereo 24-bit Sigma-Delta CODEC
  - MIC/LINE-In-THDN: -80 dB, Dynamic Range SNR: 90 dB (A-Weighted)
  - Headphone Output-THDN:-60dB, Dynamic Range SNR: 93 dB (A-Weighted)
  - Sample Rate: 8 kHz to 96 kHz
- 12-bit ADC
  - Analog input voltage range: 0~ $AV_{DD}$
  - Supports single 12-bit SAR ADC conversion
  - Up to 8 channels
  - Up to 1 MSPS conversion with ADC_CH1, and up to 200 kSPS with other channels (except ADC_CH0).
- Built-in LDO with operating voltage 3.3V
- Low Voltage Detector (LVD)
  - With 2 levels: 2.8V / 2.6V
- Low Voltage Reset (LVR)
  - Threshold voltage level: 2.4 V
- Packages
  - LQFP48, LQFP64, QFN88
  - Temperature range: -40℃~+85℃

# 2 FEATURES

## 2.1 NUC505 Features

- Core
  - ARM® Cortex®-M4F core running up to 100 MHz
  - Supports DSP extension with hardware divider
  - Supports IEEE 754 compliant Floating Point Unit (FPU)
  - Supports Memory Protection Unit (MPU)
  - One 24-bit system timer
  - Supports Power-down mode by WFI and WFE instructions
  - Single-cycle 32-bit hardware multiplier
  - Supports programmable 16 level priorities of Nested Vectored Interrupt Controller (NVIC)
  - Supports programmable mask-able interrupts
  - Boots from SPI Flash Memory or USB Device
- SRAM Memory
  - 128 KB embedded SRAM with zero-wait state
  - Supports byte-, half-word- and word-access
- SPI Memory Interface Controller
  - Supports external SPI Flash memory
  - Supports code protection
  - Supports DMA mode for code transfer from SPI Flash memory to SRAM
  - Supports CPU direct read from SPI Flash memory.
  - Supports standard (1-bit), dual (2-bit), and quad (4-bit) I/O transfer mode
  - Supports general SPI master interface protocol
- Embedded SPI Flash
  - 512 KB/ 2 MB SPI Flash
  - Configurable program code/data allocation
  - Supports 2-wired ICP update through SWD/ICE interface
  - Supports ISP update
  - Supports standard (1-bit), dual (2-bit), and quad (4-bit) I/O transfer mode
  - Supports 100 MHz clock for standard I/O transfer mode
  - Supports 80 MHz clock for dual and quad I/O transfer mode
- Security for code protection
  - 128-bit key for code protection against pirating
  - Up to 15 times programming of the key
- Clock Control
  - Built-in 32.768 kHz internal low speed RC oscillator (LIRC) for RTC function,

  Watchdog timer and wake-up operation
  - Supports 32.768 kHz external low speed crystal oscillator (LXT) for RTC function and low-power system operation
  - Supports 12 MHz external high speed crystal oscillator (HXT) for precise timing operation
  - Supports one PLL up to 240 MHz for high performance system operation. The external high speed crystal oscillator (HXT) is used as the clock source for the PLL.

● $I^2S$
  - Supports Master or Slave mode operation
  - Internal PLL for frequency adjustment
  - Capable of handling 8-, 16-, 24- and 32-bit word sizes
  - Supports Mono and Stereo audio data
  - Supports PCM mode A, PCM mode B, $I^2S$ and MSB justified data format
  - Each provides two 16-word FIFO data buffers, one for transmitting and the other for receiving
  - Generates interrupt requests when buffer levels cross a programmable boundary
  - Supports DMA mode
  - Interface with internal or external audio CODEC

● Audio CODEC
  - Embedded Stereo 24-bit Sigma-Delta CODEC output
  - ADC-THDN: -80 dB, Dynamic Range SNR: 90 dB (A-Weighted)
  - Headphone Output-THDN:-60dB, Dynamic Range SNR: 93 dB (A-Weighted)
  - Sample Rate: 8 kHz to 96 kHz

● USB 2.0 High-speed device
  - 12 programmable endpoints for Control, Bulk IN/OUT, Interrupt and Isochronous transfers
  - 2K-byte buffer
  - Auto suspend function
  - Remote wake-up capability

● USB 2.0 Full-speed host
  - Fully compliant with USB revision 1.1 specification
  - Open Host Controller Interface (OHCI) revision 1.0 compatible
  - Full-speed (12Mbps) and Low-speed (1.5Mbps) device supported
  - Control, Bulk, Interrupt and Isochronous transfers supported

● SD Host Interface
  - Supports SD (Secure Digital) card and SD_HOST interface
  - Compliant with SD Memory Card Specification Version 2.0

- Supports 1 and 4-bit modes
- Supports 50 MHz to achieve 200 Mbps at 3.3V operation
- Supports DMA master

- Timer
  - Supports 4 sets of 32-bit timers with 24-bit up-timer and one 8-bit prescale counter
  - Independent clock source for each timer
  - Provides One-shot, periodic, toggle and continuous counting operation modes
  - Supports event counting function to count the event from external pin
  - Supports input capture function to capture or reset counter value
- Watchdog Timer
  - Supports multiple clock sources from LIRC (default selection), HXT and LXT
  - 8 selectable time-out period from 1.6ms ~ 26.0sec (depending on clock source)
  - Interrupt or reset selectable on watchdog time-out
- Window Watchdog Timer
  - Supports multiple clock sources from LIRC (default selection), HXT and LXT
  - Window set by 6-bit counter with 11-bit prescale
  - Interrupt or reset selectable on time-out
- GPIO
  - Four I/O modes
  - CMOS/Schmitt trigger input selectable
  - I/O pin configured as interrupt source with edge trigger setting
  - Supports 5V-tolerance function (except PA.7~PA.0 and PD.4~PD.2 only support 3.3 V)
  - Supports up to 52/35(34)/25(18) GPIOs for QFN88/LQFP64/LQFP48 respectively
- UART
  - Supports up to three UARTs – UART0, UART1 and UART2
  - Supports 16-byte FIFOs with programmable level trigger with UART0
  - Supports 64-byte FIFOs with programmable level trigger with UART1 and UART2
  - Supports auto flow control (nCTS and nRTS) with UART1 and UART2
  - Supports IrDA (SIR) function
  - Supports RS-485 9-bit mode and direction control
  - UART1 and UART2 support LIN function
  - Programmable baud-rate generator up to 1/16 system clock
  - Supports nCTS and data wake-up function
- SPI
  - Supports two sets of SPI controller – SPI0 and SPI1
  - Supports Master or Slave mode operation

  - Supports 1-bit Transfer mode
  - Configurable bit length of a transfer word from 8 to 32-bit
  - Provides separate 8-level depth transmit and receive FIFO buffers
  - Supports MSB first or LSB first transfer sequence
  - Supports the byte reorder function
  - Supports Byte or Word Suspend mode
  - Supports 3-wired, no slave select signal, bi-direction interface
  - Supports up to 50 MHz

- I$^2$C
  - Supports up to two sets of I$^2$C devices
  - Supports Master/Slave mode
  - Bidirectional data transfer between masters and slaves
  - Multi-master bus (no central master)
  - Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
  - Serial clock synchronization allows devices with different bit rates to communicate via one serial bus
  - Serial clock synchronization can be used as a handshake mechanism to suspend and resume serial transfer
  - Programmable clocks allow versatile rate control
  - Supports multiple address recognition (four slave address with mask option)
  - Supports SMBus and PMBus
  - Supports speed up to 1Mbps
  - Supports multi-address Power-down wake-up function
- PWM
  - Four 16-bit timers
  - Programmable duty control of output waveform (PWM)
  - Auto reload mode or one-shot pulse mode
  - Capture and compare function
- RTC
  - Supports external power pin RTC_VDD33
  - Supports 32.768 kHz crystal oscillation circuit
  - Supports RTC counter (second, minute, hour) and calendar counter (day, month, year)
  - Supports Alarm registers (second, minute, hour, day, month, year)
  - Supports 32 bytes spare registers
  - Wake up from Deep Power-down mode or from Power-down mode
  - Supports wake up from Power-down mode by input pin

  - Supports chip Power-off by register setting
  - Supports Power-on time-out for low battery protection
- Analog to Digital Converter
  - Analog input voltage range: 0~ $AV_{DD}$
  - Supports single 12-bit SAR ADC conversion
  - 12-bit resolution and 10-bit accuracy is guaranteed
  - Up to 1MSPS conversion with ADC_CH1, and up to 200 kSPS with others (except ADC_CH0).
  - Up to 8 external single-ended analog input channels
  - Supports single ADC interrupt
  - An A/D conversion can be triggered by software control
- Built-in LDO with operating voltage 3.3V
- Low Voltage Detector (LVD)
  - With 2 levels: 2.8V / 2.6V
- Low Voltage Reset (LVR)
  - Threshold voltage level: 2.4 V
- Power Management
  - Advanced power management including Deep Power-down, Power-down, Idle and Normal Operating modes
  - Normal Operating mode
    - CPU runs normally and all clocks on; the current consumption is around 46 mA (at 96 MHz CPU clock)
  - Idle mode
    - CPU clock stop, and all other clocks on
  - Power-down mode
    - All clocks stop, except LXT and LIRC, with SRAM retention; the current consumption is around 700 uA
  - Deep Power-down mode
    - All clocks stop, except LXT and LIRC, without SRAM retention; the current consumption is around 7 uA
- Operating Temperature: -40℃~+85℃
- Packages
  - All Green package (RoHS)
  - QFN 88-pin (10mm x 10mm)
  - LQFP 64-pin (7mm x 7mm)
  - LQFP 48-pin (7mm x 7mm)
  - QFN 48-pin (7mm x 7mm)

# 3 ABBREVIATIONS

## 3.1 Abbreviations

| Acronym | Description |
|---|---|
| ADC | Analog-to-Digital Converter |
| APB | Advanced Peripheral Bus |
| AHB | Advanced High-Performance Bus |
| DMA | Direct Memory Access |
| FIFO | First In, First Out |
| FPU | Floating Point Unit |
| GPIO | General-Purpose Input/Output |
| HCLK | The Clock of Advanced High-Performance Bus |
| HXT | 12 MHz External High Speed Crystal Oscillator |
| ICP | In Circuit Programming |
| ISP | In System Programming |
| LDO | Low Dropout Regulator |
| LIN | Local Interconnect Network |
| LIRC | 32.768 kHz Internal Low Speed RC Oscillator |
| LXT | 32.768 kHz External Low Speed Crystal Oscillator |
| LVD | Low Voltage Detection |
| MPU | Memory Protection Unit |
| NVIC | Nested Vectored Interrupt Controller |
| PCLK | The Clock of Advanced Peripheral Bus |
| PLL | Phase-Locked Loop |
| PWM | Pulse Width Modulation |
| SD | Secure Digital |
| SPI | Serial Peripheral Interface |
| SPIM | Serial Master Interface Controller |
| SPS | Samples per Second |
| TMR | Timer Controller |
| UART | Universal Asynchronous Receiver/Transmitter |
| USB | Universal Serial Bus |
| WDT | Watchdog Timer |
| WWDT | Window Watchdog Timer |

Table 3.1-1 List of Abbreviations

# 4 PARTS INFORMATION LIST AND PIN CONFIGURATION

## 4.1 Selection Guide

### 4.1.1 NuMicro® NUC505 Base Series Selection Guide

[1]: *√ marked in the table means that only NUC505DS13Y supports Headphone Out.

[2]: The packages are not pin-to-pin compatible even though they are the same packages. LQFP64*: 7x7mm

| Part Number | Serial Flash (KB) | SRAM (KB) | ISP ROM (KB) | I/O | Timer (32-Bit) | Connectivity | | | | I$^2$S | USB 2.0 HS Device | USB 2.0 FS Host | PWM (16-Bit) | 24-Bit Audio CODEC[1] | DIGITAL MIC | ADC (12-Bit) | RTC | ISP/ICP | Package[2] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | I$^2$C | SD HOST | SPI | UART | | | | | | | | | | |
| NUC505DLA | 512 | 128 | 8 | 18 | 4 | 2 | - | 2 | 2 | 1 | 1 | - | - | √ | 1 | 5CH | - | √ | LQFP48 |
| NUC505DL13Y | 2048 | 128 | 8 | 25 | 4 | 2 | 1 | 3 | 3 | 1 | 1 | 1 | 4 | - | 1 | 5CH | √ | √ | LQFP48 |
| NUC505YLA | 512 | 128 | 8 | 18 | 4 | 2 | - | 2 | 2 | 1 | 1 | - | - | √ | 1 | 5CH | - | √ | QFN48 |
| NUC505YLA2Y | 512 | 128 | 8 | 25 | 4 | 2 | 1 | 3 | 3 | 1 | 1 | 1 | 4 | - | 1 | 5CH | √ | √ | QFN48 |
| NUC505DSA | 512 | 128 | 8 | 34 | 4 | 2 | 1 | 3 | 3 | 1 | 1 | 1 | 4 | √ | 1 | 5CH | - | √ | LQFP64* |
| NUC505DS13Y | 2048 | 128 | 8 | 35 | 4 | 2 | 1 | 3 | 3 | 1 | 1 | 1 | 4 | √* | 1 | 8CH | √ | √ | LQFP64* |
| NUC505YO13Y | 2048 | 128 | 8 | 52 | 4 | 2 | 1 | 3 | 3 | 1 | 1 | 2 | 4 | √ | 1 | 8CH | √ | √ | QFN88 |

Table 4.1-1 NuMicro® NUC505 Base Series Selection Guide

### 4.1.2 NuMicro® NUC505 Base Series Naming Rule

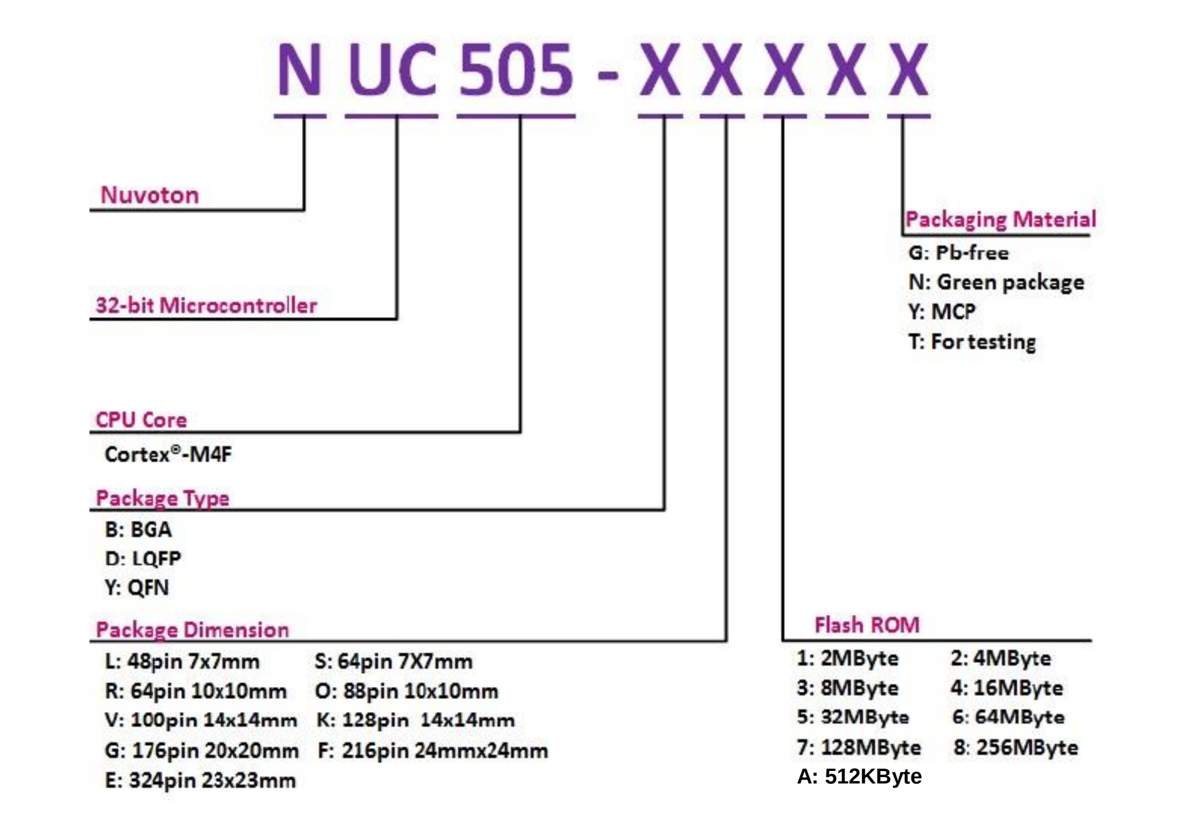


Figure 4.1-1 NuMicro® NUC505 Base Series Selection Code

## 4.2 Pin Configuration

### 4.2.1 NuMicro® NUC505DLA LQFP 48-pin

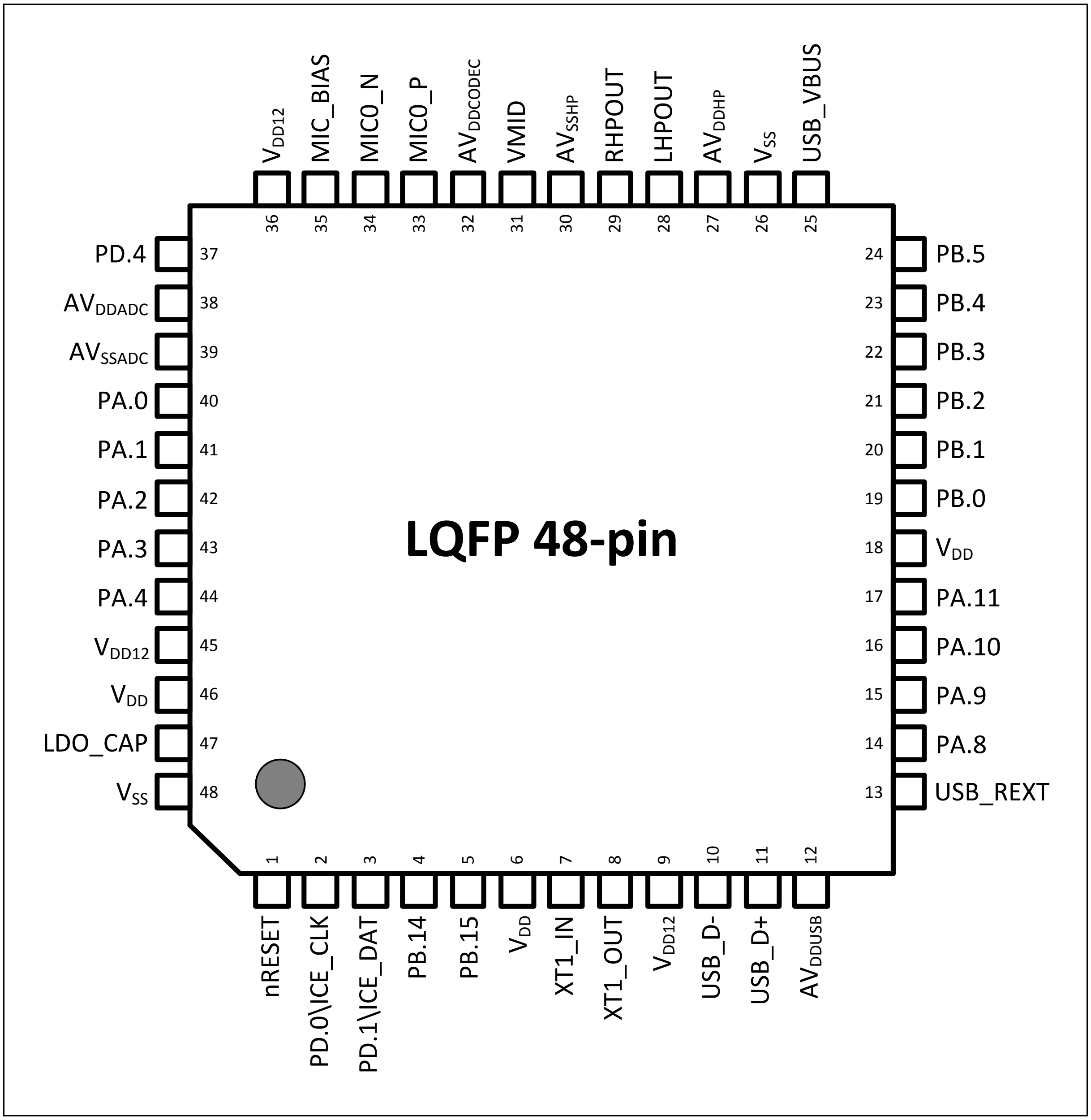


Figure 4.2-1 NuMicro® NUC505DLA LQFP 48-pin Diagram

### 4.2.2 NuMicro® NUC505DL13Y LQFP 48-pin

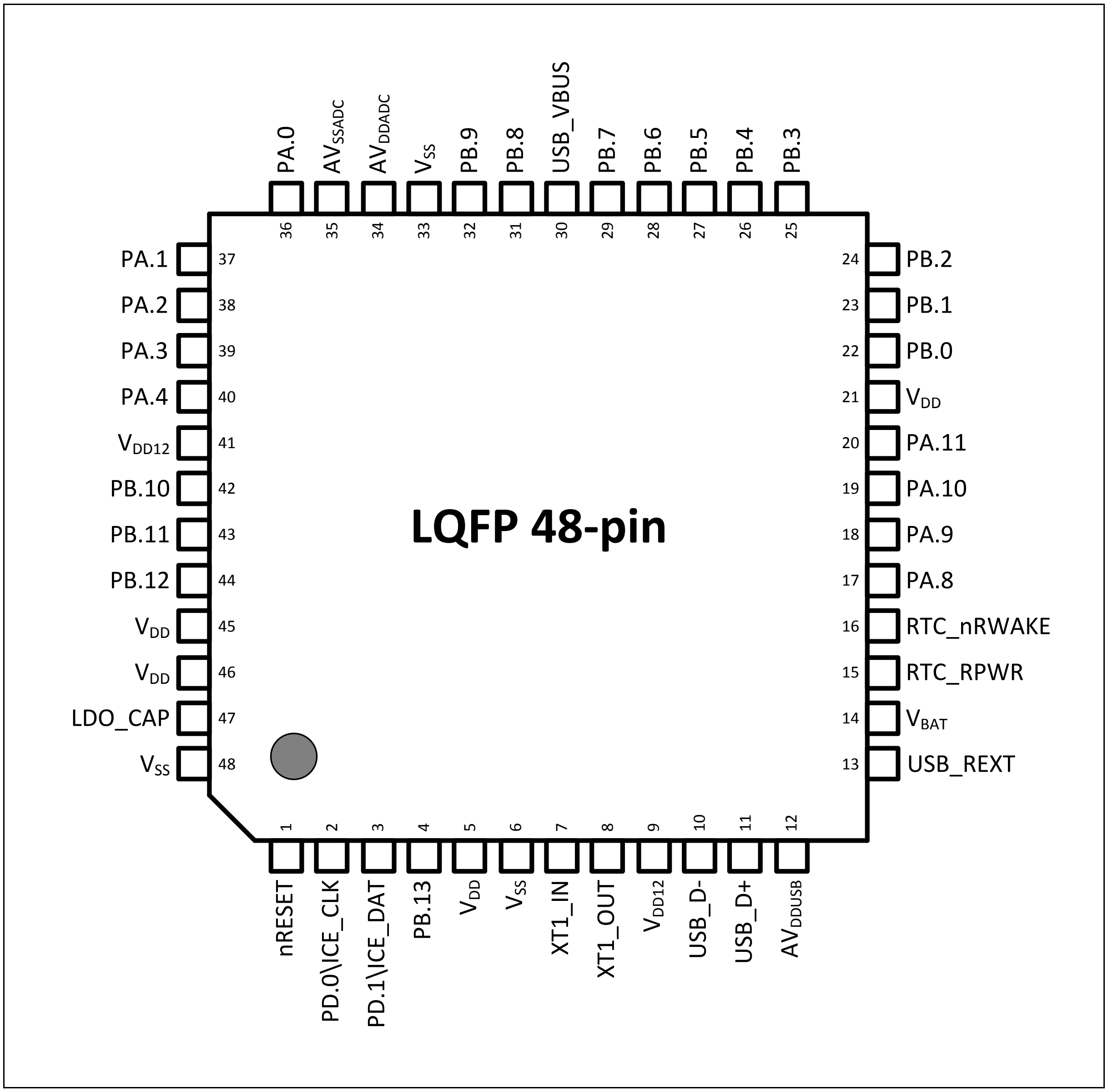
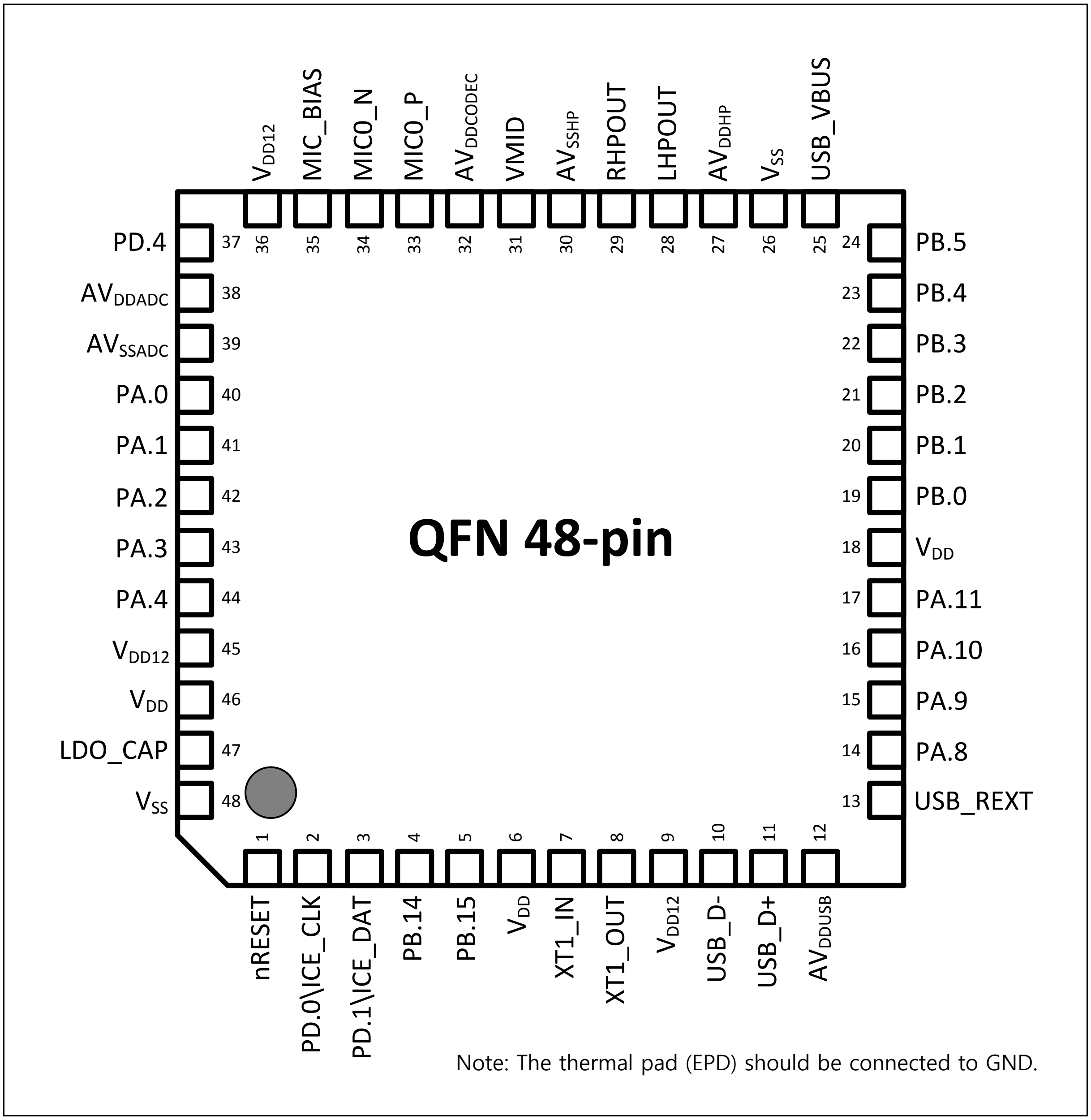


Figure 4.2-2 NuMicro® NUC505DL13Y LQFP 48-pin Diagram

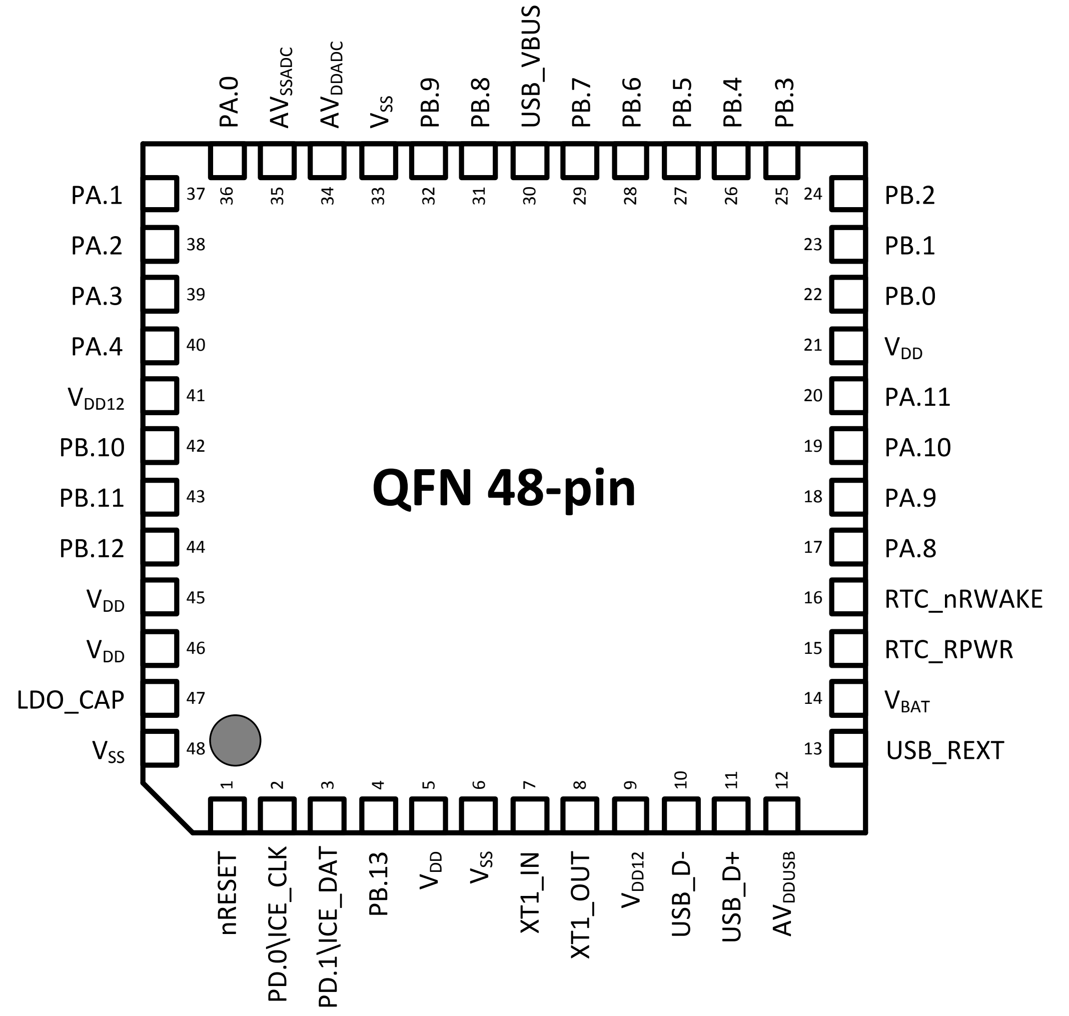
### 4.2.3 NuMicro® NUC505YLA QFN 48-pin



Note: The thermal pad (EPD) should be connected to GND.

Figure 4.2-3 NuMicro® NUC505YLA QFN 48-pin Diagram

### 4.2.4 NuMicro® NUC505YLA2Y QFN 48-pin



Note: The thermal pad (EPD) should be connected to GND.

Figure 4.2-4 NuMicro® NUC505YLA2Y QFN 48-pin Diagram

### 4.2.5 NuMicro® NUC505DSA LQFP 64-pin

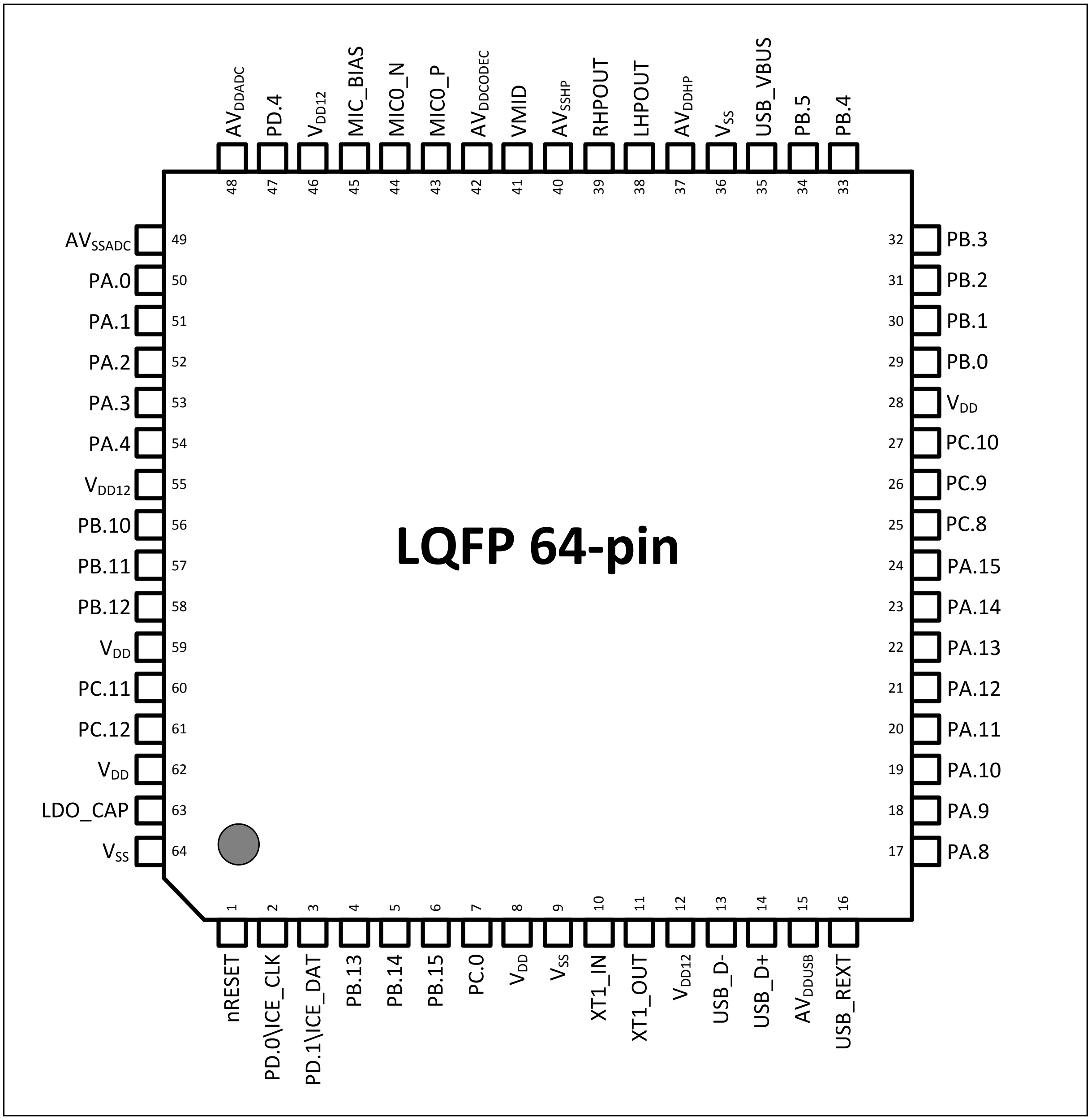


Figure 4.2-5 NuMicro® NUC505DSA LQFP 64-pin Diagram

### 4.2.6 NuMicro® NUC505DS13Y LQFP 64-pn

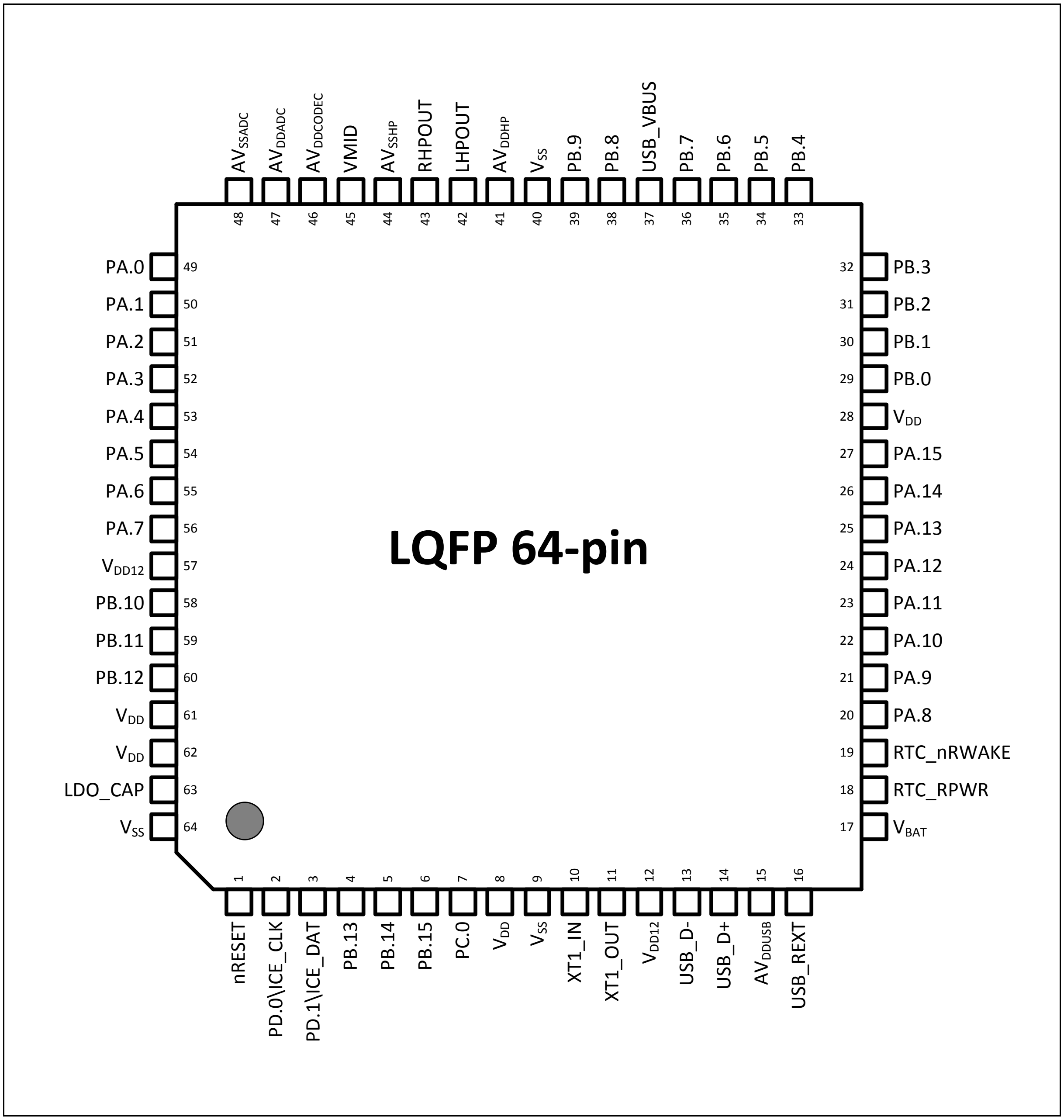
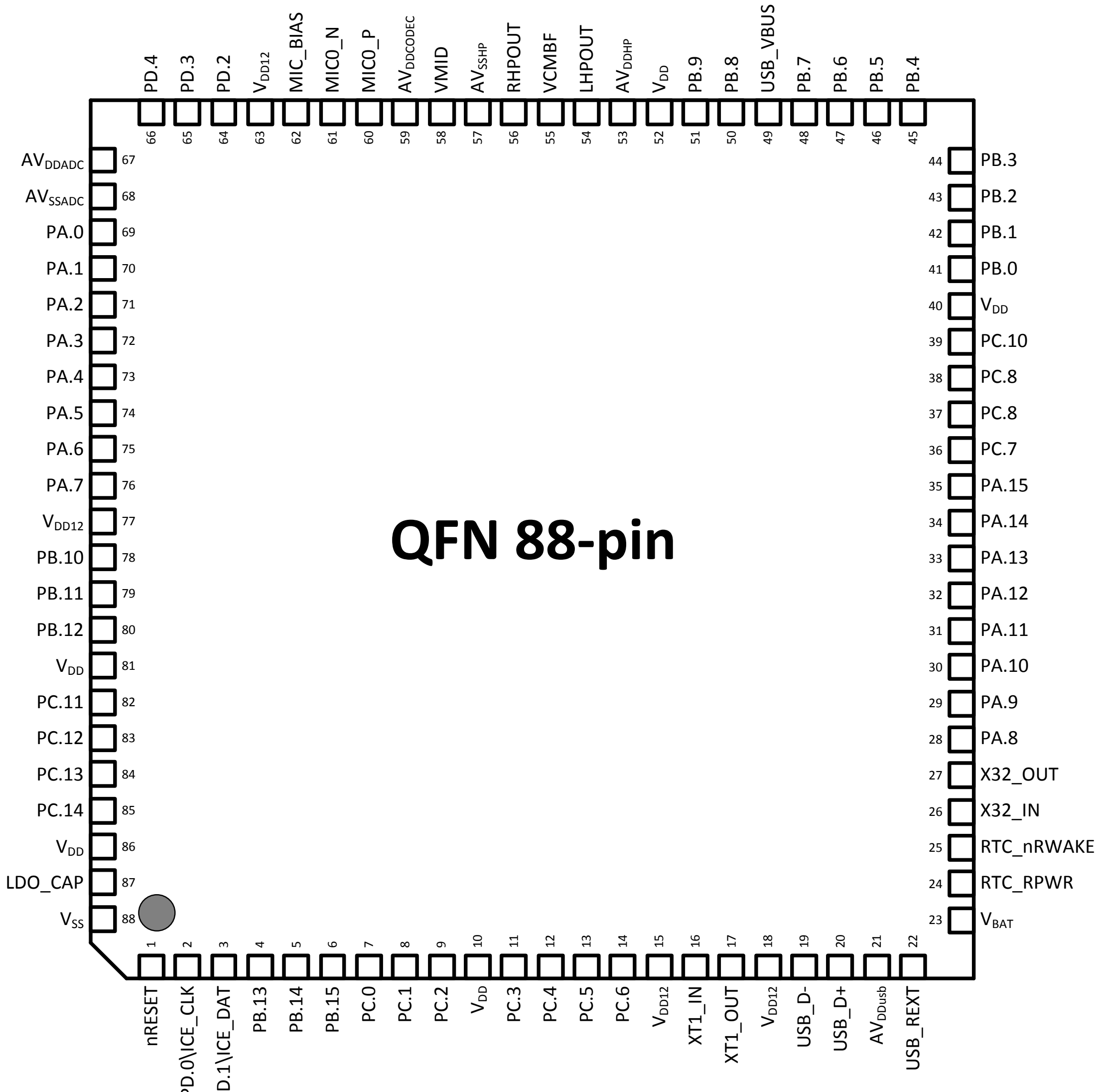


Figure 4.2-6 NuMicro® NUC505DS13Y LQFP 64-pin Diagram

### 4.2.7 NuMicro® NUC505YO13Y QFN 88-pin



Note: The thermal pad (EPD) should be connected to GND.

Figure 4.2-7 NuMicro® NUC505YO13Y QFN 88-pin Diagram

## 4.3 Pin Description

### 4.3.1 NuMicro® NUC505DLA LQFP 48-pin Description

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

PA.0 MFP0 means SYS_GPA_MFPL[2:0]=0x0.

PA.9 MFP5 means SYS_GPA_MFPH[6:4]=0x5.

| Pin No. | Pin Name | Type | MFP* | Description |
|---|---|---|---|---|
| 1 | nRESET | I | MFP0 | External reset input: active LOW, with an internal pull-up. Set this pin low reset to initial state. |
| 2 | ICE_CLK | O | MFP0 | Serial wired debugger clock pin. (In ICE mode) |
| | PD.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SCL | O | MFP2 | $I^2C0$ clock pin. |
| 3 | ICE_DAT | I/O | MFP0 | Serial wired debugger data pin. (In ICE mode) |
| | PD.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SDA | I/O | MFP2 | $I^2C0$ data input/output pin. |
| 4 | PB.14 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH1_D+ | I/O | MFP1 | USB host-lite 1differential signal D+. |
| | I2C1_SCL | O | MFP2 | I2C1 clock pin. |
| 5 | PB.15 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH1_D- | I/O | MFP1 | USB host-lite 1 differential signal D-. |
| | I2C1_SDA | I/O | MFP2 | I2C1 data input/output pin. |
| 6 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 7 | XT1_IN | I | MFP0 | External 12 MHz (high speed) crystal input pin. |
| 8 | XT1_OUT | O | MFP0 | External 12 MHz (high speed) crystal output pin. |
| 9 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 10 | USB_D- | A | MFP0 | USB differential signal D-. |
| 11 | USB_D+ | A | MFP0 | USB differential signal D+. |
| 12 | $AV_{DDUSB}$ | A | MFP0 | Power supply for analog USB, DC 3.3V. |
| 13 | USB_REXT | A | MFP0 | 12.1 KΩ used internally for USB circuitry. |

| | | | | |
|---|---|---|---|---|
| 14 | PA.8 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_SS | O | MFP1 | SPIM slave select pin. |
| | I2S_LRCLK | I/O | MFP2 | $I^2S$ left right channel clock. |
| | UART1_TXD | O | MFP3 | Data transmitter output pin for UART1. |
| 15 | PA.9 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_CLK | O | MFP1 | SPIM serial clock pin. |
| | I2S_BCLK | I/O | MFP2 | $I^2S$ bit clock pin. |
| | UART1_RXD | I | MFP3 | Data receiver input pin for UART1. |
| | SYSCFG[0] | I | MFP0 | System configuration setting bit 0. |
| 16 | PA.10 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_MOSI | I/O | MFP1 | SPIM MOSI (Master Out, Slave In) pin. |
| | I2C1_SCL | O | MFP2 | I2C1 clock pin. |
| | SD_CLK | O | MFP4 | SD/SDH mode - clock. |
| | SYSCFG[1] | I | MFP0 | System configuration setting bit 1. |
| 17 | PA.11 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_MISO | I/O | MFP1 | SPIM MISO (Master In, Slave Out) pin. |
| | I2C1_SDA | I/O | MFP2 | I2C1 data input/output pin. |
| | SD_CMD | I | MFP4 | SD/SDH mode – command/response. |
| 18 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 19 | PB.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SCL | O | MFP2 | I2C0 clock pin. |
| | UART0_TXD | O | MFP3 | Data transmitter output pin for UART0. |
| | SD_DAT2 | I/O | MFP4 | SD/SDH mode data line bit 2. |
| 20 | PB.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SDA | I/O | MFP2 | I2C0 data input/output pin. |
| | UART0_RXD | I | MFP3 | Data receiver input pin for UART0. |
| | SD_DAT3 | I/O | MFP4 | SD/SDH mode data line bit 3. |

| | | | | |
|---|---|---|---|---|
| 21 | PB.2 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_SS | O | MFP1 | SPI0 slave select pin. |
| | SD_CMD | I | MFP4 | SD/SDH mode – command/response. |
| 22 | PB.3 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_CLK | O | MFP1 | SPI0 serial clock pin. |
| | SD_CLK | O | MFP4 | SD/SDH mode – clock. |
| | SYSCFG[2] | I | MFP0 | System configuration setting bit 2. |
| 23 | PB.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_MOSI | O | MFP1 | SPI0 MOSI (Master Out, Slave In) pin. |
| | SYSCFG[3] | I | MFP0 | System configuration setting bit 3. |
| 24 | PB.5 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_MISO | I | MFP1 | SPI0 MISO (Master In, Slave Out) pin. |
| | SD_nCD | I | MFP4 | SD/SDH mode – card detect. |
| 25 | USB_VBUS33 | I | MFP0 | Detects whether USB is plug-in. |
| 26 | $V_{SS}$ | A | MFP0 | Ground |
| 27 | $AV_{DDHP}$ | A | MFP0 | Power supply for analog CODEC headphone, DC 3.3V. |
| 28 | LHPOUT | A | MFP0 | Headphone left channel output pin. |
| 29 | RHPOUT | A | MFP0 | Headphone right channel output pin. |
| 30 | $AV_{SSHP}$ | A | MFP0 | Ground for analog CODEC headphone. |
| 31 | VMID | A | MFP0 | Headphone reference power. |
| 32 | $AV_{DDCODEC}$ | A | MFP0 | Power supply for analog CODEC, DC 3.3V. |
| 33 | MIC0_P | A | MFP0 | Microphone 0 positive input. |
| 34 | MIC0_N | A | MFP0 | Microphone 0 negative input. |
| 35 | MIC_BIAS | A | MFP0 | CODEC left line-in channel or Microphone bias. |
| 36 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 37 | PD.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | RLINEIN | A | MFP1 | CODEC right line-in channel. |

NUC505 SERIES TECHNICAL REFERENCE MANUAL

| | | | | |
|---|---|---|---|---|
| 38 | $AV_{DDADC}$ | A | MFP0 | Power supply for analog SAR-ADC, DC 3.3V. |
| 39 | $AV_{SSADC}$ | A | MFP0 | Ground pin for analog SAR-ADC. |
| 40 | PA.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH0 | A | MFP1 | ADC channel 0 analog input. |
| 41 | PA.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH1 | A | MFP1 | ADC channel 1 analog input. |
| 42 | PA.2 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH2 | A | MFP1 | ADC channel 2 analog input. |
| | I2S_MCLK | O | MFP2 | $I^2S$ master clock output pin. |
| 43 | PA.3 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH3 | A | MFP1 | ADC channel 3 analog input. |
| | I2S_DI | I | MFP2 | $I^2S$ data input. |
| 44 | PA.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH4 | A | MFP1 | ADC channel 4 analog input. |
| | I2S_DO | O | MFP2 | $I^2S$ data output. |
| 45 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 46 | $V_{DD}$ | A | MFP0 | Power supply, DC 3.3V. |
| 47 | LDO_CAP | A | MFP0 | LDO output pin. |
| 48 | $V_{SS}$ | A | MFP0 | Ground. |

### 4.3.2 NuMicro® NUC505DL13Y LQFP 48-pin Description

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

PA.0 MFP0 means SYS_GPA_MFPL[2:0]=0x0.

PA.9 MFP5 means SYS_GPA_MFPH[6:4]=0x5.

| Pin No. | Pin Name | Type | MFP* | Description |
|---|---|---|---|---|
| 1 | nRESET | I | MFP0 | External reset input: active LOW, with an internal pull-up. Set this pin low reset to initial state. |
| 2 | ICE_CLK | O | MFP0 | Serial wired debugger clock pin. (In ICE mode) |
| | PD.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SCL | O | MFP2 | $I^2C0$ clock pin. |
| 3 | ICE_DAT | I/O | MFP0 | Serial wired debugger data pin. (In ICE mode) |
| | PD.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SDA | I/O | MFP2 | $I^2C0$ data input/output pin. |
| 4 | PB.13 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_MISO | I | MFP1 | SPI1 MISO (Master In, Slave Out) pin. |
| | USBH1_D- | I/O | MFP2 | USB host-lite 1 differential signal D-. |
| | UART2_nRTS | O | MFP3 | Request to Send output pin for UART2. |
| | PWM_CH3 | I/O | MFP4 | PWM channel3 output/capture input. |
| 5 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 6 | $V_{SS}$ | A | MFP0 | Ground. |
| 7 | XT1_IN | I | MFP0 | External 12 MHz (high speed) crystal input pin. |
| 8 | XT1_OUT | O | MFP0 | External 12 MHz (high speed) crystal output pin. |
| 9 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 10 | USB_D- | A | MFP0 | USB differential signal D-. |
| 11 | USB_D+ | A | MFP0 | USB differential signal D+. |
| 12 | $AV_{DDUSB}$ | A | MFP0 | Power supply for analog USB, DC 3.3V. |
| 13 | USB_REXT | A | MFP0 | 12.1 KΩ used internally for USB circuitry. |
| 14 | $V_{BAT}$ | A | MFP0 | Power supply by batteries for RTC, DC 3.3V. |

| | | | | |
|---|---|---|---|---|
| 15 | RTC_RPWR | O | MFP0 | Enable external power control source when active high. |
| 16 | RTC_nRWAKE | I | MFP0 | System power enable trigger when active low. |
| 17 | PA.8 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_SS | O | MFP1 | SPIM slave select pin. |
| | I2S_LRCLK | I/O | MFP2 | I$^2$S left right channel clock. |
| | UART1_TXD | O | MFP3 | Data transmitter output pin for UART1. |
| 18 | PA.9 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_CLK | O | MFP1 | SPIM serial clock pin. |
| | I2S_BCLK | I/O | MFP2 | I$^2$S bit clock pin. |
| | UART1_RXD | I | MFP3 | Data receiver input pin for UART1. |
| | SYSCFG[0] | I | MFP0 | System configuration setting bit 0. |
| 19 | PA.10 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_MOSI | I/O | MFP1 | SPIM MOSI (Master Out, Slave In) pin. |
| | I2C1_SCL | O | MFP2 | I2C1 clock pin. |
| | SD_CLK | O | MFP4 | SD/SDH mode - clock. |
| | SYSCFG[1] | I | MFP0 | System configuration setting bit 1. |
| 20 | PA.11 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_MISO | I/O | MFP1 | SPIM MISO (Master In, Slave Out) pin. |
| | I2C1_SDA | I/O | MFP2 | I2C1 data input/output pin. |
| | SD_CMD | I | MFP4 | SD/SDH mode – command/response. |
| 21 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 22 | PB.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SCL | O | MFP2 | I2C0 clock pin. |
| | UART0_TXD | O | MFP3 | Data transmitter output pin for UART0. |
| | SD_DAT2 | I/O | MFP4 | SD/SDH mode data line bit 2. |
| 23 | PB.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SDA | I/O | MFP2 | I2C0 data input/output pin. |

| | | | | |
|---|---|---|---|---|
| | UART0_RXD | I | MFP3 | Data receiver input pin for UART0. |
| | SD_DAT3 | I/O | MFP4 | SD/SDH mode data line bit 3. |
| 24 | PB.2 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_SS | O | MFP1 | SPI0 slave select pin. |
| | SD_CMD | I | MFP4 | SD/SDH mode – command/response. |
| 25 | PB.3 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_CLK | O | MFP1 | SPI0 serial clock pin. |
| | SD_CLK | O | MFP4 | SD/SDH mode – clock. |
| | SYSCFG[2] | I | MFP0 | System configuration setting bit 2. |
| 26 | PB.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_MOSI | O | MFP1 | SPI0 MOSI (Master Out, Slave In) pin. |
| | SYSCFG[3] | I | MFP0 | System configuration setting bit 3. |
| 27 | PB.5 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_MISO | I | MFP1 | SPI0 MISO (Master In, Slave Out) pin. |
| | SD_nCD | I | MFP4 | SD/SDH mode – card detect. |
| 28 | PB.6 | I/O | MFP0 | General purpose digital I/O pin. |
| | UART1_TXD | O | MFP3 | Data transmitter output pin for UART1. |
| | SD_DAT0 | I/O | MFP4 | SD/SDH mode data line bit 0. |
| 29 | PB.7 | I/O | MFP0 | General purpose digital I/O pin. |
| | UART1_RXD | I | MFP3 | Data receiver input pin for UART1. |
| | SD_DAT1 | I/O | MFP4 | SD/SDH mode data line bit 1. |
| 30 | USB_VBUS33 | I | MFP0 | Detects whether USB is plug-in. |
| 31 | PB.8 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH_PWEN | O | MFP1 | USB host mode to control an external overcurrent source. |
| | TM1_CNT_OUT | I/O | MFP2 | Timer1 event counter input/toggle output. |
| | UART1_nCTS | I | MFP3 | Clear to Send input pin for UART1. |
| | SD_DAT2 | I/O | MFP4 | SD/SDH mode data line bit 2. |

| | | | | |
|---|---|---|---|---|
| 32 | PB.9 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH_OVD | I | MFP1 | USB host bus power over voltage detector. |
| | TM1_EXT | I | MFP2 | Timer1 external capture input. |
| | UART1_nRTS | O | MFP3 | Request to Send output pin for UART1. |
| | SD_DAT3 | I/O | MFP4 | SD/SDH mode data line bit 3. |
| 33 | $V_{SS}$ | A | MFP0 | Ground |
| 34 | $AV_{DDADC}$ | A | MFP0 | Power supply for analog SAR-ADC, DC 3.3V. |
| 35 | $AV_{SSADC}$ | A | MFP0 | Ground pin for analog SAR-ADC. |
| 36 | PA.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH0 | A | MFP1 | ADC channel 0 analog input. |
| 37 | PA.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH1 | A | MFP1 | ADC channel 1 analog input. |
| 38 | PA.2 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH2 | A | MFP1 | ADC channel 2 analog input. |
| | I2S_MCLK | O | MFP2 | I$^2$S master clock output pin. |
| 39 | PA.3 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH3 | A | MFP1 | ADC channel 3 analog input. |
| | I2S_DI | I | MFP2 | I$^2$S data input. |
| 40 | PA.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH4 | A | MFP1 | ADC channel 4 analog input. |
| | I2S_DO | O | MFP2 | I$^2$S data output. |
| 41 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 42 | PB.10 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_SS | O | MFP1 | SPI1 slave select pin. |
| | I2C1_SCL | O | MFP2 | I2C1 clock pin. |
| | UART2_TXD | O | MFP3 | Data transmitter output pin for UART2. |
| | PWM_CH0 | I/O | MFP4 | PWM channel0 output/capture input. |

|  |  |  |  |  |
|---|---|---|---|---|
| 43 | PB.11 | I/O | MFP0 | General purpose digital I/O pin. |
|  | SPI1_CLK | O | MFP1 | SPI1 serial clock pin. |
|  | I2C1_SDA | I/O | MFP2 | I2C1 data input/output pin. |
|  | UART2_RXD | I | MFP3 | Data receiver input pin for UART2. |
|  | PWM_CH1 | I/O | MFP4 | PWM channel1 output/capture input. |
| 44 | PB.12 | I/O | MFP0 | General purpose digital I/O pin. |
|  | SPI1_MOSI | O | MFP1 | SPI1 MOSI (Master Out, Slave In) pin. |
|  | USBH1_D+ | I/O | MFP2 | USB host-lite 1 differential signal D+ |
|  | UART2_nCTS | I | MFP3 | Clear to send input pin for UART2. |
|  | PWM_CH2 | I/O | MFP4 | PWM channel2 output/capture input. |
| 45 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 46 | $V_{DD}$ | A | MFP0 | Power supply, DC 3.3V. |
| 47 | LDO_CAP | A | MFP0 | LDO output pin. |
| 48 | $V_{SS}$ | A | MFP0 | Ground. |

### 4.3.3 NuMicro® NUC505YLA QFN 48-pin Description

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

PA.0 MFP0 means SYS_GPA_MFPL[2:0]=0x0.

PA.9 MFP5 means SYS_GPA_MFPH[6:4]=0x5.

| Pin No. | Pin Name | Type | MFP* | Description |
|---|---|---|---|---|
| 1 | nRESET | I | MFP0 | External reset input: active LOW, with an internal pull-up. Set this pin low reset to initial state. |
| 2 | ICE_CLK | O | MFP0 | Serial wired debugger clock pin. (In ICE mode) |
| | PD.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SCL | O | MFP2 | $I^2C0$ clock pin. |
| 3 | ICE_DAT | I/O | MFP0 | Serial wired debugger data pin. (In ICE mode) |
| | PD.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SDA | I/O | MFP2 | $I^2C0$ data input/output pin. |
| 4 | PB.14 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH1_D+ | I/O | MFP1 | USB host-lite 1differential signal D+. |
| | I2C1_SCL | O | MFP2 | I2C1 clock pin. |
| 5 | PB.15 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH1_D- | I/O | MFP1 | USB host-lite 1 differential signal D-. |
| | I2C1_SDA | I/O | MFP2 | I2C1 data input/output pin. |
| 6 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 7 | XT1_IN | I | MFP0 | External 12 MHz (high speed) crystal input pin. |
| 8 | XT1_OUT | O | MFP0 | External 12 MHz (high speed) crystal output pin. |
| 9 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 10 | USB_D- | A | MFP0 | USB differential signal D-. |
| 11 | USB_D+ | A | MFP0 | USB differential signal D+. |
| 12 | $AV_{DDUSB}$ | A | MFP0 | Power supply for analog USB, DC 3.3V. |
| 13 | USB_REXT | A | MFP0 | 12.1 KΩ used internally for USB circuitry. |
| 14 | PA.8 | I/O | MFP0 | General purpose digital I/O pin. |

| | | | | |
|---|---|---|---|---|
| | SPIM_SS | O | MFP1 | SPIM slave select pin. |
| | I2S_LRCLK | I/O | MFP2 | I$^2$S left right channel clock. |
| | UART1_TXD | O | MFP3 | Data transmitter output pin for UART1. |
| 15 | PA.9 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_CLK | O | MFP1 | SPIM serial clock pin. |
| | I2S_BCLK | I/O | MFP2 | I$^2$S bit clock pin. |
| | UART1_RXD | I | MFP3 | Data receiver input pin for UART1. |
| | SYSCFG[0] | I | MFP0 | System configuration setting bit 0. |
| 16 | PA.10 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_MOSI | I/O | MFP1 | SPIM MOSI (Master Out, Slave In) pin. |
| | I2C1_SCL | O | MFP2 | I2C1 clock pin. |
| | SD_CLK | O | MFP4 | SD/SDH mode - clock. |
| | SYSCFG[1] | I | MFP0 | System configuration setting bit 1. |
| 17 | PA.11 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_MISO | I/O | MFP1 | SPIM MISO (Master In, Slave Out) pin. |
| | I2C1_SDA | I/O | MFP2 | I2C1 data input/output pin. |
| | SD_CMD | I | MFP4 | SD/SDH mode – command/response. |
| 18 | V$_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 19 | PB.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SCL | O | MFP2 | I2C0 clock pin. |
| | UART0_TXD | O | MFP3 | Data transmitter output pin for UART0. |
| | SD_DAT2 | I/O | MFP4 | SD/SDH mode data line bit 2. |
| 20 | PB.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SDA | I/O | MFP2 | I2C0 data input/output pin. |
| | UART0_RXD | I | MFP3 | Data receiver input pin for UART0. |
| | SD_DAT3 | I/O | MFP4 | SD/SDH mode data line bit 3. |
| 21 | PB.2 | I/O | MFP0 | General purpose digital I/O pin. |

NUC505 SERIES TECHNICAL REFERENCE MANUAL

| | | | | |
|---|---|---|---|---|
| | SPI0_SS | O | MFP1 | SPI0 slave select pin. |
| | SD_CMD | I | MFP4 | SD/SDH mode – command/response. |
| 22 | PB.3 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_CLK | O | MFP1 | SPI0 serial clock pin. |
| | SD_CLK | O | MFP4 | SD/SDH mode – clock. |
| | SYSCFG[2] | I | MFP0 | System configuration setting bit 2. |
| 23 | PB.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_MOSI | O | MFP1 | SPI0 MOSI (Master Out, Slave In) pin. |
| | SYSCFG[3] | I | MFP0 | System configuration setting bit 3. |
| 24 | PB.5 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_MISO | I | MFP1 | SPI0 MISO (Master In, Slave Out) pin. |
| | SD_nCD | I | MFP4 | SD/SDH mode – card detect. |
| 25 | USB_VBUS33 | I | MFP0 | Detects whether USB is plug-in. |
| 26 | $V_{SS}$ | A | MFP0 | Ground |
| 27 | $AV_{DDHP}$ | A | MFP0 | Power supply for analog CODEC headphone, DC 3.3V. |
| 28 | LHPOUT | A | MFP0 | Headphone left channel output pin. |
| 29 | RHPOUT | A | MFP0 | Headphone right channel output pin. |
| 30 | $AV_{SSHP}$ | A | MFP0 | Ground for analog CODEC headphone. |
| 31 | VMID | A | MFP0 | Headphone reference power. |
| 32 | $AV_{DDCODEC}$ | A | MFP0 | Power supply for analog CODEC, DC 3.3V. |
| 33 | MIC0_P | A | MFP0 | Microphone 0 positive input. |
| 34 | MIC0_N | A | MFP0 | Microphone 0 negative input. |
| 35 | MIC_BIAS | A | MFP0 | CODEC left line-in channel or Microphone bias. |
| 36 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 37 | PD.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | RLINEIN | A | MFP1 | CODEC right line-in channel. |
| 38 | $AV_{DDADC}$ | A | MFP0 | Power supply for analog SAR-ADC, DC 3.3V. |

| | | | | |
|---|---|---|---|---|
| 39 | $AV_{SSADC}$ | A | MFP0 | Ground pin for analog SAR-ADC. |
| 40 | PA.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH0 | A | MFP1 | ADC channel 0 analog input. |
| 41 | PA.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH1 | A | MFP1 | ADC channel 1 analog input. |
| 42 | PA.2 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH2 | A | MFP1 | ADC channel 2 analog input. |
| | I2S_MCLK | O | MFP2 | I$^2$S master clock output pin. |
| 43 | PA.3 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH3 | A | MFP1 | ADC channel 3 analog input. |
| | I2S_DI | I | MFP2 | I$^2$S data input. |
| 44 | PA.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH4 | A | MFP1 | ADC channel 4 analog input. |
| | I2S_DO | O | MFP2 | I$^2$S data output. |
| 45 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 46 | $V_{DD}$ | A | MFP0 | Power supply, DC 3.3V. |
| 47 | LDO_CAP | A | MFP0 | LDO output pin. |
| 48 | $V_{SS}$ | A | MFP0 | Ground. |

Note: The thermal pad (EPD) on the bottom of QFN package should be connected to GND.

### 4.3.4 NuMicro® NUC505YLA2Y QFN 48-pin Description

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

PA.0 MFP0 means SYS_GPA_MFPL[2:0]=0x0.

PA.9 MFP5 means SYS_GPA_MFPH[6:4]=0x5.

| Pin No. | Pin Name | Type | MFP* | Description |
|---|---|---|---|---|
| 1 | nRESET | I | MFP0 | External reset input: active LOW, with an internal pull-up. Set this pin low reset to initial state. |
| 2 | ICE_CLK | O | MFP0 | Serial wired debugger clock pin. (In ICE mode) |
| | PD.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SCL | O | MFP2 | I$^2$C0 clock pin. |
| 3 | ICE_DAT | I/O | MFP0 | Serial wired debugger data pin. (In ICE mode) |
| | PD.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SDA | I/O | MFP2 | I$^2$C0 data input/output pin. |
| 4 | PB.13 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_MISO | I | MFP1 | SPI1 MISO (Master In, Slave Out) pin. |
| | USBH1_D- | I/O | MFP2 | USB host-lite 1 differential signal D-. |
| | UART2_nRTS | O | MFP3 | Request to Send output pin for UART2. |
| | PWM_CH3 | I/O | MFP4 | PWM channel3 output/capture input. |
| 5 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 6 | $V_{SS}$ | A | MFP0 | Ground. |
| 7 | XT1_IN | I | MFP0 | External 12 MHz (high speed) crystal input pin. |
| 8 | XT1_OUT | O | MFP0 | External 12 MHz (high speed) crystal output pin. |
| 9 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 10 | USB_D- | A | MFP0 | USB differential signal D-. |
| 11 | USB_D+ | A | MFP0 | USB differential signal D+. |
| 12 | $AV_{DDUSB}$ | A | MFP0 | Power supply for analog USB, DC 3.3V. |
| 13 | USB_REXT | A | MFP0 | 12.1 KΩ used internally for USB circuitry. |
| 14 | $V_{BAT}$ | A | MFP0 | Power supply by batteries for RTC, DC 3.3V. |

| | | | | |
|---|---|---|---|---|
| 15 | RTC_RPWR | O | MFP0 | Enable external power control source when active high. |
| 16 | RTC_nRWAKE | I | MFP0 | System power enable trigger when active low. |
| 17 | PA.8 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_SS | O | MFP1 | SPIM slave select pin. |
| | I2S_LRCLK | I/O | MFP2 | I$^2$S left right channel clock. |
| | UART1_TXD | O | MFP3 | Data transmitter output pin for UART1. |
| 18 | PA.9 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_CLK | O | MFP1 | SPIM serial clock pin. |
| | I2S_BCLK | I/O | MFP2 | I$^2$S bit clock pin. |
| | UART1_RXD | I | MFP3 | Data receiver input pin for UART1. |
| | SYSCFG[0] | I | MFP0 | System configuration setting bit 0. |
| 19 | PA.10 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_MOSI | I/O | MFP1 | SPIM MOSI (Master Out, Slave In) pin. |
| | I2C1_SCL | O | MFP2 | I2C1 clock pin. |
| | SD_CLK | O | MFP4 | SD/SDH mode - clock. |
| | SYSCFG[1] | I | MFP0 | System configuration setting bit 1. |
| 20 | PA.11 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_MISO | I/O | MFP1 | SPIM MISO (Master In, Slave Out) pin. |
| | I2C1_SDA | I/O | MFP2 | I2C1 data input/output pin. |
| | SD_CMD | I | MFP4 | SD/SDH mode – command/response. |
| 21 | V$_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 22 | PB.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SCL | O | MFP2 | I2C0 clock pin. |
| | UART0_TXD | O | MFP3 | Data transmitter output pin for UART0. |
| | SD_DAT2 | I/O | MFP4 | SD/SDH mode data line bit 2. |
| 23 | PB.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SDA | I/O | MFP2 | I2C0 data input/output pin. |

NUC505 SERIES TECHNICAL REFERENCE MANUAL

| | | | | |
|---|---|---|---|---|
| | UART0_RXD | I | MFP3 | Data receiver input pin for UART0. |
| | SD_DAT3 | I/O | MFP4 | SD/SDH mode data line bit 3. |
| 24 | PB.2 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_SS | O | MFP1 | SPI0 slave select pin. |
| | SD_CMD | I | MFP4 | SD/SDH mode – command/response. |
| 25 | PB.3 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_CLK | O | MFP1 | SPI0 serial clock pin. |
| | SD_CLK | O | MFP4 | SD/SDH mode – clock. |
| | SYSCFG[2] | I | MFP0 | System configuration setting bit 2. |
| 26 | PB.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_MOSI | O | MFP1 | SPI0 MOSI (Master Out, Slave In) pin. |
| | SYSCFG[3] | I | MFP0 | System configuration setting bit 3. |
| 27 | PB.5 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_MISO | I | MFP1 | SPI0 MISO (Master In, Slave Out) pin. |
| | SD_nCD | I | MFP4 | SD/SDH mode – card detect. |
| 28 | PB.6 | I/O | MFP0 | General purpose digital I/O pin. |
| | UART1_TXD | O | MFP3 | Data transmitter output pin for UART1. |
| | SD_DAT0 | I/O | MFP4 | SD/SDH mode data line bit 0. |
| 29 | PB.7 | I/O | MFP0 | General purpose digital I/O pin. |
| | UART1_RXD | I | MFP3 | Data receiver input pin for UART1. |
| | SD_DAT1 | I/O | MFP4 | SD/SDH mode data line bit 1. |
| 30 | USB_VBUS33 | I | MFP0 | Detects whether USB is plug-in. |
| 31 | PB.8 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH_PWEN | O | MFP1 | USB host mode to control an external overcurrent source. |
| | TM1_CNT_OUT | I/O | MFP2 | Timer1 event counter input/toggle output. |
| | UART1_nCTS | I | MFP3 | Clear to Send input pin for UART1. |
| | SD_DAT2 | I/O | MFP4 | SD/SDH mode data line bit 2. |

| | | | | |
|---|---|---|---|---|
| 32 | PB.9 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH_OVD | I | MFP1 | USB host bus power over voltage detector. |
| | TM1_EXT | I | MFP2 | Timer1 external capture input. |
| | UART1_nRTS | O | MFP3 | Request to Send output pin for UART1. |
| | SD_DAT3 | I/O | MFP4 | SD/SDH mode data line bit 3. |
| 33 | $V_{SS}$ | A | MFP0 | Ground |
| 34 | $AV_{DDADC}$ | A | MFP0 | Power supply for analog SAR-ADC, DC 3.3V. |
| 35 | $AV_{SSADC}$ | A | MFP0 | Ground pin for analog SAR-ADC. |
| 36 | PA.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH0 | A | MFP1 | ADC channel 0 analog input. |
| 37 | PA.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH1 | A | MFP1 | ADC channel 1 analog input. |
| 38 | PA.2 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH2 | A | MFP1 | ADC channel 2 analog input. |
| | I2S_MCLK | O | MFP2 | I$^{2}$S master clock output pin. |
| 39 | PA.3 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH3 | A | MFP1 | ADC channel 3 analog input. |
| | I2S_DI | I | MFP2 | I$^{2}$S data input. |
| 40 | PA.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH4 | A | MFP1 | ADC channel 4 analog input. |
| | I2S_DO | O | MFP2 | I$^{2}$S data output. |
| 41 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 42 | PB.10 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_SS | O | MFP1 | SPI1 slave select pin. |
| | I2C1_SCL | O | MFP2 | I2C1 clock pin. |
| | UART2_TXD | O | MFP3 | Data transmitter output pin for UART2. |
| | PWM_CH0 | I/O | MFP4 | PWM channel0 output/capture input. |

| | | | | |
|---|---|---|---|---|
| 43 | PB.11 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_CLK | O | MFP1 | SPI1 serial clock pin. |
| | I2C1_SDA | I/O | MFP2 | I2C1 data input/output pin. |
| | UART2_RXD | I | MFP3 | Data receiver input pin for UART2. |
| | PWM_CH1 | I/O | MFP4 | PWM channel1 output/capture input. |
| 44 | PB.12 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_MOSI | O | MFP1 | SPI1 MOSI (Master Out, Slave In) pin. |
| | USBH1_D+ | I/O | MFP2 | USB host-lite 1 differential signal D+ |
| | UART2_nCTS | I | MFP3 | Clear to send input pin for UART2. |
| | PWM_CH2 | I/O | MFP4 | PWM channel2 output/capture input. |
| 45 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 46 | $V_{DD}$ | A | MFP0 | Power supply, DC 3.3V. |
| 47 | LDO_CAP | A | MFP0 | LDO output pin. |
| 48 | $V_{SS}$ | A | MFP0 | Ground. |

Note: The thermal pad (EPD) on the bottom of QFN package should be connected to GND.

### 4.3.5 NuMicro® NUC505DSA LQFP 64-pin Description

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

PA.0 MFP0 means SYS_GPA_MFPL[2:0]=0x0.

PA.9 MFP5 means SYS_GPA_MFPH[6:4]=0x5.

| Pin No. | Pin Name | Type | MFP | Description |
|---|---|---|---|---|
| 1 | nRESET | I | MFP0 | External reset input: active LOW, with an internal pull-up. Set this pin low reset to initial state. |
| 2 | ICE_CLK | O | MFP0 | Serial wired debugger clock pin. (In ICE mode) |
| | PD.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SCL | O | MFP2 | I2C0 clock pin. |
| 3 | ICE_DAT | I/O | MFP0 | Serial wired debugger data pin. (In ICE mode) |
| | PD.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SDA | I/O | MFP2 | I2C0 data input/output pin. |
| 4 | PB.13 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_MISO | I | MFP1 | SPI1 MISO (Master In, Slave Out) pin. |
| | USBH1_D- | I/O | MFP2 | USB host-lite 1 differential signal D-. |
| | UART2_nRTS | O | MFP3 | Request to Send output pin for UART2. |
| | PWM_CH3 | I/O | MFP4 | PWM channel3 output/capture input. |
| 5 | PB.14 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH1_D+ | I/O | MFP1 | USB host-lite 1 differential signal D+. |
| | I2C1_SCL | O | MFP2 | I2C1 clock pin. |
| 6 | PB.15 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH1_D- | I/O | MFP1 | USB host-lite 1differential signal D-. |
| | I2C1_SDA | I/O | MFP2 | I2C1 data input/output pin. |
| 7 | PC.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | SD_CMD | I | MFP1 | SD/SDH mode – command/response. |
| 8 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 9 | $V_{SS}$ | A | MFP0 | Ground |

| | | | | |
|---|---|---|---|---|
| 10 | XT1_IN | I | MFP0 | External 12 MHz (high speed) crystal input pin. |
| 11 | XT1_OUT | O | MFP0 | External 12 MHz (high speed) crystal output pin. |
| 12 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 13 | USB_D- | A | MFP0 | USB differential signal D-. |
| 14 | USB_D+ | A | MFP0 | USB differential signal D+. |
| 15 | $AV_{DDUSB}$ | A | MFP0 | Power supply for analog USB, DC 3.3V. |
| 16 | USB_REXT | A | MFP0 | 12.1 KΩ used internally for USB circuitry. |
| 17 | PA.8 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_SS | O | MFP1 | SPIM slave select pin. |
| | I2S_LRCLK | I/O | MFP2 | $I^2S$ left right channel clock. |
| | UART1_TXD | O | MFP3 | Data transmitter output pin for UART1. |
| 18 | PA.9 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_CLK | O | MFP1 | SPIM serial clock pin. |
| | I2S_BCLK | I/O | MFP2 | $I^2S$ bit clock pin. |
| | UART1_RXD | I | MFP3 | Data receiver input pin for UART1. |
| | SYSCFG[0] | I | MFP0 | System configuration setting bit 0. |
| 19 | PA.10 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_MOSI | I/O | MFP1 | SPIM MOSI (Master Out, Slave In) pin.<br>(Data 0 pin for Quad Mode I/O). |
| | I2C1_SCL | O | MFP2 | I2C1 clock pin. |
| | SD_CLK | O | MFP4 | SD/SDH mode – clock. |
| | SYSCFG[1] | I | MFP0 | System configuration setting bit 1. |
| 20 | PA.11 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_MISO | I/O | MFP1 | SPIM MISO (Master In, Slave Out) pin.<br>(Data 1 pin for Quad Mode I/O). |
| | I2C1_SDA | I/O | MFP2 | I2C1 data input/output pin. |
| | SD_CMD | I | MFP4 | SD/SDH mode – command/response. |
| 21 | PA.12 | I/O | MFP0 | General purpose digital I/O pin. |

| | | | | |
|---|---|---|---|---|
| | SPIM_D2 | I/O | MFP1 | SPIM data 2 pin for Quad Mode I/O. |
| | TM0_CNT_OUT | I | MFP2 | Timer0 event counter input/toggle output. |
| 22 | PA.13 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_D3 | I/O | MFP1 | SPIM data 3 pin for Quad Mode I/O. |
| | TM0_EXT | I | MFP2 | Timer0 external capture input. |
| | SD_nCD | I | MFP4 | SD/SDH mode – card detect. |
| 23 | PA.14 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SCL | O | MFP2 | I2C0 clock pin. |
| | SD_DAT0 | I/O | MFP4 | SD/SDH mode data line bit 0. |
| 24 | PA.15 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SDA | I/O | MFP2 | I2C0 data input/output pin. |
| | SD_DAT1 | I/O | MFP4 | SD/SDH mode data line bit 1. |
| 25 | PC.8 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2S_MCLK | O | MFP1 | I$^2$S master clock output pin. |
| 26 | PC.9 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2S_DI | I/O | MFP1 | I$^2$S data input. |
| | TM2_CNT_OUT | I/O | MFP2 | Timer2 event counter input/toggle output. |
| | PWM_CH0 | I/O | MFP3 | PWM channel0 output/capture input. |
| 27 | PC.10 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2S_DO | I/O | MFP1 | I$^2$S data output. |
| | TM2_EXT | I | MFP2 | Timer2 external capture input. |
| | PWM_CH1 | I/O | MFP3 | PWM channel1 output/capture input. |
| 28 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 29 | PB.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SCL | O | MFP2 | I2C0 clock pin. |
| | UART0_TXD | O | MFP3 | Data transmitter output pin for UART0. |
| | SD_DAT2 | I/O | MFP4 | SD/SDH mode data line bit 2. |

NUC505 SERIES TECHNICAL REFERENCE MANUAL

| | | | | |
|---|---|---|---|---|
| 30 | PB.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SDA | I/O | MFP2 | I2C0 data input/output pin. |
| | UART0_RXD | I | MFP3 | Data receiver input pin for UART0. |
| | SD_DAT3 | I/O | MFP4 | SD/SDH mode data line bit 3. |
| 31 | PB.2 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_SS | O | MFP1 | SPI0 slave select pin. |
| | SD_CMD | I | MFP4 | SD/SDH mode – command/response. |
| 32 | PB.3 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_CLK | O | MFP1 | SPI0 serial clock pin. |
| | SD_CLK | O | MFP4 | SD/SDH mode – clock. |
| | SYSCFG[2] | I | MFP0 | System configuration setting bit 2. |
| 33 | PB.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_MOSI | O | MFP1 | SPI0 MOSI (Master Out, Slave In) pin. |
| | SYSCFG[3] | I | MFP0 | System configuration setting bit 3. |
| 34 | PB.5 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_MISO | I | MFP1 | SPI0 MISO (Master In, Slave Out) pin. |
| | SD_nCD | I | MFP4 | SD/SDH mode – card detect. |
| 35 | USB_VBUS33 | I | MFP0 | Detects whether USB is plug-in. |
| 36 | $V_{SS}$ | A | MFP0 | Ground. |
| 37 | $AV_{DDHP}$ | AP | VDD | Power supply for analog CODEC headphone, DC 3.3V. |
| 38 | LHPOUT | A | MFP0 | Headphone left channel output pin. |
| 39 | RHPOUT | A | MFP0 | Headphone right channel output pin. |
| 40 | $AV_{SSHP}$ | A | MFP0 | Ground for analog CODEC headphone. |
| 41 | VMID | A | MFP0 | Headphone reference power. |
| 42 | $AV_{DDCODEC}$ | A | MFP0 | Power supply for analog CODEC, DC 3.3V. |
| 43 | MIC0_P | A | MFP0 | Microphone 0 positive input. |
| 44 | MIC0_N | A | MFP0 | Microphone 0 negative input. |

| | | | | |
|---|---|---|---|---|
| 45 | MIC_BIAS | A | MFP0 | CODEC left line-in channel or Microphone bias. |
| 46 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 47 | PD.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | RLINEIN | A | MFP1 | CODEC right line-in channel. |
| 48 | $AV_{DDADC}$ | A | MFP0 | Power supply for analog SAR-ADC, DC 3.3V. |
| 49 | $AV_{SSADC}$ | A | MFP0 | Ground pin for analog SAR-ADC. |
| 50 | PA.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH0 | A | MFP1 | ADC channel 0 analog input. |
| 51 | PA.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH1 | A | MFP1 | ADC channel 1 analog input. |
| 52 | PA.2 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH2 | A | MFP1 | ADC channel 2 analog input. |
| | I2S_MCLK | O | MFP2 | I$^2$S master clock output pin. |
| 53 | PA.3 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH3 | A | MFP1 | ADC channel 3 analog input. |
| | I2S_DI | I | MFP2 | I$^2$S data input. |
| 54 | PA.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH4 | A | MFP1 | ADC channel 4 analog input. |
| | I2S_DO | O | MFP2 | I$^2$S data output. |
| 55 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 56 | PB.10 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_SS | O | MFP1 | SPI1 slave select pin. |
| | I2C1_SCL | O | MFP2 | I2C1 clock pin. |
| | UART2_TXD | O | MFP3 | Data transmitter output pin for UART2. |
| | PWM_CH0 | I/O | MFP4 | PWM channel0 output/capture input. |
| 57 | PB.11 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_CLK | O | MFP1 | SPI1 serial clock pin. |

| | | | | |
|---|---|---|---|---|
| | I2C1_SDA | I/O | MFP2 | I2C1 data input/output pin. |
| | UART2_RXD | I | MFP3 | Data receiver input pin for UART2. |
| | PWM_CH1 | I/O | MFP4 | PWM channel1 output/capture input. |
| 58 | PB.12 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_MOSI | O | MFP1 | SPI1 MOSI (Master Out, Slave In) pin. |
| | USBH1_D+ | I/O | MFP2 | USB host-lite 1 differential signal D+ |
| | UART2_nCTS | I | MFP3 | Clear to Send input pin for UART2. |
| | PWM_CH2 | I/O | MFP4 | PWM channel2 output/capture input. |
| 59 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 60 | PC.11 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2S_LRCLK | I/O | MFP1 | I$^2$S left right channel clock. |
| | TM3_CNT_OUT | I/O | MFP2 | Timer3 event counter input/toggle output. |
| | PWM_CH2 | I/O | MFP3 | PWM channel2 output/capture input. |
| 61 | PC.12 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2S_BCLK | I/O | MFP1 | I$^2$S bit clock pin. |
| | TM3_EXT | I | MFP2 | Timer3 external capture input. |
| | PWM_CH3 | I/O | MFP3 | PWM channel3 output/capture input. |
| 62 | $V_{DD}$ | A | MFP0 | Power supply, DC 3.3V. |
| 63 | LDO_CAP | A | MFP0 | LDO output pin. |
| 64 | $V_{SS}$ | A | MFP0 | Ground. |

NUC505 SERIES TECHNICAL REFERENCE MANUAL

### 4.3.6 NuMicro® NUC505DS13Y LQFP 64-pin Description

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

PA.0 MFP0 means SYS_GPA_MFPL[2:0]=0x0.

PA.9 MFP5 means SYS_GPA_MFPH[6:4]=0x5.

| Pin No. | Pin Name | Type | MFP | Description |
|---|---|---|---|---|
| 1 | nRESET | I | MFP0 | External reset input: active LOW, with an internal pull-up. Set this pin low reset to initial state. |
| 2 | ICE_CLK | O | MFP0 | Serial wired debugger clock pin. (In ICE mode) |
| | PD.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SCL | O | MFP2 | I2C0 clock pin. |
| 3 | ICE_DAT | I/O | MFP0 | Serial wired debugger data pin. (In ICE mode) |
| | PD.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SDA | I/O | MFP2 | I2C0 data input/output pin. |
| 4 | PB.13 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_MISO | I | MFP1 | SPI1 MISO (Master In, Slave Out) pin. |
| | USBH1_D- | I/O | MFP2 | USB host-lite 1 differential signal D-. |
| | UART2_nRTS | O | MFP3 | Request to Send output pin for UART2. |
| | PWM_CH3 | I/O | MFP4 | PWM channel3 output/capture input. |
| 5 | PB.14 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH1_D+ | I/O | MFP1 | USB host-lite 1 differential signal D+. |
| | I2C1_SCL | O | MFP2 | I2C1 clock pin. |
| 6 | PB.15 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH1_D- | I/O | MFP1 | USB host-lite 1differential signal D-. |
| | I2C1_SDA | I/O | MFP2 | I2C1 data input/output pin. |
| 7 | PC.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | SD_CMD | I | MFP1 | SD/SDH mode – command/response. |
| 8 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 9 | $V_{SS}$ | A | MFP0 | Ground |

| | | | | |
|---|---|---|---|---|
| 10 | XT1_IN | I | MFP0 | External 12 MHz (high speed) crystal input pin. |
| 11 | XT1_OUT | O | MFP0 | External 12 MHz (high speed) crystal output pin. |
| 12 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 13 | USB_D- | A | MFP0 | USB differential signal D-. |
| 14 | USB_D+ | A | MFP0 | USB differential signal D+. |
| 15 | $AV_{DDUSB}$ | A | MFP0 | Power supply for analog USB, DC 3.3V. |
| 16 | USB_REXT | A | MFP0 | 12.1 KΩ used internally for USB circuitry. |
| 17 | $V_{BAT}$ | A | MFP0 | Power supply by batteries for RTC, DC 3.3V. |
| 18 | RTC_RPWR | O | MFP0 | Enable external power control source when active high. |
| 19 | RTC_nRWAKE | I | MFP0 | System power enable trigger when active low. |
| 20 | PA.8 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_SS | O | MFP1 | SPIM slave select pin. |
| | I2S_LRCLK | I/O | MFP2 | I$^2$S left right channel clock. |
| | UART1_TXD | O | MFP3 | Data transmitter output pin for UART1. |
| 21 | PA.9 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_CLK | O | MFP1 | SPIM serial clock pin. |
| | I2S_BCLK | I/O | MFP2 | I$^2$S bit clock pin. |
| | UART1_RXD | I | MFP3 | Data receiver input pin for UART1. |
| | SYSCFG[0] | I | MFP0 | System configuration setting bit 0. |
| 22 | PA.10 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_MOSI | I/O | MFP1 | SPIM MOSI (Master Out, Slave In) pin.<br>(Data 0 pin for Quad Mode I/O). |
| | I2C1_SCL | O | MFP2 | I2C1 clock pin. |
| | SD_CLK | O | MFP4 | SD/SDH mode – clock. |
| | SYSCFG[1] | I | MFP0 | System configuration setting bit 1. |
| 23 | PA.11 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_MISO | I/O | MFP1 | SPIM MISO (Master In, Slave Out) pin.<br>(Data 1 pin for Quad Mode I/O). |

|  |  |  |  |  |
|---|---|---|---|---|
|  | I2C1_SDA | I/O | MFP2 | I2C1 data input/output pin. |
|  | SD_CMD | I | MFP4 | SD/SDH mode – command/response. |
| 24 | PA.12 | I/O | MFP0 | General purpose digital I/O pin. |
|  | SPIM_D2 | I/O | MFP1 | SPIM data 2 pin for Quad Mode I/O. |
|  | TM0_CNT_OUT | I | MFP2 | Timer0 event counter input/toggle output. |
| 25 | PA.13 | I/O | MFP0 | General purpose digital I/O pin. |
|  | SPIM_D3 | I/O | MFP1 | SPIM data 3 pin for Quad Mode I/O. |
|  | TM0_EXT | I | MFP2 | Timer0 external capture input. |
|  | SD_nCD | I | MFP4 | SD/SDH mode – card detect. |
| 26 | PA.14 | I/O | MFP0 | General purpose digital I/O pin. |
|  | I2C0_SCL | O | MFP2 | I2C0 clock pin. |
|  | SD_DAT0 | I/O | MFP4 | SD/SDH mode data line bit 0. |
| 27 | PA.15 | I/O | MFP0 | General purpose digital I/O pin. |
|  | I2C0_SDA | I/O | MFP2 | I2C0 data input/output pin. |
|  | SD_DAT1 | I/O | MFP4 | SD/SDH mode data line bit 1. |
| 28 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 29 | PB.0 | I/O | MFP0 | General purpose digital I/O pin. |
|  | I2C0_SCL | O | MFP2 | I2C0 clock pin. |
|  | UART0_TXD | O | MFP3 | Data transmitter output pin for UART0. |
|  | SD_DAT2 | I/O | MFP4 | SD/SDH mode data line bit 2. |
| 30 | PB.1 | I/O | MFP0 | General purpose digital I/O pin. |
|  | I2C0_SDA | I/O | MFP2 | I2C0 data input/output pin. |
|  | UART0_RXD | I | MFP3 | Data receiver input pin for UART0. |
|  | SD_DAT3 | I/O | MFP4 | SD/SDH mode data line bit 3. |
| 31 | PB.2 | I/O | MFP0 | General purpose digital I/O pin. |
|  | SPI0_SS | O | MFP1 | SPI0 slave select pin. |
|  | SD_CMD | I | MFP4 | SD/SDH mode – command/response. |

| | | | | |
|---|---|---|---|---|
| 32 | PB.3 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_CLK | O | MFP1 | SPI0 serial clock pin. |
| | SD_CLK | O | MFP4 | SD/SDH mode – clock. |
| | SYSCFG[2] | I | MFP0 | System configuration setting bit 2. |
| 33 | PB.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_MOSI | O | MFP1 | SPI0 MOSI (Master Out, Slave In) pin. |
| | SYSCFG[3] | I | MFP0 | System configuration setting bit 3. |
| 34 | PB.5 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_MISO | I | MFP1 | SPI0 MISO (Master In, Slave Out) pin. |
| | SD_nCD | I | MFP4 | SD/SDH mode – card detect. |
| 35 | PB.6 | I/O | MFP0 | General purpose digital I/O pin. |
| | UART1_TXD | O | MFP3 | Data transmitter output pin for UART1. |
| | SD_DAT0 | I/O | MFP4 | SD/SDH mode data line bit 0. |
| 36 | PB.7 | I/O | MFP0 | General purpose digital I/O pin. |
| | UART1_RXD | I | MFP3 | Data receiver input pin for UART1. |
| | SD_DAT1 | I/O | MFP4 | SD/SDH mode data line bit 1. |
| 37 | USB_VBUS33 | I | MFP0 | Detects whether USB is plug-in. |
| 38 | PB.8 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH_PWEN | O | MFP1 | USB host mode to control an external overcurrent source. |
| | TM1_CNT_OUT | I/O | MFP2 | Timer1 event counter input/toggle output. |
| | UART1_nCTS | I | MFP3 | Clear to Send input pin for UART1. |
| | SD_DAT2 | I/O | MFP4 | SD/SDH mode data line bit 2. |
| 39 | PB.9 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH_OVD | I | MFP1 | USB host bus power over voltage detector. |
| | TM1_EXT | I | MFP2 | Timer1 external capture input. |
| | UART1_nRTS | O | MFP3 | Request to Send output pin for UART1. |
| | SD_DAT3 | I/O | MFP4 | SD/SDH mode data line bit 3. |

| | | | | |
|---|---|---|---|---|
| 40 | $V_{SS}$ | A | MFP0 | Ground. |
| 41 | $AV_{DDHP}$ | AP | VDD | Power supply for analog CODEC headphone, DC 3.3V. |
| 42 | LHPOUT | A | MFP0 | Headphone left channel output pin. |
| 43 | RHPOUT | A | MFP0 | Headphone right channel output pin. |
| 44 | $AV_{SSHP}$ | A | MFP0 | Ground for analog CODEC headphone. |
| 45 | VMID | A | MFP0 | Headphone reference power. |
| 46 | $AV_{DDCODEC}$ | A | MFP0 | Power supply for analog CODEC, DC 3.3V. |
| 47 | $AV_{DDADC}$ | A | MFP0 | Power supply for analog SAR-ADC, DC 3.3V. |
| 48 | $AV_{SSADC}$ | A | MFP0 | Ground pin for analog SAR-ADC. |
| 49 | PA.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH0 | A | MFP1 | ADC channel 0 analog input. |
| 50 | PA.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH1 | A | MFP1 | ADC channel 1 analog input. |
| 51 | PA.2 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH2 | A | MFP1 | ADC channel 2 analog input. |
| | I2S_MCLK | O | MFP2 | I$^2$S master clock output pin. |
| 52 | PA.3 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH3 | A | MFP1 | ADC channel 3 analog input. |
| | I2S_DI | I | MFP2 | I$^2$S data input. |
| 53 | PA.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH4 | A | MFP1 | ADC channel 4 analog input. |
| | I2S_DO | O | MFP2 | I$^2$S data output. |
| 54 | PA.5 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH5 | A | MFP1 | ADC channel 5 analog input. |
| 55 | PA.6 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH6 | A | MFP1 | ADC channel 6 analog input. |
| 56 | PA.7 | I/O | MFP0 | General purpose digital I/O pin. |

| | | | | |
|---|---|---|---|---|
| | ADC_CH7 | A | MFP1 | ADC channel 7 analog input. |
| 57 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 58 | PB.10 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_SS | O | MFP1 | SPI1 slave select pin. |
| | I2C1_SCL | O | MFP2 | I2C1 clock pin. |
| | UART2_TXD | O | MFP3 | Data transmitter output pin for UART2. |
| | PWM_CH0 | I/O | MFP4 | PWM channel0 output/capture input. |
| 59 | PB.11 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_CLK | O | MFP1 | SPI1 serial clock pin. |
| | I2C1_SDA | I/O | MFP2 | I2C1 data input/output pin. |
| | UART2_RXD | I | MFP3 | Data receiver input pin for UART2. |
| | PWM_CH1 | I/O | MFP4 | PWM channel1 output/capture input. |
| 60 | PB.12 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_MOSI | O | MFP1 | SPI1 MOSI (Master Out, Slave In) pin. |
| | USBH1_D+ | I/O | MFP2 | USB host-lite 1 differential signal D+ |
| | UART2_nCTS | I | MFP3 | Clear to Send input pin for UART2. |
| | PWM_CH2 | I/O | MFP4 | PWM channel2 output/capture input. |
| 61 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 62 | $V_{DD}$ | A | MFP0 | Power supply, DC 3.3V. |
| 63 | LDO_CAP | A | MFP0 | LDO output pin. |
| 64 | $V_{SS}$ | A | MFP0 | Ground. |

NUC505 SERIES TECHNICAL REFERENCE MANUAL

### 4.3.7 NuMicro® NUC505YO13Y QFN 88-pin Description

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

PA.0 MFP0 means SYS_GPA_MFPL[2:0]=0x0.

PA.9 MFP5 means SYS_GPA_MFPH[6:4]=0x5.

| Pin No. | Pin Name | Type | MFP | Description |
|---|---|---|---|---|
| 1 | nRESET | I | MFP0 | External reset input: active LOW, with an internal pull-up. Set this pin low reset to initial state. |
| 2 | ICE_CLK | O | MFP0 | Serial wired debugger clock pin. (In ICE mode) |
| | PD.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SCL | O | MFP2 | I2C0 clock pin. |
| 3 | ICE_DAT | I/O | MFP0 | Serial wired debugger data pin. (In ICE mode) |
| | PD.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SDA | I/O | MFP2 | I2C0 data input/output pin. |
| 4 | PB.13 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_MISO | I | MFP1 | SPI1 MISO (Master In, Slave Out) pin. |
| | USBH1_D- | I/O | MFP2 | USB host-lite 1 differential signal D-. |
| | UART2_nRTS | O | MFP3 | Request to Send output pin for UART2. |
| | PWM_CH3 | I/O | MFP4 | PWM channel3 output/capture input. |
| 5 | PB.14 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH1_D+ | I/O | MFP1 | USB host-lite 1differential signal D+. |
| | I2C1_SCL | O | MFP2 | I2C1 clock pin. |
| 6 | PB.15 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH1_D- | I/O | MFP1 | USB host-lite 1 differential signal D-. |
| | I2C1_SDA | I/O | MFP2 | I2C1 data input/output pin. |
| 7 | PC.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | SD_CMD | I | MFP1 | SD/SDH mode – command/response. |
| 8 | PC.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | SD_CLK | O | MFP1 | SD/SDH mode – clock. |

| Pin No. | Pin Name | Type | MFP | Description |
|---|---|---|---|---|
| 9 | PC.2 | I/O | MFP0 | General purpose digital I/O pin. |
| | SD_nCD | I | MFP1 | SD/SDH mode – card detect. |
| 10 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 11 | PC.3 | I/O | MFP0 | General purpose digital I/O pin. |
| 12 | PC.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | SD_DAT0 | I/O | MFP1 | SD/SDH mode data line bit 0. |
| 13 | PC.5 | I/O | MFP0 | General purpose digital I/O pin. |
| | SD_DAT1 | I/O | MFP1 | SD/SDH mode data line bit 1. |
| 14 | PC.6 | I/O | MFP0 | General purpose digital I/O pin. |
| | SD_DAT2 | I/O | MFP1 | SD/SDH mode data line bit 2. |
| 15 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 16 | XT1_IN | I | MFP0 | External 12 MHz (high speed) crystal input pin. |
| 17 | XT1_OUT | O | MFP0 | External 12 MHz (high speed) crystal output pin. |
| 18 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 19 | USB_D- | A | MFP0 | USB differential signal D-. |
| 20 | USB_D+ | A | MFP0 | USB differential signal D+. |
| 21 | $AV_{DDUSB}$ | A | MFP0 | Power supply for analog USB, DC 3.3V. |
| 22 | USB_REXT | A | MFP0 | 12.1 KΩ used internally for USB circuitry. |
| 23 | $V_{BAT}$ | A | MFP0 | Power supply by batteries for RTC, DC 3.3V. |
| 24 | RTC_RPWR | O | MFP0 | Enable external power control source when active high. |
| 25 | RTC_nRWAKE | I | MFP0 | System power enable trigger when active low. |
| 26 | X32_IN | I | MFP0 | External 32.768 kHz (low speed) crystal input pin. |
| 27 | X32_OUT | O | MFP0 | External 32.768 kHz (low speed) crystal output pin. |
| 28 | PA.8 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_SS | O | MFP1 | SPIM slave select pin. |
| | I2S_LRCLK | I/O | MFP2 | $I^2S$ left right channel clock. |

| Pin No. | Pin Name | Type | MFP | Description |
|---|---|---|---|---|
| | UART1_TXD | O | MFP3 | Data transmitter output pin for UART1. |
| 29 | PA.9 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_CLK | O | MFP1 | SPIM serial clock pin. |
| | I2S_BCLK | I/O | MFP2 | I$^2$S bit clock pin. |
| | UART1_RXD | I | MFP3 | Data receiver input pin for UART1. |
| | SYSCFG[0] | I | MFP0 | System configuration setting bit 0. |
| 30 | PA.10 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_MOSI | I/O | MFP1 | SPIM MOSI (Master Out, Slave In) pin.<br>(Data 0 pin for Quad Mode I/O). |
| | I2C1_SCL | O | MFP2 | I2C1 clock pin. |
| | SD_CLK | O | MFP4 | SD/SDH mode – clock. |
| | SYSCFG[1] | I | MFP0 | System configuration setting bit 1. |
| 31 | PA.11 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_MISO | I/O | MFP1 | SPIM MISO (Master In, Slave Out) pin.<br>(Data 1 pin for Quad Mode I/O). |
| | I2C1_SDA | I/O | MFP2 | I2C1 data input/output pin. |
| | SD_CMD | I | MFP4 | SD/SDH mode – command/response. |
| 32 | PA.12 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_D2 | I/O | MFP1 | SPIM data 2 pin for Quad Mode I/O. |
| | TM0_CNT_OUT | I/O | MFP2 | Timer0 event counter input/toggle output. |
| 33 | PA.13 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPIM_D3 | I/O | MFP1 | SPIM data 3 pin for Quad Mode I/O. |
| | TM0_EXT | I | MFP2 | Timer0 external capture input. |
| | SD_nCD | I | MFP4 | SD/SDH mode – card detect. |
| 34 | PA.14 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SCL | O | MFP2 | I2C0 clock pin. |
| | SD_DAT0 | I/O | MFP4 | SD/SDH mode data line bit 0. |

| Pin No. | Pin Name | Type | MFP | Description |
|---|---|---|---|---|
| 35 | PA.15 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SDA | I/O | MFP2 | I2C0 data input/output pin. |
| | SD_DAT1 | I/O | MFP4 | SD/SDH mode data line bit 1. |
| 36 | PC.7 | I/O | MFP0 | General purpose digital I/O pin. |
| | SD_DAT3 | I/O | MFP1 | SD/SDH mode data line bit 3. |
| 37 | PC.8 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2S_MCLK | O | MFP1 | I$^2$S master clock output pin. |
| 38 | PC.9 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2S_DI | I | MFP1 | I$^2$S data input. |
| | TM2_CNT_OUT | I/O | MFP2 | Timer2 event counter input/toggle output. |
| | PWM_CH0 | I/O | MFP3 | PWM channel0 output/capture input. |
| 39 | PC.10 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2S_DO | O | MFP1 | I$^2$S data output. |
| | TM2_EXT | I | MFP2 | Timer2 external capture input. |
| | PWM_CH1 | I/O | MFP3 | PWM channel1 output/capture input. |
| 40 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 41 | PB.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SCL | O | MFP2 | I2C0 clock pin. |
| | UART0_TXD | O | MFP3 | Data transmitter output pin for UART0. |
| | SD_DAT2 | I/O | MFP4 | SD/SDH mode data line bit 2. |
| 42 | PB.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2C0_SDA | I/O | FMP2 | I2C0 data input/output pin. |
| | UART0_RXD | I | FMP3 | Data receiver input pin for UART0. |
| | SD_DAT3 | I/O | MFP4 | SD/SDH mode data line bit 3. |
| 43 | PB.2 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_SS | O | MFP1 | SPI0 slave select pin. |

| Pin No. | Pin Name | Type | MFP | Description |
|---|---|---|---|---|
| | SD_CMD | I | MFP4 | SD/SDH mode – command/response. |
| 44 | PB.3 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_CLK | O | MFP1 | SPI0 serial clock pin. |
| | SD_CLK | O | MFP4 | SD/SDH mode – clock. |
| | SYSCFG[2] | I | MFP0 | System configuration setting bit 2. |
| 45 | PB.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_MOSI | O | MFP1 | SPI0 MOSI (Master Out, Slave In) pin. |
| | SYSCFG[3] | I | MFP0 | System configuration setting bit 3. |
| 46 | PB.5 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI0_MISO | I | MFP1 | SPI0 MISO (Master In, Slave Out) pin. |
| | SD_nCD | I | MFP4 | SD/SDH mode – card detect. |
| 47 | PB.6 | I/O | MFP0 | General purpose digital I/O pin. |
| | UART1_TXD | O | MFP3 | Data transmitter output pin for UART1. |
| | SD_DAT0 | I/O | MFP4 | SD/SDH mode data line bit 0. |
| 48 | PB.7 | I/O | MFP0 | General purpose digital I/O pin. |
| | UART1_RXD | I | MFP3 | Data receiver input pin for UART1. |
| | SD_DAT1 | I/O | MFP4 | SD/SDH mode data line bit 1. |
| 49 | USB_VBUS33 | I | MFP0 | Detects whether USB is plug-in. |
| 50 | PB.8 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH_PWEN | O | MFP1 | USB host mode to control an external overcurrent source. |
| | TM1_CNT_OUT | I/O | MFP2 | Timer1 event counter input/toggle output. |
| | UART1_nCTS | I | MFP3 | Clear to Send input pin for UART1. |
| | SD_DAT2 | I/O | MFP4 | SD/SDH mode data line bit 2. |
| 51 | PB.9 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH_VOD | I | MFP1 | USB host bus power over voltage detector. |
| | TM1_EXT | I | MFP2 | Timer1 external capture input. |

| Pin No. | Pin Name | Type | MFP | Description |
|---|---|---|---|---|
| | UART1_nRTS | O | MFP3 | Request to Send output pin for UART1. |
| | SD_DAT3 | I/O | MFP4 | SD/SDH mode data line bit 3. |
| 52 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 53 | $AV_{DDHP}$ | A | MFP0 | Power supply for analog CODEC headphone, DC 3.3V. |
| 54 | LHPOUT | A | MFP0 | Headphone left channel output pin. |
| 55 | VCMBF | A | MFP0 | Internal CODEC function, keep floating. |
| 56 | RHPOUT | A | MFP0 | Headphone right channel output pin. |
| 57 | $AV_{SSHP}$ | A | MFP0 | Ground for analog CODEC headphone. |
| 58 | VMID | A | MFP0 | Headphone reference power. |
| 59 | $AV_{DDCODEC}$ | A | MFP0 | Power supply for analog CODEC, DC 3.3V. |
| 60 | MIC0_P | A | MFP0 | Microphone 0 positive input. |
| 61 | MIC0_N | A | MFP0 | Microphone 0 negative input. |
| 62 | MIC_BIAS | A | MFP0 | CODEC left line-in channel or Microphone bias. |
| 63 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 64 | PD.2 | I/O | MFP0 | General purpose digital I/O pin. |
| | MIC1_P | A | MFP1 | Microphone 1 positive input. |
| 65 | PD.3 | I/O | MFP0 | General purpose digital I/O pin. |
| | MIC1_N | A | MFP1 | Microphone 1 negative input. |
| 66 | PD.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | RLINEIN | A | MFP1 | CODEC right line-in channel. |
| 67 | $AV_{DDADC}$ | A | MFP0 | Power supply for analog SAR-ADC, DC 3.3V. |
| 68 | $AV_{SSADC}$ | A | MFP0 | Ground pin for analog SAR-ADC. |
| 69 | PA.0 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH0 | A | MFP1 | ADC channel 0 analog input. |
| 70 | PA.1 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH1 | A | MFP1 | ADC channel 1 analog input. |

| Pin No. | Pin Name | Type | MFP | Description |
|---|---|---|---|---|
| 71 | PA.2 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH2 | A | MFP1 | ADC channel 2 analog input. |
| | I2S_MCLK | O | MFP2 | I$^2$S master clock output pin. |
| 72 | PA.3 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH3 | A | MFP1 | ADC channel 3 analog input. |
| | I2S_DI | I | MFP2 | I$^2$S data input. |
| 73 | PA.4 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH4 | A | MFP1 | ADC channel 4 analog input. |
| | I2S_DO | O | MFP2 | I$^2$S data output. |
| 74 | PA.5 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH5 | A | MFP1 | ADC channel 5 analog input. |
| 75 | PA.6 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH6 | A | MFP1 | ADC channel 6 analog input. |
| 76 | PA.7 | I/O | MFP0 | General purpose digital I/O pin. |
| | ADC_CH7 | A | MFP1 | ADC channel 7 analog input. |
| 77 | $V_{DD12}$ | A | MFP0 | Power supply for I/O ports, DC 1.2V |
| 78 | PB.10 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_SS | O | MFP1 | SPI1 slave select pin. |
| | I2C1_SCL | O | MFP2 | I2C1 clock pin. |
| | UART2_TXD | O | MFP3 | Data transmitter output pin for UART2. |
| | PWM_CH0 | I/O | MFP4 | PWM channel0 output/capture input. |
| 79 | PB.11 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_CLK | O | MFP1 | SPI1 serial clock pin. |
| | I2C1_SDA | I/O | MFP2 | I2C1 data input/output pin. |
| | URAT2_RXD | I | MFP3 | Data receiver input pin for UART2. |
| | PWM_CH1 | I/O | MFP4 | PWM channel1 output/capture input. |

| Pin No. | Pin Name | Type | MFP | Description |
|---|---|---|---|---|
| 80 | PB.12 | I/O | MFP0 | General purpose digital I/O pin. |
| | SPI1_MOSI | O | MFP1 | SPI1 MOSI (Master Out, Slave In) pin. |
| | USBH1_D+ | I/O | MFP2 | USB host-lite 1 differential signal D+. |
| | UART2_nCTS | I | MFP3 | Clear to Send input pin for UART2. |
| | PWM_CH2 | I/O | MFP4 | PWM channel2 output/capture input. |
| 81 | $V_{DD}$ | A | MFP0 | Power supply for I/O ports, DC 3.3V. |
| 82 | PC.11 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2S_LRCLK | I/O | MFP1 | $I^2S$ left right channel clock. |
| | TM3_CNT_OUT | I/O | MFP2 | Timer3 event counter input/toggle output. |
| | PWM_CH2 | I/O | MFP3 | PWM channel2 output/capture input. |
| 83 | PC.12 | I/O | MFP0 | General purpose digital I/O pin. |
| | I2S_BCLK | I/O | MFP1 | $I^2S$ bit clock pin. |
| | TM3_EXT | I | MFP2 | Timer3 external capture input. |
| | PWM_CH3 | I/O | MFP3 | PWM channel3 output/capture input. |
| 84 | PC.13 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH2_D+ | I/O | MFP1 | USB host-lite 2 differential signal D+. |
| 85 | PC.14 | I/O | MFP0 | General purpose digital I/O pin. |
| | USBH2_D- | I/O | MFP1 | USB host-lite 2 differential signal D-. |
| 86 | $V_{DD}$ | A | MFP0 | Power supply, DC 3.3V. |
| 87 | LDO_CAP | A | MFP0 | LDO output pin. |
| 88 | $V_{SS}$ | A | MFP0 | Ground. |

Note: The thermal pad (EPD) on the bottom of QFN package should be connected to GND.

### 4.3.8 Summary GPIO Multi-function Pin Description

| MPF0 | MPF1 | MPF2 | MPF3 | MPF4 | Other | Driving |
|---|---|---|---|---|---|---|
| PA.0 | ADC_CH0 | | | | | 2~16mA |
| PA.1 | ADC_CH 1 | | | | | 2~16mA |
| PA.2 | ADC_CH 2 | | | | | 2~16mA |
| PA.3 | ADC_CH 3 | | | | | 2~16mA |
| PA.4 | ADC_CH 4 | | | | | 2~16mA |
| PA.5 | ADC_CH 5 | | | | | 2~16mA |
| PA.6 | ADC_CH 6 | | | | | 2~16mA |
| PA.7 | ADC_CH 7 | | | | | 2~16mA |
| PA.8 | SPIM_SS | I2S_LRCLK | UART1_TXD | | | 8mA |
| PA.9 | SPIM_CLK | I2S_BCLK | UART1_RXD | | SYSCFG[0] | 8mA |
| PA.10 | SPIM_MOSI | I2C1_SCL | | SD_CLK | SYSCFG[1] | 8mA |
| PA.11 | SPIM_MISO | I2C1_SDA | | SD_CMD | | 8mA |
| PA.12 | SPIM_D2 | TM0_CNT_OUT | | | | 8mA |
| PA.13 | SPIM_D3 | TM0_EXT | | SD_nCD | | 8mA |
| PA.14 | | I2C0_SCL | | SD_DAT0 | | 4mA |
| PA.15 | | I2C0_SDA | | SD_DAT1 | | 4mA |
| PB.0 | | I2C0_SCL | UART0_TXD | SD_DAT2 | | 4mA |
| PB.1 | | I2C0_SDA | UART0_RXD | SD_DAT3 | | 4mA |
| PB.2 | SPI0_SS | | | SD_CMD | | 4mA |
| PB.3 | SPI0_CLK | | | SD_CLK | SYSCFG[2] | 4mA |
| PB.4 | SPI0_MOSI | | | | SYSCFG[3] | 4mA |
| PB.5 | SPI0_MISO | | | SD_nCD | | 4mA |
| PB.6 | | | UART1_TXD | SD_DAT0 | | 4mA |
| PB.7 | | | UART1_RXD | SD_DAT1 | | 4mA |
| PB.8 | USBH_PWEN | TM1_CNT_OUT | UART1_nCTS | SD_DAT2 | | 4mA |
| PB.9 | USBH_OVD | TM1_EXT | UART1_nRTS | SD_DAT3 | | 4mA |
| PB.10 | SPI1_SS | I2C1_SCL | UART2_TXD | PWM_CH0 | | 4mA |
| PB.11 | SPI1_CLK | I2C1_SDA | UART2_RXD | PWM_CH1 | | 4mA |
| PB.12 | SPI1_MOSI | USBH1_D+ | UART2_nCTS | PWM_CH2 | | 8mA |
| PB.13 | SPI1_MISO | USBH1_D- | UART2_nRST | PWM_CH3 | | 8mA |
| PB.14 | USBH1_D+ | I2C1_SCL | | | | 8mA |
| PB.15 | USBH1_D- | I2C1_SDA | | | | 8mA |

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| PC.0 | SD_CMD |  |  |  |  | 8mA |
| PC.1 | SD_CLK |  |  |  |  | 8mA |
| PC.2 | SD_nCD |  |  |  |  | 8mA |
| PC.3 |  |  |  |  |  | 8mA |
| PC.4 | SD_DAT0 |  |  |  |  | 8mA |
| PC.5 | SD_DAT1 |  |  |  |  | 8mA |
| PC.6 | SD_DAT2 |  |  |  |  | 8mA |
| PC.7 | SD_DAT3 |  |  |  |  | 8mA |
| PC.8 | I2S_MCLK |  |  |  |  | 4mA |
| PC.9 | I2S_DI | TM2_CNT_OUT | PWM_CH0 |  |  | 4mA |
| PC.10 | I2S_DO | TM2_EXT | PWM_CH1 |  |  | 4mA |
| PC.11 | I2S_LRCLK | TM3_CNT_OUT | PWM_CH2 |  |  | 4mA |
| PC.12 | I2S_BCLK | TM3_EXT | PWM_CH3 |  |  | 4mA |
| PC.13 | USBH2_D+ |  |  |  |  | 8mA |
| PC.14 | USBH2_D- |  |  |  |  | 8mA |
| PD.0 |  | I2C0_SCL |  |  | ICE_CLK | 4mA |
| PD.1 |  | I2C0_SDA |  |  | ICE_DAT | 4mA |
| PD.2 |  |  |  |  | MIC1_P | 2~16mA |
| PD.3 |  |  |  |  | MIC1_N | 2~16mA |
| PD.4 |  |  |  |  | RLINEIN | 2~16mA |

### 4.3.9 GPIO Multi-function Pin Summary

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

PA.0 MFP0 means SYS_GPA_MFPL[2:0]=0x0.

PA.9 MFP5 means SYS_GPA_MFPH[6:4]=0x5.

| Group | Pin Name | GPIO | MFP* | Type | Description |
|---|---|---|---|---|---|
| ADC | ADC_CH0 | PA.0 | MFP1 | A | ADC0 analog input. |
| | ADC_CH1 | PA.1 | MFP1 | A | ADC1 analog input. |
| | ADC_CH2 | PA.2 | MFP1 | A | ADC2 analog input. |
| | ADC_CH3 | PA.3 | MFP1 | A | ADC3 analog input. |
| | ADC_CH4 | PA.4 | MFP1 | A | ADC4 analog input. |
| | ADC_CH5 | PA.5 | MFP1 | A | ADC5 analog input. |
| | ADC_CH6 | PA.6 | MFP1 | A | ADC6 analog input. |
| | ADC_CH7 | PA.7 | MFP1 | A | ADC7 analog input. |
| CODEC | MIC1_P | PD.2 | MFP1 | A | Audio MIC1 analog positive input pin |
| | MIC1_N | PD.3 | MFP1 | A | Audio MIC1 analog negative input pin |
| | RLINEIN | PD.4 | MFP1 | A | Audio right line-in analog pin. |
| I2C0 | I2C0_SCL | PA.14 | MFP2 | I/O | I2C0 clock pin. |
| | I2C0_SCL | PB.0 | MFP2 | I/O | I2C0 clock pin. |
| | I2C0_SCL | PD.0 | MFP2 | I/O | I2C0 clock pin. |
| | I2C0_SDA | PA.15 | MFP2 | I/O | I2C0 data input/output pin. |
| | I2C0_SDA | PB.1 | MFP2 | I/O | I2C0 data input/output pin. |
| | I2C0_SDA | PD.1 | MFP2 | I/O | I2C0 data input/output pin. |
| I2C1 | I2C1_SCL | PA.10 | MFP2 | I/O | I2C1 clock pin. |
| | I2C1_SCL | PB.10 | MFP2 | I/O | I2C1 clock pin. |
| | I2C1_SCL | PB.14 | MFP2 | I/O | I2C1 clock pin. |
| | I2C1_SDA | PA.11 | MFP2 | I/O | I2C1 data input/output pin. |
| | I2C1_SDA | PB.11 | MFP2 | I/O | I2C1 data input/output pin. |
| | I2C1_SDA | PB.15 | MFP2 | I/O | I2C1 data input/output pin. |
| I$^2$S | I2S_MCLK | PA.2 | MFP2 | O | I$^2$S master clock output pin. |
| | I2S_MCLK | PC.8 | MFP1 | O | I$^2$S master clock output pin. |
| | I2S_BCLK | PA.9 | MFP2 | I/O | I$^2$S bit clock pin. |
| | I2S_BCLK | PC.12 | MFP1 | I/O | I$^2$S bit clock pin. |
| | I2S_LRCLK | PA.8 | MFP2 | I/O | I$^2$S left right channel pin. |
| | I2S_LRCLK | PC.11 | MFP1 | I/O | I$^2$S left right channel pin. |

| Group | Pin Name | GPIO | MFP* | Type | Description |
|---|---|---|---|---|---|
| | I2S_DO | PA.4 | MFP2 | O | $I^2S$ data output. |
| | I2S_DO | PC.10 | MFP1 | O | $I^2S$ data output. |
| | I2S_DI | PA.3 | MFP2 | I | $I^2S$ data input. |
| | I2S_DI | PC.9 | MFP1 | I | $I^2S$ data input. |
| ICE | ICE_CLK | PD.0 | MFP0 | I | Serial wired debugger clock pin |
| | ICE_DAT | PD.1 | MFP0 | I/O | Serial wired debugger data pin |
| PWM | PWM_CH0 | PB.10 | MFP4 | I/O | PWM output/capture input. |
| | PWM_CH0 | PC.9 | MFP3 | I/O | PWM output/capture input. |
| | PWM_CH1 | PB.11 | MFP4 | I/O | PWM output/capture input. |
| | PWM_CH1 | PC.10 | MFP3 | I/O | PWM output/capture input. |
| | PWM_CH2 | PB.12 | MFP4 | I/O | PWM output/capture input. |
| | PWM_CH2 | PC.11 | MFP3 | I/O | PWM output/capture input. |
| | PWM_CH3 | PB.13 | MFP4 | I/O | PWM output/capture input. |
| | PWM_CH3 | PC.12 | MFP3 | I/O | PWM output/capture input. |
| SPIM | SPIM_SS | PA.8 | MFP1 | O | SPIM slave select pin. |
| | SPIM_CLK | PA.9 | MFP1 | O | SPIM serial clock pin. |
| | SPIM_MOSI | PA.10 | MFP1 | I/O | SPIM MOSI (Master Out, Slave In) pin. |
| | SPIM_MISO | PA.11 | MFP1 | I/O | SPIM MISO (Master In, Slave Out) pin. |
| | SPIM_D2 | PA.12 | MFP1 | I/O | SPIM data-2 bit in quad mode. |
| | SPIM_D3 | PA.13 | MFP1 | I/O | SPIM data-3 bit in quad mode. |
| SPI0 | SPI0_SS | PB.2 | MFP1 | O | SPI0 slave select pin. |
| | SPI0_CLK | PB.3 | MFP1 | O | SPI0 serial clock pin. |
| | SPI0_MOSI | PB.4 | MFP1 | I/O | SPI0 MOSI (Master Out, Slave In) pin. |
| | SPI0_MISO | PB.5 | MFP1 | I/O | SPI0 MISO (Master In, Slave Out) pin. |
| SPI1 | SPI1_SS | PB.10 | MFP1 | O | SPI1 slave select pin. |
| | SPI1_CLK | PB.11 | MFP1 | O | SPI1 serial clock pin. |
| | SPI1_MOSI | PB.12 | MFP1 | I/O | SPI1 MOSI (Master Out, Slave In) pin. |
| | SPI1_MISO | PB.13 | MFP1 | I/O | SPI1 MISO (Master In, Slave Out) pin. |
| Timer | TM0_CNT_OUT | PA.12 | MFP2 | I/O | Timer0 event counter input / toggle output. |
| | TM0_EXT | PA.13 | MFP2 | I | Timer0 external counter input |
| | TM1_CNT_OUT | PB.8 | MFP2 | I/O | Timer1 event counter input / toggle output. |
| | TM1_EXT | PB.9 | MFP2 | I | Timer1 external counter input |
| | TM2_CNT_OUT | PC.9 | MFP2 | I/O | Timer2 event counter input / toggle output. |

| Group | Pin Name | GPIO | MFP* | Type | Description |
|---|---|---|---|---|---|
| | TM2_EXT | PC.10 | MFP2 | I | Timer2 external counter input |
| | TM3_CNT_OUT | PC.11 | MFP2 | I/O | Timer3 event counter input / toggle output. |
| | TM3_EXT | PC.12 | MFP2 | I | Timer3 external counter input |
| UART0 | UART0_RXD | PB.1 | MFP3 | I | Data receiver input pin for UART0. |
| | UART0_TXD | PB.0 | MFP3 | O | Data transmitter output pin for UART0. |
| UART1 | UART1_RXD | PA.9 | MFP3 | I | Data receiver input pin for UART1. |
| | UART1_RXD | PB.7 | MFP3 | I | Data receiver input pin for UART1. |
| | UART1_TXD | PA.8 | MFP3 | O | Data transmitter output pin for UART1. |
| | UART1_TXD | PB.6 | MFP3 | O | Data transmitter output pin for UART1. |
| | UART1_nCTS | PB.8 | MFP3 | I | Clear to Send input pin for UART1. |
| | UART1_nRTS | PB.9 | MFP3 | O | Request to Send output pin for UART1. |
| UART2 | UART2_RXD | PB.11 | MFP3 | I | Data receiver input pin for UART2. |
| | UART2_TXD | PB.10 | MFP3 | O | Data transmitter output pin for UART2. |
| | UART2_nCTS | PB.12 | MFP3 | I | Clear to Send input pin for UART2. |
| | UART2_nRTS | PB.13 | MFP3 | O | Request to Send output pin for UART2. |
| USB Host Lite | USBH_PWEN | PB.8 | MFP1 | O | USB host to control an external overcurrent source. |
| | USBH_VOD | PB.9 | MFP1 | I | USB host lite over voltage detector |
| | USBH2_D+ | PC.13 | MFP1 | A | USB host lite 2 differential signal D+. |
| | USBH2_D- | PC.14 | MFP1 | A | USB host lite 2 differential signal D-. |
| | USBH1_D+ | PB.12 | MFP2 | A | USB host lite 1 differential signal D+. |
| | USBH1_D+ | PB.14 | MFP1 | A | USB host lite 1 differential signal D+. |
| | USBH1_D- | PB.13 | MFP2 | A | USB host lite 1 differential signal D-. |
| | USBH1_D- | PB.15 | MFP1 | A | USB host lite 1 differential signal D-. |
| SDH | SD_CLK | PA.10 | MFP4 | O | SD/SDH mode - clock |
| | SD_CLK | PB.3 | MFP4 | O | SD/SDH mode – clock |
| | SD_CLK | PC.1 | MFP1 | O | SD/SDH mode – clock |
| | SD_CMD | PA.11 | MFP4 | O | SD/SDH mode – command/response |
| | SD_CMD | PB.2 | MFP4 | O | SD/SDH mode – command/response |
| | SD_CMD | PC.0 | MFP1 | O | SD/SDH mode – command/response |
| | SD_nCD | PA.13 | MFP4 | I | SD/SDH mode – card detect. |
| | SD_nCD | PB.5 | MFP4 | I | SD/SDH mode – card detect. |
| | SD_nCD | PC.2 | MFP1 | I | SD/SDH mode – card detect. |
| | SD_DAT0 | PA.14 | MFP4 | I/O | SD/SDH mode data line bit 0. |

| Group | Pin Name | GPIO | MFP* | Type | Description |
|---|---|---|---|---|---|
| | SD_DAT0 | PB.6 | MFP4 | I/O | SD/SDH mode data line bit 0. |
| | SD_DAT0 | PC.4 | MFP1 | I/O | SD/SDH mode data line bit 0. |
| | SD_DAT1 | PA.15 | MFP4 | I/O | SD/SDH mode data line bit 1. |
| | SD_DAT1 | PB.7 | MFP4 | I/O | SD/SDH mode data line bit 1. |
| | SD_DAT1 | PC.5 | MFP1 | I/O | SD/SDH mode data line bit 1. |
| | SD_DAT2 | PB.0 | MFP4 | I/O | SD/SDH mode data line bit 2. |
| | SD_DAT2 | PB.8 | MFP4 | I/O | SD/SDH mode data line bit 2. |
| | SD_DAT2 | PC.6 | MFP1 | I/O | SD/SDH mode data line bit 2. |
| | SD_DAT3 | PB.1 | MFP4 | I/O | SD/SDH mode data line bit 3. |
| | SD_DAT3 | PB.9 | MFP4 | I/O | SD/SDH mode data line bit 3. |
| | SD_DAT3 | PC.7 | MFP1 | I/O | SD/SDH mode data line bit 3. |

Table 4.3-1 NUC505 GPIO Multi-function Table

# 5 BLOCK DIAGRAM

## 5.1 NuMicro® NUC505 Series Block Diagram

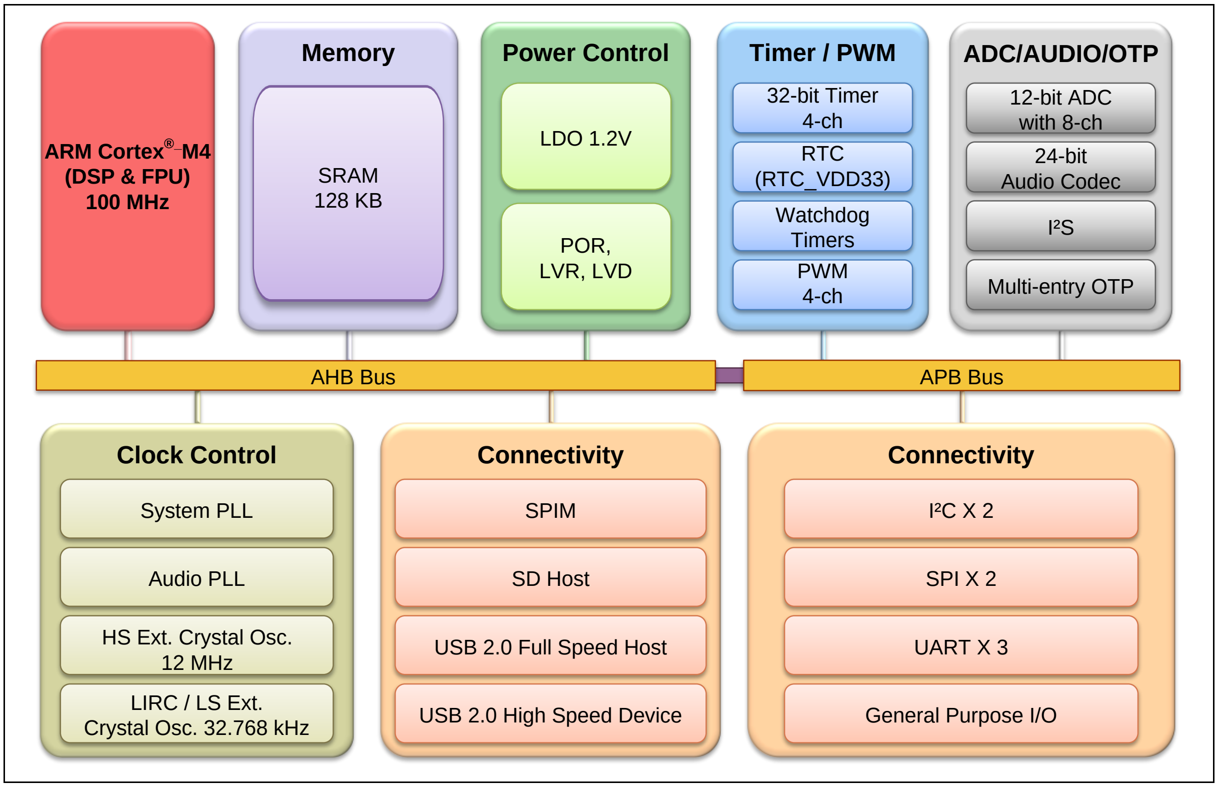
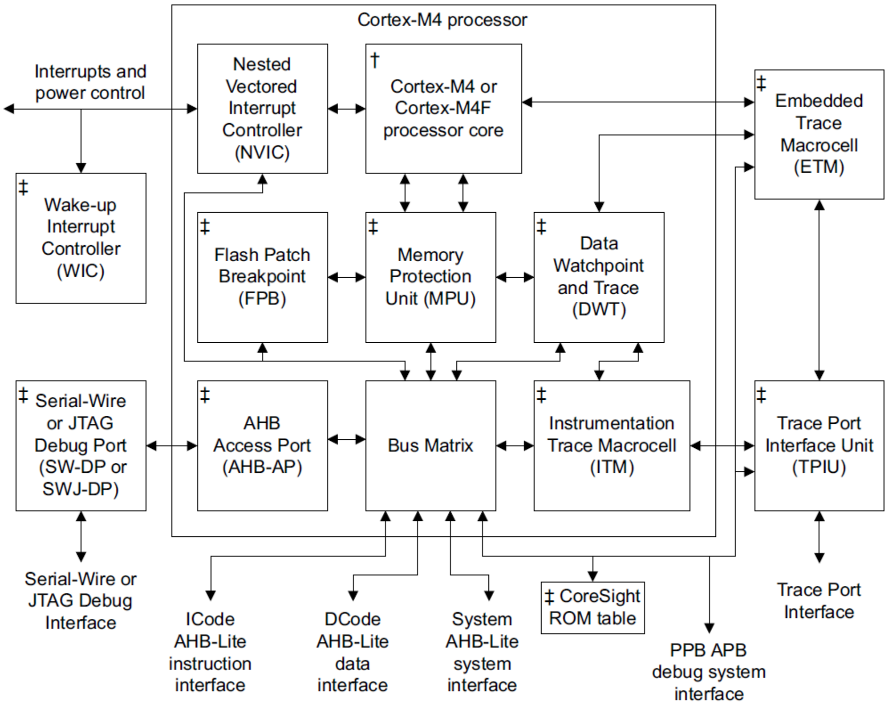


Figure 5.1-1 NuMicro® NUC505 Block Diagram

# 6 FUNCTIONAL DESCRIPTION

## 6.1 ARM® Cortex®-M4 Core

The Cortex®-M4 processor, a configurable, multistage, 32-bit RISC processor, has three AMBA AHB-Lite interfaces for best parallel performance and includes a NVIC component. The processor has optional hardware debug functionality, which can execute Thumb code, and is compatible with other Cortex®-M profile processors. The profile supports two modes -Thread mode and Handler mode. Handler mode is entered as a result of an exception. An exception return can only be issued in Handler mode. Thread mode is entered on Reset, and can be entered as a result of an exception return. The Cortex®-M4F is a processor with the same capability as the Cortex®-M4 processor and includes floating point arithmetic functionality. The NUC505 is embedded with Cortex®-M4F processor. Throughout this document the name Cortex®-M4 refers to both Cortex®-M4 and Cortex®-M4F processors. The following figure shows the functional controller of the processor.


Cortex-M4 processor
Interrupts and power control
Nested Vectored Interrupt Controller (NVIC)
† Cortex-M4 or Cortex-M4F processor core
‡ Embedded Trace Macrocell (ETM)
‡ Wake-up Interrupt Controller (WIC)
‡ Flash Patch Breakpoint (FPB)
‡ Memory Protection Unit (MPU)
‡ Data Watchpoint and Trace (DWT)
‡ Serial-Wire or JTAG Debug Port (SW-DP or SWJ-DP)
‡ AHB Access Port (AHB-AP)
Bus Matrix
‡ Instrumentation Trace Macrocell (ITM)
‡ Trace Port Interface Unit (TPIU)
Serial-Wire or JTAG Debug Interface
ICode AHB-Lite instruction interface
DCode AHB-Lite data interface
System AHB-Lite system interface
‡ CoreSight ROM table
PPB APB debug system interface
Trace Port Interface

† For the Cortex-M4F processor, the core includes a Floating Point Unit (FPU)
‡ Optional component

Figure 6.1-1 Cortex®-M4 Block Diagram

Cortex®-M4 processor features:

- A low gate count processor core, with low latency interrupt processing that has:
  - A subset of the Thumb instruction set, defined in the ARMv7-M Architecture Reference Manual.

  - Banked Stack Pointer (SP).
  - Hardware integer divide instructions, SDIV and UDIV.
  - Handler and Thread modes.
  - Thumb and Debug states.
  - Support for interruptible-continued instructions LDM, STM, PUSH, and POP for low interrupt latency.
  - Automatic processor state saving and restoration for low latency Interrupt Service Routine (ISR) entry and exit.
  - Support for ARMv6 big-endian byte-invariant or little-endian accesses.
  - Support for ARMv6 unaligned accesses.
- Floating Point Unit (FPU) in the Cortex®-M4F processor providing:
  - 32-bit instructions for single-precision (C float) data-processing operations.
  - Combined Multiply and Accumulate instructions for increased precision (Fused MAC).
  - Hardware support for conversion, addition, subtraction, multiplication with optional accumulate, division, and square-root.
  - Hardware support for denormals and all IEEE rounding modes.
  - 32 dedicated 32-bit single precision registers, also addressable as 16 double-word registers.
  - Decoupled three-stage pipeline.
- Nested Vectored Interrupt Controller (NVIC) closely integrated with the processor core to achieve low latency interrupt processing. Features include:
  - External interrupts. Configurable from 1 to 240; the NUC505 has been configured with 32 interrupts.
  - Bits of priority, configurable from bit 3 to bit 7.
  - Dynamic reprioritization of interrupts.
  - Supports priority grouping which enables selection of preempting interrupt levels and non-preempting interrupt levels.
  - Supports tril-chaining and late arrival of interrupts, which enables back-to- back interrupt processing without the overhead of state saving and restoration between interrupts.
  - Processor state automatically saved on interrupt entry, and restored on interrupt exit with on instruction overhead.
  - Supports Wake-up Interrupt Controller (WIC) with Power-down mode.
- Memory Protection Unit (MPU). An optional MPU for memory protection, including:
  - Eight memory regions.
  - Sub Region Disable (SRD), enabling efficient use of memory regions.
  - The ability to enable a background region that implements the default memory map attributes.
- Low-cost debug solution that features:

  - Debug access to all memory and registers in the system, including access to memory mapped devices, access to internal core registers when the core is halted, and access to debug control registers even while SYSRESETn is asserted.
  - Serial Wire Debug Port (SW-DP) or Serial Wire JTAG Debug Port (SWJ-DP) debug access. But NUC505 only supports SW-DP.
  - Optional Flash Patch and Breakpoint (FPB) unit for implementing breakpoints and code patches.
  - Optional Data Watchpoint and Trace (DWT) unit for implementing watchpoints, data tracing, and system profiling.
  -

- Bus interfaces:
  - Three Advanced High-performance Bus-Lite (AHB-Lite) interfaces: ICode, Dcode, and System bus interfaces.

  Private Peripheral Bus (PPB) based on Advanced Peripheral Bus (APB) interface.
  - Bit-band support that includes atomic bit-band write and read operations.
  - Memory access alignment.
  - Write buffer for buffering of write data.
  - Exclusive access transfers for multiprocessor systems.

## 6.2 System Manager

### 6.2.1 Overview

The following functions are included in system manager section

- System reset
- System memory map
- Bus arbitration algorithm
- Global control registers
- System Timer (Systick)
- Nested Vectored Interrupt Control (NVIC)
- System control register map and description

### 6.2.2 System Reset

- Hardware Reset
  - Power-on Reset (POR)
  - Low level on the nRESET Pin (nRST)
  - Watchdog time-out reset (WDT)
  - Low voltage reset (LVR)
- Software Reset
  - SYSRESETREQ (AIRCR[2])
  - CPU Reset (SYS_IPRST0[0])
  - CHIPRST (SYS_IPRST0 [1])

**Note1:** SYSRESETREQ (AIRCR[2]) reset the whole chip including all peripherals, but does not reset SPIM function, vector map module parameter setting, and PA.8~PA.15 multi-function setting.

**Note2:** CPU Rest (SYS_IPRST0[0]) only resets the CPU function.

**Note3:** CHIPRST (SYS_IPRST0[1]) reset the whole chip including all peripherals.

### 6.2.3 System Power-on Setting

The power-on setting is used to configure the chip to enter the specified state when the chip is powered up or reset. Since each pin of power-on setting has an internal pulled-up resistor during reset period, if the application needs to set the configuration to “0”, the proper pull-down must be added for the corresponding configuration pins.

| PB.4 | PB.3 | PA.10 | PA.9 | Description | Register Mapping |
|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | Boot from Internal MCP SPI Flash | SYS_BOOTSET[3:0] |
| 1 | 1 | 1 | 0 | Boot from USB | SYS_BOOTSET[3:0] |
| 1 | 1 | 0 | 1 | Boot from External SPI Flash | SYS_BOOTSET[3:0] |
| 1 | 0 | 1 | 1 | Boot from ICP Mode | SYS_BOOTSET[3:0] |
| 0 | 1 | 1 | 1 | SWD/ICE Mode with Internal SPI Flash | SYS_BOOTSET[3:0] |
| 0 | 1 | 1 | 0 | SWD/ICE Mode with External SPI Flash | SYS_BOOTSET[3:0] |

Table 6.2-1 System Power-on Setting Guide

### 6.2.4 System Power Distribution

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

- Audio CODEC power from $AV_{DDCODEC,}$ $AV_{DDHP}$, and $AV_{SSHP}$ provides the power for audio CODEC operation.
- Analog-to-Digital converter (ADC) power from $AV_{DDADC}$ and $AV_{SSADC}$ provides the power for ADC operation.
- Digital power from $V_{DD}$ and $V_{SS}$ supplies the power to the internal regulator which provides a fixed 1.2 V power for digital operation and I/O pins.
- USB transceiver power from $AV_{DDUSB}$ offers the power for operating the USB transceiver.
- RTC power from $V_{BAT}$ provides the power for RTC and 80 bytes backup registers.

The outputs of internal voltage regulators, LDO and $V_{DD}$, require an external capacitor which should be located close to the corresponding pin. Analog power ($AV_{DDCODEC}$ and $AV_{DDADC}$) should be the same voltage level of the digital power ($V_{DD}$). The following figure shows the power distribution of the NuMicro® NUC505.

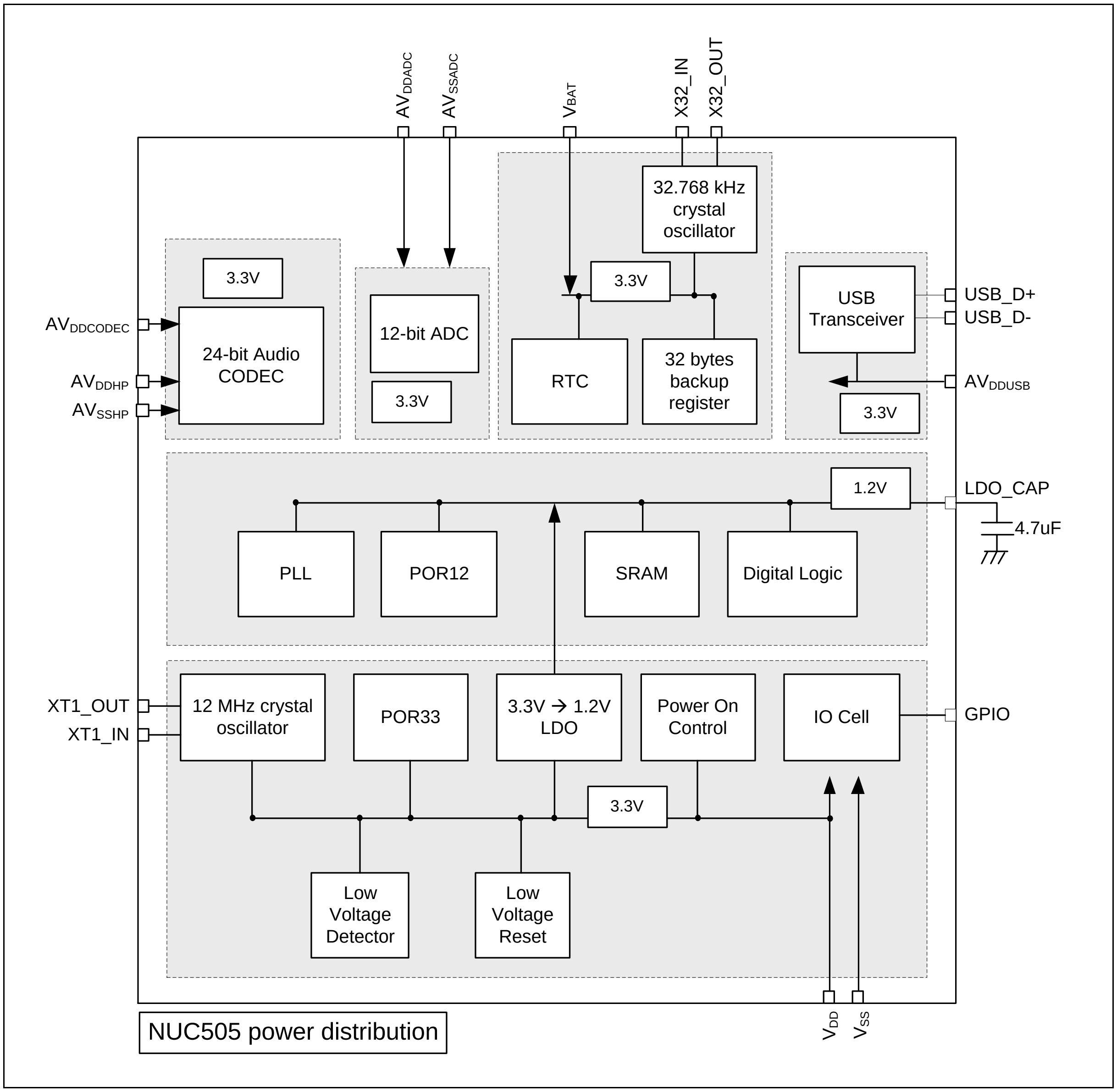


Figure 6.2-1 NuMicro® NUC505 Power Distribution Diagram

### 6.2.5 System Memory Mapping

The NUC505 provides a 4G-byte address space for programmers. The memory locations assigned to each on-chip modules are shown in Table 6.2-2. The detailed registers and memory addressing or programming will be described in the following sections for individual on-chip modules. The NUC505 only supports little-endian data format.

| Address Space | Token | Modules |
|---|---|---|
| **Memory Space** | | |
| 0x1FFF_0000 – 0x1FFF_7FFF | IBR_BA | Internal Boot ROM (IBR) Memory Space |

| | | |
|---|---|---|
| 0x2000_0000 – 0x2000_7FFF | SRAM1_BA | SRAM1 Memory Space (32K Bytes) |
| 0x2000_8000 – 0x2000_FFFF | SRAM2_BA | SRAM2 Memory Space (32K Bytes) |
| 0x2001_0000 – 0x2001_7FFF | SRAM3_BA | SRAM3 Memory Space (32K Bytes) |
| 0x2001_8000 – 0x2001_FFFF | SRAM4_BA | SRAM4 Memory Space (32K Bytes) |
| 0x0000_0000 – 0x0FFF_FFFF | FLASH_BA | SPI Flash/ROM Memory Space |
| **AHB Controllers Space (0x4000_0000 ~0x4000FFFF)** | | |
| 0x4000_0000 – 0x4000_01FF | GCR_BA | Global Control Registers |
| 0x4000_0200 – 0x4000_02FF | CLK_BA | Clock Control Registers |
| 0x4000_7000 – 0x4000_7FFF | SPIM_BA | SPIM Control Register |
| 0x4000_9000 – 0x4000_9FFF | USBD_BA | USB Device Controller Registers |
| 0x4000_A000 – 0x4000_AFFF | SDH_BA | SDH Control Register |
| 0x4000_B000 – 0x4000_BFFF | USBH_BA | USB Host Controller Registers |
| **APB Controllers Space (0x400E_0000~0x400E_FFFF)** | | |
| 0x400E_1000 – 0x400E_1FFF | SPI1_BA | SPI1 Master/Slave Controller Registers (SPI1) |
| 0x400E_2000 – 0x400E_2FFF | ADC_BA | ADC Controller Registers |
| 0x400E_3000 – 0x400E_3FFF | GPIO_BA | GPIO Controller Registers |
| 0x400E_4000 – 0x400E_4FFF | I2C0_BA | I2C0 Interface Control Registers |
| 0x400E_5000 – 0x400E_5FFF | I2C1_BA | I2C1 Interface Control Registers |
| 0x400E_6000 – 0x400E_6FFF | PWM_BA | PWM Controller Registers |
| 0x400E_7000 – 0x400E_7FFF | RTC_BA | Real Time Clock (RTC) Control Register |
| 0x400E_8000 – 0x400E_8FFF | I2S_BA | Inter-IC Sound (I$^2$S) Control Register |
| 0x400E_9000 – 0x400E_9FFF | SPI0_BA | SPI0 Master/Slave Controller Registers (SPI0) |
| 0x400E_A000 – 0x400E_AFFF | Timer01_BA | Timer0/Timer1 Control Registers |
| 0x400E_B000 – 0x400E_BFFF | Timer23_BA | Timer2/Timer3 Control Registers |
| 0x400E_C000 – 0x400E_CFFF | UART0_BA | UART0 Control Registers (Normal Speed) |
| 0x400E_D000 – 0x400E_DFFF | UART1_BA | UART1 Control Registers (High Speed) |
| 0x400E_E000 – 0x400E_EFFF | UART2_BA | UART2 Control Registers (High Speed) |
| 0x400E_F000 – 0x400E_FFFF | WDT_BA | WDT Interface Control Registers |
| **System Controllers Space (0xE000_E000 ~ 0xE000_EFFF)** | | |
| 0xE000_E010 – 0xE000_E0FF | SCS_BA | System Timer Control Registers |
| 0xE000_E100 – 0xE000_ECFF | SCS_BA | External Interrupt Controller Control Registers |
| 0xE000_ED00 – 0xE000_ED8F | SCS_BA | System Control Registers |

Table 6.2-2 Address Space Assignments for On-Chip Controllers

### 6.2.6 SRAM Memory Organization

The NUC505 supports embedded SRAM with a total of 128 Kbytes and the SRAM organization is separated to four banks: SRAM bank0, SRAM bank1, SRAM bank2, and SRAM bank3. Each of these four banks has 32 Kbytes address space and can be accessed simultaneously.

- Supports a total of 128 Kbytes SRAM
- Supports byte / half word / word write
- Supports fixed 32 Kbytes SRAM banks for independent access
- Supports remap address to 0x1FF0_0000
- Supports remap arbitrary memory block of 128 Kbytes SRAM to 0x0000_0000 by using vector map module

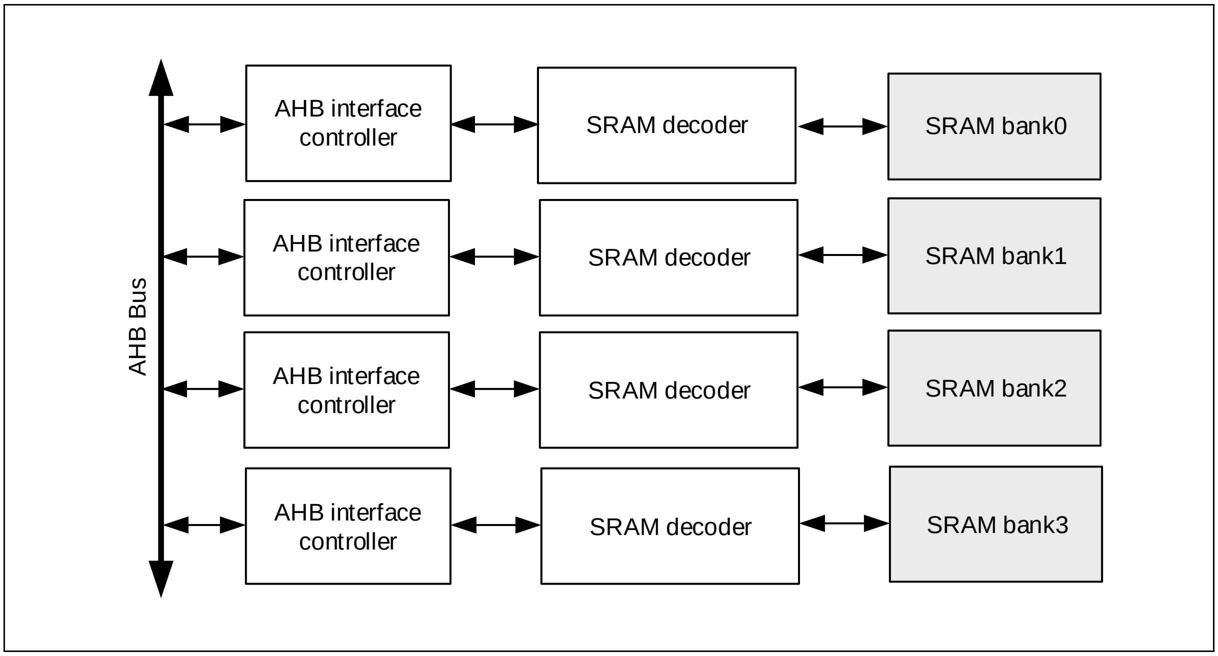


Figure 6.2-2 SRAM Block Diagram

Figure 6.2-3 shows the SRAM organization of NUC505. There are four SRAM banks in NUC505 and each bank is addressed to 32 Kbytes. The bank0 address space is from 0x2000_0000 to 0x2000_7FFF. The bank1 address space is from 0x2000_8000 to 0x2000_FFFF. The bank2 address space is from 0x2001_0000 to 0x2001_7FFF. The bank3 address space is from 0x2001_8000 to 0x2001_FFFF.

The address of each bank is remapping from 0x2000_0000 to 0x1FF0_0000. CPU can access SRAM bank0 through 0x2000_0000 to 0x2000_7FFF or 0x1FF0_0000 to 0x1FF0_7FFF, SRAM bank1 through 0x2000_8000 to 0x2000_FFFF or 0x1FF0_8000 to 0x1FF0_FFFF, SRAM bank2 through 0x2001_0000 to 0x2001_7FFF or 0x1FF1_0000 to 0x1FF1_7FFF, and SRAM bank3 through 0x2001_8000 to 0x2001_FFFF or 0x1FF1_8000 to 0x1FF1_FFFF.

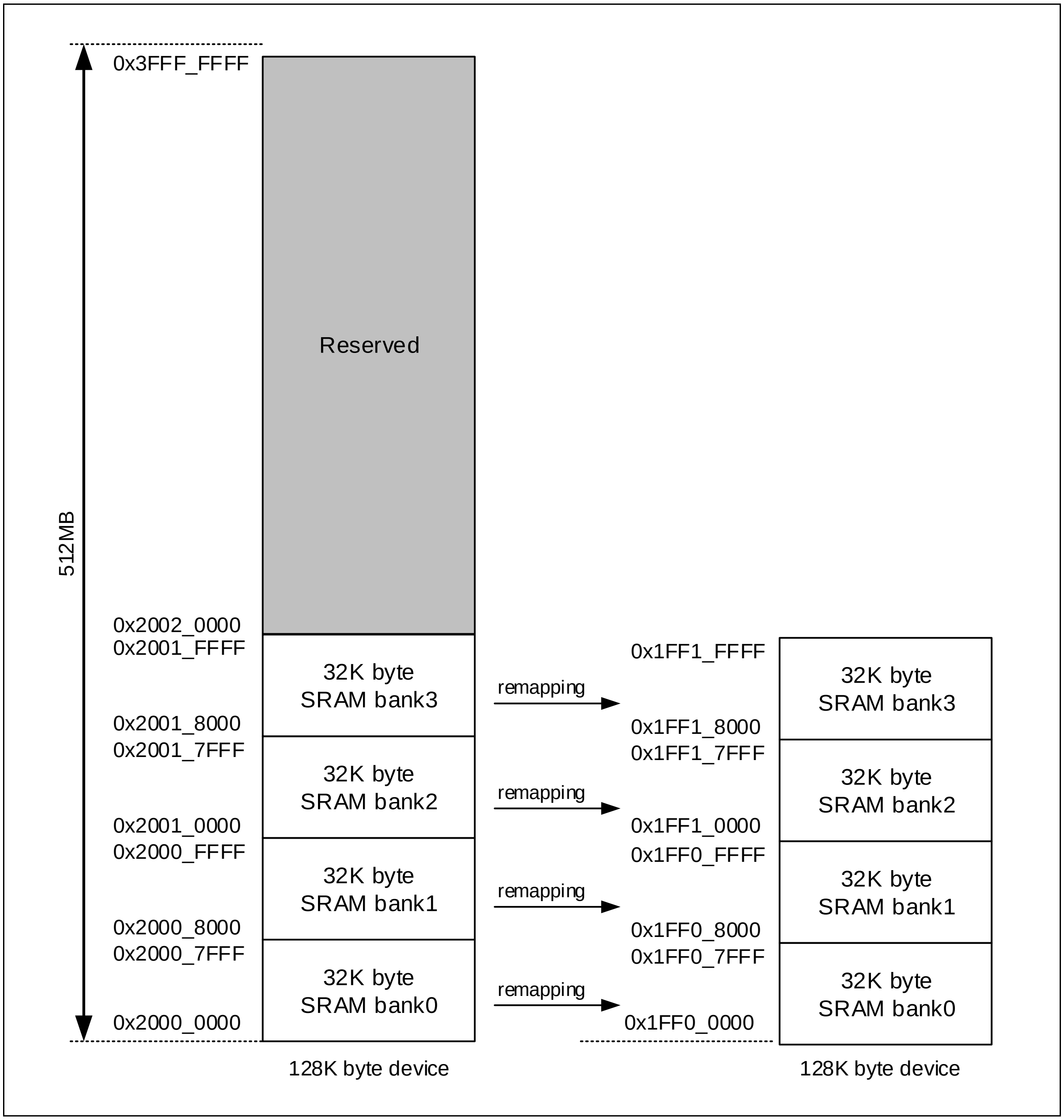


Figure 6.2-3 SRAM Memory Organization

Figure 6.2-4 shows the vector map module diagram. Arbitrary memory block in 128 Kbytes SRAM can be remapped to the SPI flash block and its start address is 0x0000_0000. The location and size with the memory block are controlled by the register SYS_RVMPADDR[31:0] and the register SYS_RVMPLEN[31:24]. The SYS_RVMPADDR indicates the start address of the memory block and SYS_RVMPLEN describes about the size of the memory block (the unit is 1 Kbyte).

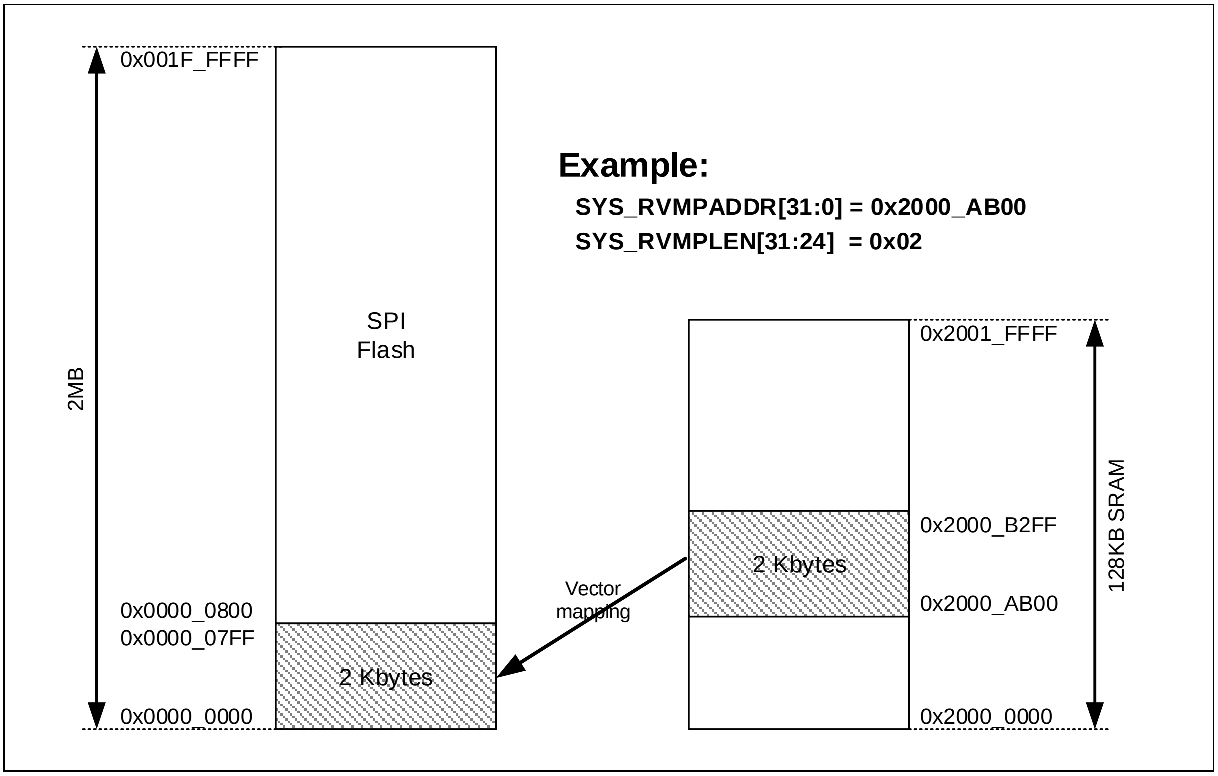


Figure 6.2-4 Vector Map Module Block

### 6.2.7 AHB Bus Arbitration

The internal bus of NUC505 is an AHB-Compliant Bus and supports to connect with the standard AHB master or slave. The NUC505 AHB arbiter provides a choice of two arbitration algorithms for simultaneous requests. These two arbitration algorithms are the Fixed-priority mode and the Round-robin- priority (rotate) mode. The selection of modes and types is determined in the **PRISEL** field of the ***SYS_AHBCTL*** control register.

#### *6.2.7.1 Fixed Priority Mode*

Fixed priority mode is selected if **PRISEL** = 0. The order of priorities on the AHB mastership among the on-chip master modules are listed in Table 6.2-3.

| **Priority Sequence<br>(PRISEL = 0)** | **AHB Bus Priority** |
|---|---|
| 1 (Lowest) | Cortex-M4 I |
| 2 | Cortex-M4 D |
| 3 | Cortex-M4 System |
| 4 | SPIM |
| 5 | USBD |

| | |
|---|---|
| 6 | USBH |
| 6 | SDH |
| 8 (Highest) | $I^2S$ |

Table 6.2-3 AHB Bus Priority Order in Fixed Priority Mode If two or more master modules request to access AHB bus at the same time, the higher priority request will get the permission to access AHB bus.

| **Priority Sequence**<br>**(PRISEL = 0)** | **AHB Bus Priority** |
|---|---|
| 1 (Lowest) | Cortex-M4 I |
| 2 | Cortex-M4 D |
| 3 | Cortex-M4 System |
| 4 | SPIM |
| 5 | USBD |
| 6 | USBH |
| 6 | SDH |
| 8 (Highest) | $I^2S$ |

Table 6.2-3 AHB Bus Priority Order in Fixed Priority Mode

The SPI flash controller normally has the lowest priority (except CPU interface) under the fixed priority mode. The NUC505 provides a mechanism to raise the priority of CPU request to the highest. If the **CPUHPRI** bit (bit-4 of ***SYS_AHBCTL*** *control register*) is set to 1, the **PRISTS** bit (bit-5 of ***SYS_AHBCTL*** *control register*) will be automatically set to 1 while an unmasked external IRQ occurs. Under this circumstance, the ARM core will become the highest priority to access AHB bus.

The programmer can recover the original priority order by directly writing “1” to clear the **PRISTS** bit. For example, this can be done that at the end of an interrupt service routine. Note that **PRISTS** only can be automatically set to 1 by an external interrupt when **CPUHPRI** = 1. It will not take effect for a programmer to directly write 1 to **PRISTS** to raise ARM core’s AHB priority.

### *6.2.7.2 Round Robin Priority Mode*

Round-robin priority mode is selected if **PRISEL** = 1. The AHB bus arbiter uses a round robin arbitration scheme for every master module to gain the bus ownership in turn. That is the requestor having the highest priority becomes the lowest-priority requestor after it has been granted access**.**

### *6.2.7.3 Rotate rule Example*

In the default sequence of AHB Master Bus, the priority is $I^2S$>SDH > USBH > USBD >SPIM >M4(S) > M4(D) > M4(I).

### 6.2.8 Registers Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GCR Base Address:**<br>**GCR_BA = 0x4000_0000** | | | | |
| **SYS_PDID** | GCR_BA+0x00 | R | Part Device Identification Number Register | 0x0055_0505 |
| **SYS_BOOTSET** | GCR_BA+0x04 | R/W | System Power-on Configuration Register | 0x0000_00XX |
| **SYS_IPRST0** | GCR_BA+0x08 | R/W | Peripheral Reset Control Register 0 | 0x0000_0009 |
| **SYS_LVDCTL** | GCR_BA+0x0C | R/W | Low Voltage Detection Control Register | 0x0000_0009 |
| **SYS_WAKEUP** | GCR_BA+0x10 | R/W | Wake-Up Control and Status Resister | 0x0000_0000 |
| **SYS_IPRST1** | GCR_BA+0x14 | R/W | Peripheral Reset Control Register 1 | 0x0000_0000 |
| **SYS_NMICTL** | GCR_BA+0x18 | R/W | Non Maskable Interrupt Control Register | 0x0000_0000 |
| **SYS_RSTSTS** | GCR_BA+0x1C | R/W | Reset Status Control Register | 0x0000_0000 |
| **SYS_AHBCTL** | GCR_BA+0x20 | R/W | AHB Bus Control Register | 0x0000_0000 |
| **SYS_GPA_MFPL** | GCR_BA+0x30 | R/W | GPIOA Low Byte Multi-function Control Register | 0x0000_0000 |
| **SYS_GPA_MFPH** | GCR_BA+0x34 | R/W | GPIOA High Byte Multi-function Control Register | 0x0000_0000 |
| **SYS_GPB_MFPL** | GCR_BA+0x38 | R/W | GPIOB Low Byte Multi-function Control Register | 0x0000_0000 |
| **SYS_GPB_MFPH** | GCR_BA+0x3C | R/W | GPIOB High Byte Multi-function Control Register | 0x0000_0000 |
| **SYS_GPC_MFPL** | GCR_BA+0x40 | R/W | GPIOC Low Byte Multi-function Control Register | 0x0000_0000 |
| **SYS_GPC_MFPH** | GCR_BA+0x44 | R/W | GPIOC High Byte Multi-function Control Register | 0x0000_0000 |
| **SYS_GPD_MFPL** | GCR_BA+0x48 | R/W | GPIOD Low Byte Multi-function Control Register | 0x0000_0000 |
| **SYS_LVMPADDR** | GCR_BA+0x50 | R/W | Load VECMAP Address Parameter Control Register | 0x1FFF_0000 |
| **SYS_LVMPLEN** | GCR_BA+0x54 | R/W | Load VECMAP Length Parameter Control Register | 0x0000_0008 |
| **SYS_RVMPADDR** | GCR_BA+0x58 | R | Real VECMAP Address Parameter Register | 0x1FFF_0000 |
| **SYS_RVMPLEN** | GCR_BA+0x5C | R/W | Real VECMAP Length Parameter Control Register | 0x0800_0000 |
| **SYS_EPADPUEN** | GCR_BA+0x6C | R/W | Embedded SPI Flash Pad Pull-up Enable Control Register | 0x0000_001F |
| **SYS_GPADS** | GCR_BA+0x70 | R/W | GPIOA Driving Strength Control Register | 0x0000_0000 |
| **SYS_GPAIBE** | GCR_BA+0x74 | R/W | GPIOA Input Buffer Enable Control Register | 0xFFFF_0000 |
| **SYS_GPBIBE** | GCR_BA+0x78 | R/W | GPIOB Input Buffer Enable Control Register | 0xFFFF_0000 |
| **SYS_GPCIBE** | GCR_BA+0x7C | R/W | GPIOC Input Buffer Enable Control Register | 0xFFFF_0000 |
| **SYS_GPDIBE** | GCR_BA+0x80 | R/W | GPIOD Input Buffer Enable Control Register | 0xFFFF_0000 |
| **SYS_GPDDS** | GCR_BA+0x84 | R/W | GPIOD Driving Strength Control Register | 0x0000_0000 |
| **SYS_RSTDBCNT** | GCR_BA+0x100 | R/W | External nRESET Pin De-bounce Counter Control Register | 0x0000_04B0 |

| | | | | |
|---|---|---|---|---|
| **SYS_RSTDBEN** | GCR_BA+0x104 | R/W | External nRESET Pin De-bounce Control Register | 0x0000_0000 |

### 6.2.9 Register Description

**Part Device Identification Number Register (SYS_PDID)**

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_PDID** | GCR_BA+0x00 | R | Part Device Identification Number Register | 0x0055_0505 |

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

| Bits | Description | |
|---|---|---|
| [31:20] | **Reserved** | Reserved. |
| [23:0] | **PDID** | **Part Device Identification Number (Read Only)**<br>This register reflects device part number code. Software can read this register to identify which device is used. |

### System Power-on Configuration Register (SYS_BOOTSET)

This register provides specific information for software to identify this chip's power-on setting. BOOTSET[3:0] are the status of the power-on setting pins. They can be modified by software programming.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_BOOTSET** | GCR_BA+0x04 | R/W | System Power-on Configuration Register | 0x0000_00XX |

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

| Bits | Description | |
|---|---|---|
| [31:4] | **Reserved** | Reserved. |
| [3:0] | **BOOTSET** | **System Mode Configuration**<br>0110 = Boot from ICE Mode with external SPI Flash.<br>0111 = Boot from ICE Mode with SPI Flash.<br>1011 = Boot from ICP Mode.<br>1101 = Boot from external SPI Flash.<br>1110 = Boot from USB.<br>1111 = Boot from SPI Flash.<br>**Note:** If BOOTSET is equal to ICE Mode, the software cannot change BOOTSET to other mode. But other modes don't have this limitation. |

### Peripheral Reset Control Register 0 (SYS_IPRST0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_IPRST0** | GCR_BA+0x08 | R/W | Peripheral Reset Control Register 0 | 0x0000_0009 |

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

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1] | **CHIPRST** | **Chip One-shot Reset**<br>Setting this bit will reset the whole chip, including processor core and all peripherals, and this bit will automatically return to 0 after the 2 clock cycles.<br>The CHIPRST is same as the POR reset, all the chip controllers is reset and the chip setting from flash are also reload.<br>0 = Chip normal operation.<br>1 = Chip one-shot reset. |
| [0] | **CPURST** | **Processor Core One-shot Reset**<br>Setting this bit will only reset the processor core, and this bit will automatically return to 0 after the 2 clock cycles.<br>0 = Processor core normal operation.<br>1 = Processor core one-shot reset. |

### Low Voltage Detection Control Register (SYS_LVDCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_LVDCTL** | GCR_BA+0x0C | R/W | Low Voltage Detection Control Register | 0x0000_0009 |

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

| Bits | Description | |
|---|---|---|
| [31:5] | **Reserved** | Reserved. |
| [4] | **PORENB** | **Power on Reset Enable Control**<br>0 = Function Enabled. (Default)<br>1 = Function Disabled. |
| [3] | **LVREN** | **Low Voltage Reset Enable Control**<br>0 = Low Voltage Reset Disabled.<br>1 = Low Voltage Reset Enabled. (default)<br>**Note:** The voltage threshold level is 2.4V |
| [2] | **LVDEN** | **Low Voltage Detection Enable Control**<br>0 = Detection Disabled.<br>1 = Detection Enabled. |
| [1] | **LVDSEL** | **Low Voltage Detection Level Selection**<br>0 = The threshold level is 2.6V.<br>1 = The threshold level is 2.8V. |
| [0] | **LVDIF** | **Low Voltage Detect Flag**<br>0 = Low voltage Period.<br>1 = Normal voltage Period.<br>**Note:** This bit is useful when LVDEN is enabled. |

### Wake-Up Control and Status Register (SYS_WAKEUP)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_WAKEUP | GCR_BA+0x10 | R/W | Wake-Up Control and Status Resister | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | USBHWF | USBDWF | UART2WF | UART1WF | UART0WF | TMR3WF |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TMR2WF | TMR1WF | TMR0WF | WDTWF | RTCWF | GPIOWF | I2C1WF | I2C0WF |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | USBHWE | USBDWE | UART2WE | UART1WE | UART0WE | TMR3WE |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TMR2WE | TMR1WE | TMR0WE | WDTWE | RTCWE | GPIOWE | I2C1WE | I2C0WE |

| Bits | Description | |
|---|---|---|
| [31:30] | **Reserved** | Reserved. |
| [29] | **USBHWF** | **USB Host Wake-up Flag**<br>0 = USB Host wake-up source is deasserted.<br>1 = USB Host wake-up source is asserted.<br>**Note:** Write 1 to clear this bit |
| [28] | **USBDWF** | **USB Device Wake-up Flag**<br>0 = USB Device wake-up source is deasserted.<br>1 = USB Device wake-up source is asserted.<br>**Note:** Write 1 to clear this bit |
| [27] | **UART2WF** | **UART2 Wake-up Flag**<br>0 = UART2 wake-up source is deasserted.<br>1 = UART2 wake-up source is asserted.<br>**Note:** Write 1 to clear this bit |
| [26] | **UART1WF** | **UART1 Wake-up Flag**<br>0 = UART1 wake-up source is deasserted.<br>1 = UART1 wake-up source is asserted.<br>**Note:** Write 1 to clear this bit |
| [25] | **UART0WF** | **UART0 Wake-up Flag**<br>0 = UART0 wake-up source is deasserted.<br>1 = UART0 wake-up source is asserted.<br>**Note:** Write 1 to clear this bit |

| | | |
|---|---|---|
| [24] | **TMR3WF** | **Timer3 Wake-up Flag**<br>0 = Timer3 wake-up source is deasserted.<br>1 = Timer3 wake-up source is asserted.<br>**Note:** Write 1 to clear this bit |
| [23] | **TMR2WF** | **Timer2 Wake-up Flag**<br>0 = Timer2 wake-up source is deasserted.<br>1 = Timer2 wake-up source is asserted.<br>**Note:** Write 1 to clear this bit |
| [22] | **TMR1WF** | **Timer1 Wake-up Flag**<br>0 = Timer1 wake-up source is deasserted.<br>1 = Timer1 wake-up source is asserted.<br>**Note:** Write 1 to clear this bit |
| [21] | **TMR0WF** | **Timer0 Wake-up Flag**<br>0 = Timer0 wake-up source is deasserted.<br>1 = Timer0 wake-up source is asserted.<br>**Note:** Write 1 to clear this bit |
| [20] | **WDTWF** | **WDT Wake-up Flag**<br>0 = WDT wake-up source is deasserted.<br>1 = WDT wake-up source is asserted.<br>**Note:** Write 1 to clear this bit |
| [19] | **RTCWF** | **RTC Wake-up Flag**<br>0 = RTC wake-up source is deasserted.<br>1 = RTC wake-up source is asserted.<br>**Note:** Write 1 to clear this bit |
| [18] | **GPIOWF** | **GPIO Wake-up Flag**<br>0 = GPIO wake-up source is deasserted.<br>1 = GPIO wake-up source is asserted.<br>**Note:** Write 1 to clear this bit |
| [17] | **I2C1WF** | **I2C1 Wake-up Flag**<br>0 = I2C1 wake-up source is deasserted.<br>1 = I2C1 wake-up source is asserted.<br>**Note:** Write 1 to clear this bit |
| [16] | **I2C0WF** | **I2C0 Wake-up Flag**<br>0 = I2C0 wake-up source is deasserted.<br>1 = I2C0 wake-up source is asserted.<br>**Note:** Write 1 to clear this bit |
| [15:14] | **Reserved** | Reserved. |
| [13] | **USBHWE** | **USB Host Wake-up Enable Control**<br>0 = USB Host wake-up Disabled.<br>1 = USB Host wake-up Enabled. |
| [12] | **USBDWE** | **USB Device Wake-up Enable Control**<br>0 = USB Device wake-up Disabled.<br>1 = USB Device wake-up Enabled. |

| | | |
|---|---|---|
| [11] | **UART2WE** | **UART2 Wake-up Enable Control**<br>0 = UART2 wake-up Disabled.<br>1 = UART2 wake-up Enabled. |
| [10] | **UART1WE** | **UART1 Wake-up Enable Control**<br>0 = UART1 wake-up Disabled.<br>1 = UART1 wake-up Enabled. |
| [9] | **UART0WE** | **UART0 Wake-up Enable Control**<br>0 = UART0 wake-up Disabled.<br>1 = UART0 wake-up Enabled. |
| [8] | **TMR3WE** | **Timer3 Wake-up Enable Control**<br>0 = Timer3 wake-up Disabled.<br>1 = Timer3 wake-up Enabled. |
| [7] | **TMR2WE** | **Timer2 Wake-up Enable Control**<br>0 = Timer2 wake-up Disabled.<br>1 = Timer2 wake-up Enabled. |
| [6] | **TMR1WE** | **Timer1 Wake-up Enable Control**<br>0 = Timer1 wake-up Disabled.<br>1 = Timer1 wake-up Enabled. |
| [5] | **TMR0WE** | **Timer0 Wake-up Enable Control**<br>0 = Timer0 wake-up Disabled.<br>1 = Timer0 wake-up Enabled. |
| [4] | **WDTWE** | **WDT Wake-up Enable Control**<br>0 = WDT wake-up Disabled.<br>1 = WDT wake-up Enabled. |
| [3] | **RTCWE** | **RTC Wake-up Enable Control**<br>0 = RTC wake-up Disabled.<br>1 = RTC wake-up Enabled. |
| [2] | **GPIOWE** | **GPIO Wake-up Enable Control**<br>0 = GPIO wake-up Disabled.<br>1 = GPIO wake-up Enabled. |
| [1] | **I2C1WE** | **I2C1 Wake-up Enable Control**<br>0 = I2C1 wake-up Disabled.<br>1 = I2C1 wake-up Enabled. |
| [0] | **I2C0WE** | **I2C0 Wake-up Enable Control**<br>0 = I2C0 wake-up Disabled.<br>1 = I2C0 wake-up Enabled. |

### Peripheral Reset Control Register (SYS_IPRST1)

This register provides specific read-only information for software to identify this chip.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_IPRST1** | GCR_BA+0x14 | R/W | Peripheral Reset Control Register 1 | 0x0000_0000 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| **31** | **30** | **29** | **28** | **27** | **26** | **25** | **24** |
| **SPI1RST** | **SPI0RST** | **Reserved** | **ADCRST** | **GPIORST** | **Reserved** | **SRAMRST** | **SDHRST** |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | **USBHRST** | **I2SRST** | **Reserved** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | **WDTPRST** | **TMR3RST** | **USBDRST** | **SPIMRST** | **I2C1RST** | **I2C0RST** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **PWMRST** | **UART2RST** | **TMR2RST** | **WDTFRST** | **TMR1RST** | **TMR0RST** | **UART1RST** | **UART0RST** |

| Bits | Description | |
|---|---|---|
| [31] | **SPI1RST** | **SPI1 Controller Reset**<br>Set this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = SPI1 controller normal operation.<br>1 = SPI1 controller reset. |
| [30] | **SPI0RST** | **SPI0 Controller Reset**<br>Set this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = SPI0 controller normal operation.<br>1 = SPI0 controller reset. |
| [29] | **Reserved** | Reserved. |
| [28] | **ADCRST** | **ADC Controller Reset**<br>Set this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = ADC controller normal operation.<br>1 = ADC controller reset. |
| [27] | **GPIORST** | **GPIO Controller Reset**<br>Set this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = GPIO controller normal operation.<br>1 = GPIO controller reset. |
| [26] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [25] | **SRAMRST** | **SRAM Controller Reset**<br>Set this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = SRAM controller normal operation.<br>1 = SRAM controller reset. |
| [24] | **SDHRST** | **SDH Controller Reset**<br>Set this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = SDH controller normal operation.<br>1 = SDH controller reset. |
| [23:19] | **Reserved** | Reserved. |
| [18] | **USBHRST** | **USB Host Controller Reset**<br>Setting this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = USB Host controller normal operation.<br>1 = USB Host controller reset. |
| [17] | **I2SRST** | **I$^{2}$S Controller Reset**<br>Setting this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = I$^{2}$S controller normal operation.<br>1 = I$^{2}$S controller reset. |
| [16:14] | **Reserved** | Reserved. |
| [13] | **WDTPRST** | **WDT Controller Reset**<br>Setting this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = WDT controller normal operation.<br>1 = WDT controller reset. |
| [12] | **TMR3RST** | **Timer3 Controller Reset**<br>Setting this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = Timer3 controller normal operation.<br>1 = Timer3 controller reset. |
| [11] | **USBDRST** | **USB Device Controller Reset**<br>Setting this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = USB Device controller normal operation.<br>1 = USB Device controller reset. |
| [10] | **SPIMRST** | **SPIM Controller Reset**<br>Setting this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = SPIM controller normal operation.<br>1 = SPIM controller reset. |

| | | |
|---|---|---|
| [9] | **I2C1RST** | **I2C1 Controller Reset**<br>Setting this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = I2C1 controller normal operation.<br>1 = I2C1 controller reset. |
| [8] | **I2C0RST** | **I2C0 Controller Reset**<br>Setting this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = I2C0 controller normal operation.<br>1 = I2C0 controller reset. |
| [7] | **PWMRST** | **PWM Controller Reset**<br>Setting this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = PWM controller normal operation.<br>1 = PWM controller reset. |
| [6] | **UART2RST** | **UART2 Controller Reset**<br>Setting this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = UART2 controller normal operation.<br>1 = UART2 controller reset. |
| [5] | **TMR2RST** | **Timer2 Controller Reset**<br>Setting this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = Timer2 controller normal operation.<br>1 = Timer2 controller reset. |
| [4] | **WDTFRST** | **WDT Hardware Controller Reset**<br>Setting this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = WDT controller normal operation.<br>1 = WDT controller reset. |
| [3] | **TMR1RST** | **Timer1 Controller Reset**<br>Setting this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = Timer1 controller normal operation.<br>1 = Timer1 controller reset. |
| [2] | **TMR0RST** | **Timer0 Controller Reset**<br>Setting this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = Timer0 controller normal operation.<br>1 = Timer0 controller reset. |
| [1] | **UART1RST** | **UART1 Controller Reset**<br>Setting this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = UART1 controller normal operation.<br>1 = UART1 controller reset. |

| | | |
|---|---|---|
| [0] | **UART0RST** | **UART0 Controller Reset**<br>Setting this bit to 1 will generate a reset signal to the USB host controller. User needs to set this bit to 0 to release from the reset state.<br>0 = UART0 controller normal operation.<br>1 = UART0 controller reset. |

### NMI Control Register (SYS_NMICTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_NMICTL | GCR_BA+0x18 | R/W | Non Maskable Interrupt Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| LVDIF | EINT3IF | EINT2IF | EINT1IF | EINT0IF | PORIF | WDTIF | RTCIF |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| LVDIEN | EINT3IEN | EINT2IEN | EINT1IEN | EINT0IEN | PORIEN | WDTIEN | RTCIEN |

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23] | **LVDIF** | **Low Voltage Detect (LVD) Interrupt Flag (Read Only)**<br>0 = LVD interrupt is deasserted.<br>1 = LVD interrupt is asserted. |
| [22] | **EINT3IF** | **External GPIO Group 3 Interrupt Flag (Read Only)**<br>0 = External GPIO group 3 interrupt is deasserted.<br>1 = External GPIO group 3 interrupt is asserted. |
| [21] | **EINT2IF** | **External GPIO Group 2 Interrupt Flag (Read Only)**<br>0 = External GPIO group 2 interrupt is deasserted.<br>1 = External GPIO group 2 interrupt is asserted. |
| [20] | **EINT1IF** | **External GPIO Group 1 Interrupt Flag (Read Only)**<br>0 = External GPIO group 1 interrupt is deasserted.<br>1 = External GPIO group 1 interrupt is asserted. |
| [19] | **EINT0IF** | **External GPIO Group 0 Interrupt Flag (Read Only)**<br>0 = External GPIO group 0 interrupt is deasserted.<br>1 = External GPIO group 0 interrupt is asserted. |
| [18] | **PORIF** | **Power on Reset (POR) Interrupt Flag (Read Only)**<br>0 = POR interrupt is deasserted.<br>1 = POR interrupt is asserted. |
| [17] | **WDTIF** | **Watch Dog Timer (WDT) Interrupt Flag (Read Only)**<br>0 = WDT interrupt is deasserted.<br>1 = WDT interrupt is asserted. |

| | | |
|---|---|---|
| [16] | **RTCIF** | **RTC Interrupt Flag (Read Only)**<br>0 = RTC interrupt is deasserted.<br>1 = RTC interrupt is asserted. |
| [15:8] | **Reserved** | Reserved. |
| [7] | **LVDIEN** | **Low Voltage Detect NMI Source Enable Control**<br>0 = LVD NMI source Disabled.<br>1 = LVD NMI source Enabled. |
| [6] | **EINT3IEN** | **External GPIO Group 3 NMI Source Enable Control**<br>0 = External GPIO group 3 NMI source Disabled.<br>1 = External GPIO group 3 NMI source Enabled. |
| [5] | **EINT2IEN** | **External GPIO Group 2 NMI Source Enable Control**<br>0 = External GPIO group 2 NMI source Disabled.<br>1 = External GPIO group 2 NMI source Enabled. |
| [4] | **EINT1IEN** | **External GPIO Group 1 NMI Source Enable Control**<br>0 = External GPIO group 1 NMI source Disabled.<br>1 = External GPIO group 1 NMI source Enabled. |
| [3] | **EINT0IEN** | **External GPIO Group 0 NMI Source Enable Control**<br>0 = External GPIO group 0 NMI source Disabled.<br>1 = External GPIO group 0 NMI source Enabled. |
| [2] | **PORIEN** | **Power on Interrupt NMI Source Enable Control**<br>0 = POR NMI source Disabled.<br>1 = POR NMI source Enabled. |
| [1] | **WDTIEN** | **WDT Interrupt NMI Source Enable Control**<br>0 = WDT NMI source Disabled.<br>1 = WDT NMI source Enabled. |
| [0] | **RTCIEN** | **RTC Interrupt NMI Source Enable Control**<br>0 = RTC NMI source Disabled.<br>1 = RTC NMI source Enabled. |

### Reset Status Control Register (SYS_RSTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_RSTSTS | GCR_BA+0x1C | R/W | Reset Status Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:5] | **Reserved** | Reserved. |
| [4] | **PORF** | **Reset Status for  POR Reset**<br>0 = No effect.<br>1 =  Power-on Reset (POR) had issued the reset signal to reset the system.<br>**Note:** Write 1 to clear this bit to 0. |
| [3] | **CPURF** | **Reset Status for Software Setting**<br>0 = No effect.<br>1 = The CPURST had be triggered to reset the CPU or the CHIPREST had be triggered to reset the system.<br>**Note:** Write 1 to clear this bit to 0. |
| [2] | **WDTRF** | **Reset Status for Watch Dog Reset**<br>0 = No effect.<br>1 = The watchdog timer or window watchdog timer had issued the reset signal to reset the system.<br>**Note:** Write 1 to clear this bit to 0. |
| [1] | **LVRF** | **Reset Status for Low Voltage Reset**<br>0 = No effect.<br>1 = LVR controller had issued the reset signal to reset the system.<br>**Note:** Write 1 to clear this bit to 0. |
| [0] | **PINRF** | **Reset Status for External NReset Pin**<br>0 = No effect.<br>1 = nRESET pin had issued the reset signal to reset the system.<br>**Note:** Write 1 to clear this bit to 0. |

### AHB Bus Control Register (SYS_AHBCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_AHBCTL** | GCR_BA+0x20 | R/W | AHB Bus Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:6] | **Reserved** | Reserved. |
| [5] | **PRISTS** | **Interrupt Active Status in CPUHPRI Enabled Mode**<br>If it is high, the CPU has the highest AHB bus priority. It is set when the CPUHPRI is enabled and the external IRQ is active. This bit is cleared by writing 1 to it. Therefore, if exiting from the IRQ when CPUHPRI is enabled, PRISTS must be cleared. Otherwise, the CPU always has the highest AHB bus priority.<br>0 = No effect.<br>1 = The highest AHB bus priority for CPU is active. |
| [4] | **CPUHPRI** | **Enable Raising the Priority of CPU in IRQ Period**<br>It can be used to reduce the interrupt latency in a real-time system. Setting this bit, the CPU will have the highest AHB priority.<br>0 = No effect.<br>1 = The function that CPU has the highest AHB bus priority in IRQ period Enabled. |
| [3:1] | **Reserved** | Reserved. |
| [0] | **PRISEL** | **AHB Bus Arbitration Mode Control**<br>0 = Fixed priority mode.<br>1 = Round-robin priority mode (rotate).<br>The priority mode for fixed priority mode is $I^2S$ > SDH > USBH > USBD > SPIM > M4(S) > M4(D) > M4(I) |

### GPIOA Low Byte Multi-function Control Register (SYS_GPA_MFPL)

Please refer to 4.3.9 GPIO Multi-function Pin Summary

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_GPA_MFPL** | GCR_BA+0x30 | R/W | GPIOA Low Byte Multi-function Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [30:28] | **PA7MFP** | **PA.7 Multi-function Pin Selection** |
| [26:24] | **PA6MFP** | **PA.6 Multi-function Pin Selection** |
| [22:20] | **PA5MFP** | **PA.5 Multi-function Pin Selection** |
| [18:16] | **PA4MFP** | **PA.4 Multi-function Pin Selection** |
| [14:12] | **PA3MFP** | **PA.3 Multi-function Pin Selection** |
| [10:8] | **PA2MFP** | **PA.2 Multi-function Pin Selection** |
| [6:4] | **PA1MFP** | **PA.1 Multi-function Pin Selection** |
| [2:0] | **PA0MFP** | **PA.0 Multi-function Pin Selection** |

### GPIOA High Byte Multi-function Control Register (SYS_GPA_MFPH)

Please refer to 4.3.9 GPIO Multi-function Pin Summary

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_GPA_MFPH** | GCR_BA+0x34 | R/W | GPIOA High Byte Multi-function Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [30:28] | **PA15MFP** | **PA.15 Multi-function Pin Selection** |
| [26:24] | **PA14MFP** | **PA.14 Multi-function Pin Selection** |
| [22:20] | **PA13MFP** | **PA.13 Multi-function Pin Selection** |
| [18:16] | **PA12MFP** | **PA.12 Multi-function Pin Selection** |
| [14:12] | **PA11MFP** | **PA.11 Multi-function Pin Selection** |
| [10:8] | **PA10MFP** | **PA.10 Multi-function Pin Selection** |
| [6:4] | **PA9MFP** | **PA.9 Multi-function Pin Selection** |
| [2:0] | **PA8MFP** | **PA.8 Multi-function Pin Selection** |

### GPIOB Low Byte Multi-function Control Register (SYS_GPB_MFPL)

Please refer to 4.3.9 GPIO Multi-function Pin Summary

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_GPB_MFPL** | GCR_BA+0x38 | R/W | GPIOB Low Byte Multi-function Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [30:28] | **PB7MFP** | **PB.7 Multi-function Pin Selection** |
| [26:24] | **PB6MFP** | **PB.6 Multi-function Pin Selection** |
| [22:20] | **PB5MFP** | **PB.5 Multi-function Pin Selection** |
| [18:16] | **PB4MFP** | **PB.4 Multi-function Pin Selection** |
| [14:12] | **PB3MFP** | **PB.3 Multi-function Pin Selection** |
| [10:8] | **PB2MFP** | **PB.2 Multi-function Pin Selection** |
| [6:4] | **PB1MFP** | **PB.1 Multi-function Pin Selection** |
| [2:0] | **PB0MFP** | **PB.0 Multi-function Pin Selection** |

### GPIOB High Byte Multi-function Control Register (SYS_GPB_MFPH)

Please refer to 4.3.9 GPIO Multi-function Pin Summary

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_GPB_MFPH** | GCR_BA+0x3C | R/W | GPIOB High Byte Multi-function Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [30:28] | **PB15MFP** | **PB.15 Multi-function Pin Selection** |
| [26:24] | **PB14MFP** | **PB.14 Multi-function Pin Selection** |
| [22:20] | **PB13MFP** | **PB.13 Multi-function Pin Selection** |
| [18:16] | **PB12MFP** | **PB.12 Multi-function Pin Selection** |
| [14:12] | **PB11MFP** | **PB.11 Multi-function Pin Selection** |
| [10:8] | **PB10MFP** | **PB.10 Multi-function Pin Selection** |
| [6:4] | **PB9MFP** | **PB.9 Multi-function Pin Selection** |
| [2:0] | **PB8MFP** | **PB.8 Multi-function Pin Selection** |

### GPIOC Low Byte Multi-function Control Register (SYS_GPC_MFPL)

Please refer to 4.3.9 GPIO Multi-function Pin Summary

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPC_MFPL | GCR_BA+0x40 | R/W | GPIOC Low Byte Multi-function Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [30:28] | PC7MFP | PC.7 Multi-function Pin Selection |
| [26:24] | PC6MFP | PC.6 Multi-function Pin Selection |
| [22:20] | PC5MFP | PC.5 Multi-function Pin Selection |
| [18:16] | PC4MFP | PC.4 Multi-function Pin Selection |
| [14:12] | PC3MFP | PC.3 Multi-function Pin Selection |
| [10:8] | PC2MFP | PC.2 Multi-function Pin Selection |
| [6:4] | PC1MFP | PC.1 Multi-function Pin Selection |
| [2:0] | PC0MFP | PC.0 Multi-function Pin Selection |

### GPIOC High Byte Multi-function Control Register (SYS_GPC_MFPH)

Please refer to 4.3.9 GPIO Multi-function Pin Summary

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_GPC_MFPH** | GCR_BA+0x44 | R/W | GPIOC High Byte Multi-function Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [26:24] | **PC14MFP** | **PC.14 Multi-function Pin Selection** |
| [22:20] | **PC13MFP** | **PC.13 Multi-function Pin Selection** |
| [18:16] | **PC12MFP** | **PC.12 Multi-function Pin Selection** |
| [14:12] | **PC11MFP** | **PC.11 Multi-function Pin Selection** |
| [10:8] | **PC10MFP** | **PC.10 Multi-function Pin Selection** |
| [6:4] | **PC9MFP** | **PC.9 Multi-function Pin Selection** |
| [2:0] | **PC8MFP** | **PC.8 Multi-function Pin Selection** |

### GPIOD Low Byte Multi-function Control Register (SYS_GPD_MFPL)

Please refer to 4.3.9 GPIO Multi-function Pin Summary

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_GPD_MFPL** | GCR_BA+0x48 | R/W | GPIOD Low Byte Multi-function Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:7] | **Reserved** | Reserved. |
| [18:16] | **PD4MFP** | **PD.4 Multi-function Pin Selection** |
| [14:12] | **PD3MFP** | **PD.3 Multi-function Pin Selection** |
| [10:8] | **PD2MFP** | **PD.2 Multi-function Pin Selection** |
| [6:4] | **PD1MFP** | **PD.1 Multi-function Pin Selection** |
| [2:0] | **PD0MFP** | **PD.0 Multi-function Pin Selection** |

### Load VECMAP Address Parameter Control Register (SYS_LVMPADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_LVMPADDR** | GCR_BA+0x50 | R/W | Load VECMAP Address Parameter Control Register | 0x1FFF_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ADDR** | **Load VECMAP Address Register**<br>This is the start address for mapping to the address 0x0000_0000 in VECMAP function. Only when CPU reset or setting RLDVMP to 1, the loading signal will be loaded to the SYS_RVMPADDR register.<br>**Note:** This register can only be reset by CHIPRST and HW Reset. |

### Load VECMAP Length Parameter Control Register (SYS_LVMPLEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_LVMPLEN** | GCR_BA+0x54 | R/W | Load VECMAP Length Parameter Control Register | 0x0000_0008 |

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

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **LEN** | **LD_VECMAP Length**<br>This is the memory length loading signal for mapping to the address 0x0000_0000 in VECMAP function. Only when OCPU reset or setting RLDVMP to 1, the loading signal will be loaded to the SYS_RVMPLEN register.<br>Mapping memory length = VMP_LEN * 1K (bytes).<br>**Note1:** The maximum mapping length is 128K bytes<br>**Note2:** This register only can be reset by CHIPRST and HW Reset. |

### Real VECMAP Address Parameter Register (SYS_RVMPADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_RVMPADDR** | GCR_BA+0x58 | R | Real VECMAP Address Parameter Register | 0x1FFF_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ADDR** | **Real VECMAP Address Register Parameter**<br>This is the real start address parameter for mapping to the address 0x0000_0000 in VECMAP function. (Default value is mapping to IBR_ROM start address.)<br>**Note:** This register is only loaded from ADDR during CPU Reset process (SYS_IPRST0[0]) or setting RLDVMP. |

### Real VECMAP Length Parameter Control Register (SYS_RVMPLEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_RVMPLEN** | GCR_BA+0x5C | R/W | Real VECMAP Length Parameter Control Register | 0x0800_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **LEN** | **Real VECMAP Length**<br>This is the real memory length for mapping to the address 0x0000_0000 in VECMAP function.<br>Mapping memory length = VMP_LEN * 1K (bytes).<br>**Note1:** Read Only<br>**Note2:** These bits are only loaded from LEN when setting CPURST (SYS_IPRST0[0]) or setting RLDVMP. |
| [23:1] | **Reserved** | Reserved. |
| [0] | **RLDVMP** | **Load VECMAP Parameter Signal**<br>0 = No effect.<br>1 = Load VECMAP Address and Length.<br>**Note:** This bit is auto cleared to 0 |

### Embedded SPI Flash Pad Pull-up Enable Control Register (SYS_EPADPUEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_EPADPUEN** | GCR_BA+0x6C | R/W | Embedded SPI Flash Pad Pull-up Enable Control Register | 0x0000_001F |

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

| Bits | Description | |
|---|---|---|
| [31:5] | **Reserved** | Reserved. |
| [4:0] | **EPADPUEN** | **Embedded SPI Flash Pad Pull-up Enable Control**<br>00000 = All embedded pads are pull-up Disabled.<br>10010 = Only SPI Flash MISO pads are pull-up Enabled.<br>11111 = All embedded pads are pull-up Enabled.<br>Other = Reserved.<br>**Note:** In Power-down mode, user should set EPADPUEN[4:0] to 0x12. |

### GPIOA Driving Strength Control Register (SYS_GPADS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SYS_GPADS | GCR_BA+0x70 | R/W | GPIOA Driving Strength Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | PA7DS | | | Reserved | PA6DS | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| Reserved | PA5DS | | | Reserved | PA4DS | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | PA3DS | | | Reserved | PA2DS | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Reserved | PA1DS | | | Reserved | PA0DS | | |

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30:28] | **PA7DS** | **PA.7 Driving Strength Control**<br>Setting driving strength for PA.7 analog / digital combo pin.<br>000 = 2.0 mA (Default).<br>001 = 6.5 mA.<br>010 = 8.7 mA.<br>011 = 13.0 mA.<br>100 = 15.2 mA.<br>101 = 19.5 mA.<br>110 = 21.7 mA.<br>111 = 26.1 mA. |
| [27] | **Reserved** | Reserved. |
| [26:24] | **PA6DS** | **PA.6 Driving Strength Control**<br>Setting driving strength for PA.6 analog / digital combo pin.<br>000 = 2.0 mA (Default).<br>001 = 6.5 mA.<br>010 = 8.7 mA.<br>011 = 13.0 mA.<br>100 = 15.2 mA.<br>101 = 19.5 mA.<br>110 = 21.7 mA.<br>111 = 26.1 mA. |
| [23] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [22:20] | **PA5DS** | **PA.5 Driving Strength Control**<br>Setting driving strength for PA.5 analog / digital combo pin.<br>000 = 2.0 mA (Default).<br>001 = 6.5 mA.<br>010 = 8.7 mA.<br>011 = 13.0 mA.<br>100 = 15.2 mA.<br>101 = 19.5 mA.<br>110 = 21.7 mA.<br>111 = 26.1 mA. |
| [19] | **Reserved** | Reserved. |
| [18:16] | **PA4DS** | **PA.4 Driving Strength Control**<br>Setting driving strength for PA.4 analog / digital combo pin.<br>000 = 2.0 mA (Default).<br>001 = 6.5 mA.<br>010 = 8.7 mA.<br>011 = 13.0 mA.<br>100 = 15.2 mA.<br>101 = 19.5 mA.<br>110 = 21.7 mA.<br>111 = 26.1 mA. |
| [15] | **Reserved** | Reserved. |
| [14:12] | **PA3DS** | **PA.3 Driving Strength Control**<br>Setting driving strength for PA.3 analog / digital combo pin.<br>000 = 2.0 mA (Default).<br>001 = 6.5 mA.<br>010 = 8.7 mA.<br>011 = 13.0 mA.<br>100 = 15.2 mA.<br>101 = 19.5 mA.<br>110 = 21.7 mA.<br>111 = 26.1 mA. |
| [11] | **Reserved** | Reserved. |
| [10:8] | **PA2DS** | **PA.2 Driving Strength Control**<br>Setting driving strength for PA.2 analog / digital combo pin.<br>000 = 2.0 mA (Default).<br>001 = 6.5 mA.<br>010 = 8.7 mA.<br>011 = 13.0 mA.<br>100 = 15.2 mA.<br>101 = 19.5 mA.<br>110 = 21.7 mA.<br>111 = 26.1 mA. |
| [7] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [6:4] | **PA1DS** | **PA.1 Driving Strength Control**<br>Setting driving strength for PA.1 analog / digital combo pin.<br>000 = 2.0 mA (Default).<br>001 = 6.5 mA.<br>010 = 8.7 mA.<br>011 = 13.0 mA.<br>100 = 15.2 mA.<br>101 = 19.5 mA.<br>110 = 21.7 mA.<br>111 = 26.1 mA. |
| [3] | **Reserved** | Reserved. |
| [2:0] | **PA0DS** | **PA.0 Driving Strength Control**<br>Setting driving strength for PA.0 analog / digital combo pin.<br>000 = 2.0 mA (Default).<br>001 = 6.5 mA.<br>010 = 8.7 mA.<br>011 = 13.0 mA.<br>100 = 15.2 mA.<br>101 = 19.5 mA.<br>110 = 21.7 mA.<br>111 = 26.1 mA. |

## GPIOA Input Buffer Enable Control Register (SYS_GPAIBE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_GPAIBE** | GCR_BA+0x74 | R/W | GPIOA Input Buffer Enable Control Register | 0xFFFF_0000 |

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

| Bits | Description | |
|---|---|---|
| [30:24] | **SMTENx** | **SMTENx (x = 8,9,…,15) Schmitt Trigger Input Buffer Enable Control.**<br>0 = PA.x schmitt Trigger Input Buffer Disabled.<br>1 = PA.x schmitt Trigger Input Buffer Enabled (Default).<br>**Note:** If both Schmitt Trigger and CMOS input buffer are set to 0, the input signal from PAD will always be zero. |
| [23:16] | **DINONx** | **DINONx (x = 0,1,…,7) Input Buffer Enable Control.**<br>0 = PA.x Input Buffer Disabled.<br>1 = PA.x Input Buffer Enabled (Default).<br>**Note1:** If setting to 0, the input signal from PAD will always be zero.<br>**Note2:** If using PA.0~PA.7 as analog pads, remember to disable input buffer. |
| [15:8] | **CMOSENx** | **CMOSENx (x = 8,9,…,15)CMOS Input Buffer Enable Control.**<br>0 = PA.x CMOS Input Buffer Disabled. (Default)<br>1 = PA.x CMOS Input Buffer Enabled.<br>**Note:** If both Schmitt Trigger and CMOS input buffer are set to 0, the input signal from PAD will always be zero. |
| [7:0] | **IBSELx** | **IBSELx (x = 0,1,…,7) Input Buffer Select from Schmitt Trigger or CMOS.**<br>0 = PA.x CMOS Input Buffer (Default).<br>1 = PA.x Schmitt Trigger Input Buffer. |

**Note:** Schmitt Trigger and CMOS input buffer cannot be enabled at the same time.

### GPIOB Input Buffer Enable Control Register (SYS_GPBIBE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_GPBIBE** | GCR_BA+0x78 | R/W | GPIOB Input Buffer Enable Control Register | 0xFFFF_0000 |

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

| Bits | Description | |
|---|---|---|
| [30:16] | **SMTENx** | **SMTENx (x = 0,1,…,15) Schmitt Trigger Input Buffer Enable Control.**<br>0 = PB.x Schmitt Trigger Input Buffer Disabled.<br>1 = PB.x Schmitt Trigger Input Buffer Enabled (Default).<br>**Note:** If both Schmitt Trigger and CMOS input buffer are set to 0, the input signal from PAD will always be zero. |
| [15:0] | **CMOSENx** | **CMOSENx (x = 0,1,…,15)CMOS Input Buffer Enable Control.**<br>0 = PB.x CMOS Input Buffer Disabled. (Default)<br>1 =PB.x CMOS Input Buffer Enabled.<br>**Note:** If both Schmitt Trigger and CMOS input buffer are set to 0, the input signal from PAD will always be zero. |

**Note:** Schmitt Trigger and CMOS input buffer cannot be enabled at the same time.

NUC505 SERIES TECHNICAL REFERENCE MANUAL

### GPIOC Input Buffer Enable Control Register (SYS_GPCIBE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_GPCIBE** | GCR_BA+0x7C | R/W | GPIOC Input Buffer Enable Control Register | 0xFFFF_0000 |

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

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30:16] | **SMTENx** | **SMTENx (x = 0,1,…,14) Schmitt Trigger Input Buffer Enable Control.**<br>0 = PC.x Schmitt Trigger Input Buffer Disabled.<br>1 = PC.x Schmitt Trigger Input Buffer Enabled (Default).<br>**Note:** If both Schmitt Trigger and CMOS input buffer are set to 0, the input signal from PAD will always be zero. |
| [15] | **Reserved** | Reserved. |
| [14:0] | **CMOSENx** | **CMOSENx (x = 0,1,…,14)CMOS Input Buffer Enable Control.**<br>0 = PC.x CMOS Input Buffer Disabled. (Default)<br>1 =PC.x CMOS Input Buffer Enabled.<br>**Note:** If both Schmitt Trigger and CMOS input buffer are set to 0, the input signal from PAD will always be zero. |

**Note:** Schmitt Trigger and CMOS input buffer cannot be enabled at the same time.

### GPIOD Input Buffer Enable Control Register (SYS_GPDIBE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_GPDIBE** | GCR_BA+0x80 | R/W | GPIOD Input Buffer Enable Control Register | 0xFFFF_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:21] | **Reserved** | Reserved. |
| [20:18] | **DINONx** | **DINONx (x = 2,3,4) Input Buffer Enable Control.**<br>0 = PD.x Input Buffer Disabled.<br>1 = PD.x Input Buffer Enabled (Default).<br>**Note1:** If setting to 0, the input signal from PAD will always be zero.<br>**Note2:** If using PD.2, PD.3, and PD.4 as analog pads, user must disable PD.2, PD.3, and PD.4 input buffer. |
| [17:16] | **SMTENx** | **SMTENx (x = 0,1) Schmitt Trigger Input Buffer Enable Control.**<br>0 = PD.x Schmitt Trigger Input Buffer Disabled.<br>1 = PD.x Schmitt Trigger Input Buffer Enabled (Default).<br>**Note:** If both Schmitt Trigger and CMOS input buffer are set to 0, the input signal from PAD will always be zero. |
| [15:5] | **Reserved** | Reserved. |
| [4:2] | **IBSELx** | **IBSELx (x = 2,3,4) Input Buffer Select from Schmitt Trigger or CMOS.**<br>0 = PD.x CMOS Input Buffer (Default).<br>1 = PD.x Schmitt Trigger Input Buffer. |
| [1:0] | **CMOSENx** | **CMOSENx (x = 0,1)CMOS Input Buffer Enable Control.**<br>0 = PD.x CMOS Input Buffer Disabled (Default).<br>1 = PD.x CMOS Input Buffer Enabled.<br>**Note:** If both Schmitt Trigger and CMOS input buffer are set to 0, the input signal from PAD will always be zero. |

**Note:** Schmitt Trigger and CMOS input buffer cannot be enabled at the same time.

### GPIOD Driving Strength Control Register (SYS_GPDDS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_GPDDS** | GCR_BA+0x84 | R/W | GPIOD Driving Strength Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | PD4DS | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | PD3DS | | | Reserved | PD2DS | | |

| Bits | Description | |
|---|---|---|
| [31:11] | **Reserved** | Reserved. |
| [10:8] | **PD4DS** | **PD.4 Driving Strength Control**<br>Setting driving strength for PD.4 analog / digital combo pin.<br>000 = 2.0 mA (Default).<br>001 = 6.5 mA.<br>010 = 8.7 mA.<br>011 = 13.0 mA.<br>100 = 15.2 mA.<br>101 = 19.5 mA.<br>110 = 21.7 mA.<br>111 = 26.1 mA. |
| [7] | **Reserved** | Reserved. |
| [6:4] | **PD3DS** | **PD.3 Driving Strength Control**<br>Setting driving strength for PD.3 analog / digital combo pin.<br>000 = 2.0 mA (Default).<br>001 = 6.5 mA.<br>010 = 8.7 mA.<br>011 = 13.0 mA.<br>100 = 15.2 mA.<br>101 = 19.5 mA.<br>110 = 21.7 mA.<br>111 = 26.1 mA. |
| [3] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [2:0] | **PD2DS** | **PD.2 Driving Strength Control**<br>Setting driving strength for PD.2 analog / digital combo pin.<br>000 =  2.0 mA (Default).<br>001 =  6.5 mA.<br>010 =  8.7 mA.<br>011 = 13.0 mA.<br>100 = 15.2 mA.<br>101 = 19.5 mA.<br>110 = 21.7 mA.<br>111 = 26.1 mA. |

## External nRESET Pin De-bounce Counter Control Register (SYS_RSTDBCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_RSTDBCNT** | GCR_BA+0x100 | R/W | External nRESET Pin De-bounce Counter Control Register | 0x0000_04B0 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **RSTDBCNT** | **External NRESET De-bounce Counter**<br>This 16-bit external nRESET pin de-bounce counter can specify the external nRESET pin de-bounce time up to around 5.46ms (0xFFFF) @XIN=12 MHz.<br>**Note:** The default external nRESET pin de-bounce time is 0.1ms (0x04B0) at XIN = 12 MHz. |

### External nRESET Pin De-bounce Control Register (SYS_RSTDBEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYS_RSTDBEN** | GCR_BA+0x104 | R/W | External nRESET Pin De-bounce Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **RSTDBEN** | **External NRESET De-bounce Control**<br>This bit is to enable or disable the external nRESET pin de-bounce process.<br>0 = De-bounce function Disabled.<br>1 = De-bounce function Enabled. |

### 6.2.10 System Timer (Systick)

The Cortex®-M4 includes an integrated system timer, SysTick, which provides a simple, 24-bit clear-on-write, decrementing, wrap-on-zero counter with a flexible control mechanism. The counter can be used as a Real Time Operating System (RTOS) tick timer or as a simple counter.

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

The SYST_CVR value is UNKNOWN on reset. Software should write to the register to clear it to zero before enabling the feature. This ensures the timer will count from the SYST_RVR value rather than an arbitrary value when it is enabled.

If the SYST_RVR is zero, the timer will be maintained with a current value of zero after it is reloaded with this value. This mechanism can be used to disable the feature independently from the timer enable bit.

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

### 6.2.10.1 System Timer Control Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SCS Base Address:**<br>**SCS_BA = 0xE000_E000** | | | | |
| **SYST_CSR** | SCS_BA+0x10 | R/W | SysTick Control and Status Register | 0x0000_0000 |
| **SYST_RVR** | SCS_BA+0x14 | R/W | SysTick Reload Value Register | 0xXXXX_XXXX |
| **SYST_CVR** | SCS_BA+0x18 | R/W | SysTick Current Value Register | 0xXXXX_XXXX |

### *6.2.10.2 System Timer Control Register Description*

**SysTick Control and Status (SYST_CSR)**

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYST_CSR** | SCS_BA+0x10 | R/W | SysTick Control and Status Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:17] | **Reserved** | Reserved. |
| [16] | **COUNTFLAG** | **Returns 1 If Timer Counted to 0 Since Last Time this Register Was Read**<br>COUNTFLAG is set by a count transition from 1 to 0.<br>COUNTFLAG is cleared on read or by a write to the Current Value register. |
| [15:3] | **Reserved** | Reserved. |
| [2] | **CLKSRC** | **SysTick Counting Clock Source Select**<br>0 = Clock source is (optional) external reference clock.<br>1 = Core clock used for SysTick. |
| [1] | **TICKINT** | **SysTick Interrupt Enable Control**<br>0 = Counting down to 0 does not cause the SysTick exception to be pended. Software can use COUNTFLAG to determine if a count to zero has occurred.<br>1 = Counting down to 0 will cause the SysTick exception to be pended. Clearing the SysTickcurrent value register by a register write in software will not cause SysTick to be pended. |
| [0] | **ENABLE** | **SysTick Function Enable Control**<br>0 = Counter Disabled.<br>1 = Counter will operate in a multi-shot manner. |

## SysTickReload Value Register (SYST_RVR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYST_RVR** | SCS_BA+0x14 | R/W | SysTick Reload Value Register | 0xXXXX_XXXX |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **RELOAD** | **SysTick Reload Value**<br>The value to load into the Current Value register when the counter reaches 0. |

## SysTickCurrent Value Register (SYST_CVR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SYST_CVR** | SCS_BA+0x18 | R/W | SysTick Current Value Register | 0xXXXX_XXXX |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **CURRENT** | **Current Counter Value**<br>This is the value of the counter at the time it is sampled.The counter does not provide read-modify-write protection.The register is write-clear.A software write of any value will clear the register to 0.Unsupported bits RAZ (See SysTick reload value register). |

### 6.2.11 Nested Vectored Interrupt Control (NVIC)

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

- An implementation-defined number of interrupts, in the range 1-240 interrupts.
- A programmable priority level of 0-16 for each interrupts. A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.
- Level and pulse detection of interrupt signals.
- Dynamic reprioritization of interrupts.
- Grouping of priority values into group priority and subpriority fields.
- Interrupt tail-chaining.
- An external Non Maskable Interrupt (NMI)
- WIC, providing Power-down mode support.

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

#### *6.2.11.1Exception Model and System Interrupt Map*

The following table lists the exception model supported by NUC505 series. Software can set 16 levels of priority on some of these exceptions as well as on all interrupts. The highest user-configurable priority is denoted as “0x00” and the lowest priority is denoted as “0xF0” (The 4-LSB always 0). The default priority of all the user-configurable interrupts is “0x00”. Note that priority “0” is treated as the fourth priority on the system, after three system exceptions “Reset”, “NMI” and “Hard Fault”.

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

The vector table contains the initialization value for the stack pointer on reset, and the entry point addresses for all exception handlers. The vector number on previous page defines the order of entries in the vector table associated with exception handler entry as illustrated in previous section.

| Exception Type | Vector Number | Vector Address | Priority |
|---|---|---|---|
| Reset | 1 | 0x00000004 | -3 |
| NMI | 2 | 0x00000008 | -2 |
| Hard Fault | 3 | 0x0000000C | -1 |
| Memory Manager Fault | 4 | 0x00000010 | Configurable |
| Bus Fault | 5 | 0x00000014 | Configurable |
| Usage Fault | 6 | 0x00000018 | Configurable |
| Reserved | 7 ~ 10 | | Reserved |
| SVCall | 11 | 0x0000002C | Configurable |
| Debug Monitor | 12 | 0x00000030 | Configurable |
| Reserved | 13 | | Reserved |
| PendSV | 14 | 0x00000038 | Configurable |
| SysTick | 15 | 0x0000003C | Configurable |
| Interrupt (IRQ0 ~ IRQ31) | 16 ~ 47 | 0x00000000 +<br>(Vector Number)*4 | Configurable |

Table 6.2-4 Exception Model

| Vector Number | Interrupt Number | Interrupt Name | Interrupt Description |
|---|---|---|---|
| | (Bit in Interrupt Registers) | | |
| 0 ~ 15 | - | - | System exceptions |
| 16 | 0 | PWR_INT | Power On Interrupt |
| 17 | 1 | WDT_INT | Watch Dog Timer interrupt |
| 18 | 2 | Reserved | Reserved |
| 19 | 3 | I2S_INT | $I^2S$ interrupt |
| 20 | 4 | EINT0_INT | External GPIO Group 0 interrupt |
| 21 | 5 | EINT1_INT | External GPIO Group 1 interrupt |
| 22 | 6 | EINT2_INT | External GPIO Group 2 interrupt |
| 23 | 7 | EINT3_INT | External GPIO Group 3 interrupt |
| 24 | 8 | SPIM_INT | SPIM interrupt |
| 25 | 9 | USBD_INT | USB Device 20 interrupt |
| 26 | 10 | TM0_INT | Timer0 interrupt |
| 27 | 11 | TM1_INT | Timer1 interrupt |
| 28 | 12 | TM2_INT | Timer2 interrupt |

| | | | |
|---|---|---|---|
| 29 | 13 | TM3_INT | Timer3 interrupt |
| 30 | 14 | SDH_INT | SDH interrupt |
| 31 | 15 | PWM0_INT | PWM0 interrupt |
| 32 | 16 | PWM1_INT | PWM1 interrupt |
| 33 | 17 | PWM2_INT | PWM2 interrupt |
| 34 | 18 | PWM3_INT | PWM3 interrupt |
| 35 | 19 | RTC_INT | Real Time Clock interrupt |
| 36 | 20 | SPI0_INT | SPI0 interrupt |
| 37 | 21 | I2C1_INT | I2C1 interrupt |
| 38 | 22 | I2C0_INT | I2C0 interrupt |
| 39 | 23 | UART0_INT | UART0 interrupt |
| 40 | 24 | UART1_INT | UART1 interrupt |
| 41 | 25 | ADC_INT | ADC interrupt |
| 42 | 26 | wwdt_INT | Window Watch Dog Timer interrupt |
| 43 | 27 | USBH_INT | USB Host 1.1 interrupt |
| 44 | 28 | UART2_INT | UART2 interrupt |
| 45 | 29 | LVD_INT | Low Voltage Detection interrupt |
| 46 | 30 | SPI1_INT | SPI1 interrupt |
| 47 | 31 | Reserved | Reserved |

Table 6.2-5 Interrupt Number Table

#### *6.2.11.2Operation Description*

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

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

NVIC interrupts are prioritized by updating an 8-bit field within a 32-bit register (each register supporting four interrupts, and each interrupt uses MSB 4 bits of the 8-bit field).

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

### 6.2.11.3NVIC Control Registers

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SCS Base Address:**<br>**SCS_BA = 0xE000_E000**<br>**NVIC_IPRn_BA = 0xE000_E400+(0x04*n)**<br>**n = 0,1..7** | | | | |
| **NVIC_ISER0** | SCS_BA+0x100 | R/W | IRQ0 ~ IRQ31 Set-Enable Control Register | 0x0000_0000 |
| **NVIC_ICER0** | SCS_BA+0x180 | R/W | IRQ0 ~ IRQ31 Clear-Enable Control Register | 0x0000_0000 |
| **NVIC_ISPR0** | SCS_BA+0x200 | R/W | IRQ0 ~ IRQ31 Set-Pending Control Register | 0x0000_0000 |
| **NVIC_ICPR0** | SCS_BA+0x280 | R/W | IRQ0 ~ IRQ31 Clear-Pending Control Register | 0x0000_0000 |
| **NVIC_IABR0** | SCS_BA+0x300 | R/W | IRQ0 ~ IRQ31 Active Bit Register | 0x0000_0000 |
| **NVIC_IPRN** | NVIC_IPRn_BA+0x00 | R/W | IRQ0 ~ IRQ31 Interrupt Priority Control Register | 0x0000_0000 |
| **NVIC_STIR** | SCS_BA+0xF00 | W | Software Trigger Interrupt Registers | 0x0000_0000 |

## IRQ0 ~ IRQ31 Set-Enable Control Register (NVIC_ISERn)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **NVIC_ISER0** | SCS_BA+0x100 | R/W | IRQ0 ~ IRQ31 Set-Enable Control Register | 0x0000_0000 |

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

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

### IRQ0 ~ IRQ31Clear-Enable Control Register (NVIC_ICERn)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **NVIC_ICER0** | SCS_BA+0x180 | R/W | IRQ0 ~ IRQ31 Clear-Enable Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **CLRENA** | **Interrupt Clear Enable Control**<br>The NVIC_ICER0 registers disable interrupts, and show which interrupts are enabled.<br>Write Operation:<br>0 = No effect.<br>1 = Interrupt Disabled.<br>Read Operation:<br>0 = Interrupt Disabled.<br>1 = Interrupt Enabled. |

### IRQ0 ~ IRQ31 Set-Pending Control Register (NVIC_ISPRn)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **NVIC_ISPR0** | SCS_BA+0x200 | R/W | IRQ0 ~ IRQ31 Set-Pending Control Register | 0x0000_0000 |

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

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

### IRQ0 ~ IRQ31Clear-Pending Control Register (NVIC_ICPRn)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **NVIC_ICPR0** | SCS_BA+0x280 | R/W | IRQ0 ~ IRQ31 Clear-Pending Control Register | 0x0000_0000 |

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

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

### IRQ0 ~ IRQ31 Active Bit Register (NVIC_IABRn)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **NVIC_IABR0** | SCS_BA+0x300 | R/W | IRQ0 ~ IRQ31 Active Bit Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ACTIVE** | **Interrupt Active Flags**<br>The NVIC_IABR0 registers indicate which interrupts are active.<br>0 = Interrupt is not active.<br>1 = Interrupt is active. |

### IRQ0 ~ IRQ31 InterruptPriorityRegister (NVIC_IPRn)

Each implementation-defined priority field can hold a priority value, 0-15. The lower the value, the greater the priority of the corresponding interrupts. Register priority value fields are eight bits wide, and on-implemented low-order bits read as zero and ignore writes.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **NVIC_IPRN** | NVIC_IPRn_BA+0x00 | R/W | IRQ0 ~ IRQ31 Interrupt Priority Control Register | 0x0000_0000 |

**Note: n = 0,1..7**

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **PRI_4n+3** | | | | **Reserved** | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **PRI_4n+2** | | | | **Reserved** | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **PRI_4n+1** | | | | **Reserved** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **PRI_4n+0** | | | | **Reserved** | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **PRI_4n+3** | **Priority of IRQ_4n+3**<br>“0” denotes the highest priority and “15” denotes the lowest priority |
| [27:24] | **Reserved** | Reserved. |
| [23:20] | **PRI_4n+2** | **Priority of IRQ_4n+2**<br>“0” denotes the highest priority and “15” denotes the lowest priority |
| [19:16] | **Reserved** | Reserved. |
| [15:12] | **PRI_4n+1** | **Priority of IRQ_4n+1**<br>“0” denotes the highest priority and “15” denotes the lowest priority |
| [11:8] | **Reserved** | Reserved. |
| [7:4] | **PRI_4n+0** | **Priority of IRQ_4n+0**<br>“0” denotes the highest priority and “15” denotes the lowest priority |
| [3:0] | **Reserved** | Reserved. |

## Software Trigger InterruptRegister (NVIC_STIR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **NVIC_STIR** | SCS_BA+0xF00 | W | Software Trigger Interrupt Registers | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:9] | **Reserved** | Reserved. |
| [8:0] | **INTID** | **Write to the STIR to Generate an Interrupt From Software**<br>When the USERSETMPEND bit in the SCR is set to 1, unprivileged software can access the STIR<br>Interrupt ID of the interrupt to trigger, in the range 0-47. For example, a value of 0x03 specifies interrupt IRQ3. |

### 6.2.12 System Control Register Map and Description

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

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

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

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

### Interrupt Control and State Register (ICSR)

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

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

| Bits | Description | |
|---|---|---|
| [31] | NMIPENDSET | **NMI Set-pending Bit**<br>Write Operation:<br>0 = No effect.<br>1 = Changes NMI exception state to pending.<br>Read Operation:<br>0 = NMI exception is not pending.<br>1 = NMI exception is pending.<br>**Note:** Because NMI is the highest-priority exception, normally the processor enters the NMI exception handler as soon as it detects a write of 1 to this bit. Entering the handler then clears this bit to 0. This means a read of this bit by the NMI exception handler returns 1 only if the NMI signal is reasserted while the processor is executing that handler. |
| [30:29] | Reserved | Reserved. |
| [28] | PENDSVSET | **PendSV Set-pending Bit**<br>Write Operation:<br>0 = No effect.<br>1 = Changes PendSV exception state to pending.<br>Read Operation:<br>0 = PendSV exception is not pending.<br>1 = PendSV exception is pending.<br>**Note:** Writing 1 to this bit is the only way to set the PendSV exception state to pending. |
| [27] | PENDSVCLR | **PendSV Clear-pending Bit**<br>Write Operation:<br>0 = No effect.<br>1 = Removes the pending state from the PendSV exception.<br>**Note:** This is a write only bit. To clear the PENDSV bit, you must “write 0 to PENDSVSET and write 1 to PENDSVRTC_CAL” at the same time. |

| | | |
|---|---|---|
| [26] | **PENDSTSET** | **SysTick Exception Set-pending Bit**<br>Write Operation:<br>0 = No effect.<br>1 = Change SysTick exception state to pending.<br>Read Operation:<br>0 = SysTick exception is not pending.<br>1 = SysTick exception is pending. |
| [25] | **PENDSTCLR** | **SysTick Exception Clear-pending Bit**<br>Write Operation:<br>0 = No effect.<br>1 = Removes the pending state from the SysTick exception.<br>**Note:** This is a write only bit. To clear the PENDST bit, you must “write 0 to PENDSTSET and write 1 to PENDSTRTC_CAL” at the same time. |
| [24] | **Reserved** | Reserved. |
| [23] | **ISRPREEMPT** | **Interrupt Preempt Bit (Read Only)**<br>If set, a pending exception will be serviced on exit from the debug halt state. |
| [22] | **ISRPENDING** | **Interrupt Pending Flag, Excluding NMI and Faults (Read Only)**<br>0 = Interrupt not pending.<br>1 = Interrupt pending. |
| [21:18] | **Reserved** | Reserved. |
| [17:12] | **VECTPENDING** | **Number of the Highest Pended Exception**<br>Indicates the exception number of the highest priority pending enabled exception.<br>0 = No pending exceptions.<br>Non-zero = The exception number of the highest priority pending enabled exception.<br>**Note:** The value indicated by this field includes the effect of the BASEPRI and FAULTMASK registers, but does not include any effect of the PRIMASK register. |
| [11] | **RETTOBASE** | **Preempted Active Exceptions Indicator**<br>Indicate whether there are preempted active exceptions.<br>0 = There are preempted active exceptions to execute.<br>1 = There are no active exceptions, or the currently-executing exception is the only active exception. |
| [10:6] | **Reserved** | Reserved. |
| [5:0] | **VECTACTIVE** | **Number of the Current Active Exception**<br>0 = Thread mode.<br>Non-zero = The exception number of the currently active exception. |

## Application Interrupt and Reset Control Register (AIRCR)

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

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

| Bits | Description | |
|---|---|---|
| [31:16] | **VECTORKEY** | **Register Access Key**<br>When writing this register, this field should be 0x05FA; otherwise, the write action will be unpredictable.<br>The VECTORKEY filed is used to prevent accidental write to this register from resetting the system or clearing of the exception status. |
| [15] | **ENDIANNESS** | **Data Endianness**<br>0 = Little-endian.<br>1 = Big-endian. |
| [14:11] | **Reserved** | Reserved. |
| [10:8] | **PRIGROUP** | **Interrupt Priority Grouping**<br>This field determines the Split of Group priority from subpriority, |
| [7:3] | **Reserved** | Reserved. |
| [2] | **SYSRESETREQ** | **System Reset Request**<br>Writing this bit to 1 will cause a reset signal to be asserted to the chip and indicate a reset is requested<br>**Note:** This bit is write only and self-cleared as part of the reset sequence. |
| [1] | **VECTCLRACTIVE** | **Exception Active Status Clear Bit**<br>Setting this bit to 1 will clear all active state information for fixed and configurable exceptions<br>**Note1:** This bit is write only and can only be written when the core is halted.<br>**Note2:** It is the debugger’s responsibility to re-initialize the stack. |
| [0] | **Reserved** | Reserved. |

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

Table 6.2-6 Priority Grouping

## System Control Register (SCR)

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

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

| Bits | Description | |
|---|---|---|
| [31:5] | **Reserved** | Reserved. |
| [4] | **SEVONPEND** | **Send Event on Pending**<br>0 = Only enabled interrupts or events can wake up the processor, while disabled interrupts are excluded.<br>1 = Enabled events and all interrupts, including disabled interrupts, can wake up the processor.<br>When an event or interrupt enters pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE.<br>The processor also wakes up on execution of an SEV instruction or an external event. |
| [3] | **Reserved** | Reserved. |
| [2] | **SLEEPDEEP** | **Processor Deep Sleep and Sleep Mode Selection**<br>Control whether the processor uses Idle mode or Power-down mode as its low power operation.<br>0 = Idle mode.<br>1 = Power-down mode. |
| [1] | **SLEEPONEXIT** | **Sleep-on-exit Enable Control**<br>This bit indicates Sleep-On-Exit when returning from handler mode to thread mode.<br>0 = Do not enter idle mode when returning to Thread mode.<br>1 = Enter Idle mode, or Power-down mode, on return from an ISR to Thread mode.<br>Setting this bit to 1 will enable an interrupt driven application to avoid returning to an empty main application. |
| [0] | **Reserved** | Reserved. |

### System Handler PriorityRegister 1 (SHPR1)

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

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:16] | **PRI_6** | **Priority of System Handler 6 – UsageFault**<br>“0” denotes the highest priority and “3” denotes the lowest priority. |
| [15:8] | **PRI_5** | **Priority of System Handler 5 – BusFault**<br>“0” denotes the highest priority and “3” denotes the lowest priority. |
| [7:0] | **PRI_4** | **Priority of System Handler 4 – MemManage**<br>“0” denotes the highest priority and “3” denotes the lowest priority. |

## System Handler PriorityRegister 2 (SHPR2)

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

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

| Bits | Description | |
|---|---|---|
| [31:30] | **PRI_11** | **Priority of System Handler 11 – SVCall**<br>“0” denotes the highest priority and “3” denotes the lowest priority. |
| [29:0] | **Reserved** | Reserved. |

### System Handler PriorityRegister 3 (SHPR3)

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

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

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

## 6.3 Clock Controller

### 6.3.1 Overview

The clock controller generates clocks for the whole chip. The clocks include AHB, APB and engine clocks for all of devices like USB device, USB host, UART and so on. There are two PLL clocks, PLL and APLL, derived from external HXT clock input. The PLL clock allows the processor to operate at a high internal clock frequency. Also, the APLL is used to generate more accuracy frequency for audio CODEC. They also implement the power control function, include the individually clock on or off control register, clock source selector and divider. These functions minimize the extra power consumption and the chip runs on the just right condition. In Power-down mode, the controller turns off the crystal oscillator to minimize the chip power consumption.

### 6.3.2 Clock Diagram

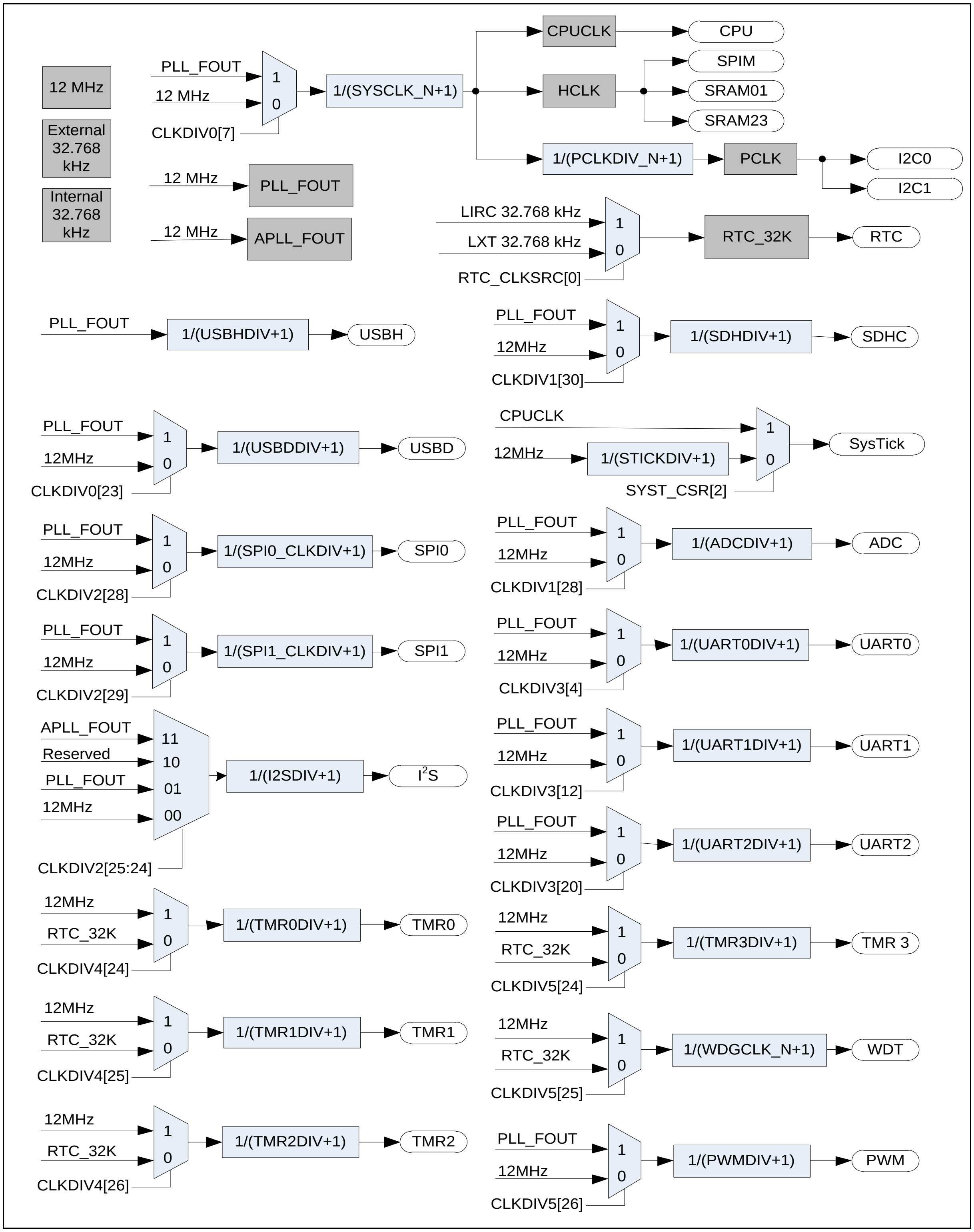


Figure 6.3-1 Clock Generator Global View Diagram

NUC505 SERIES TECHNICAL REFERENCE MANUAL

### 6.3.3 Clock Generator

The clock generator consists of 4 clock sources, which are listed below:

- Real-time clock (RTC_CLK) source can be selected from external 32.768 kHz external low speed crystal oscillator (LXT) or 32.768 kHz internal low speed RC oscillator (LIRC)
- 12 MHz external high speed crystal oscillator (HXT)
- Programmable System PLL output clock frequency (PLL_FOUT)
- Programmable Audio PLL output clock frequency (APLL_FOUT)

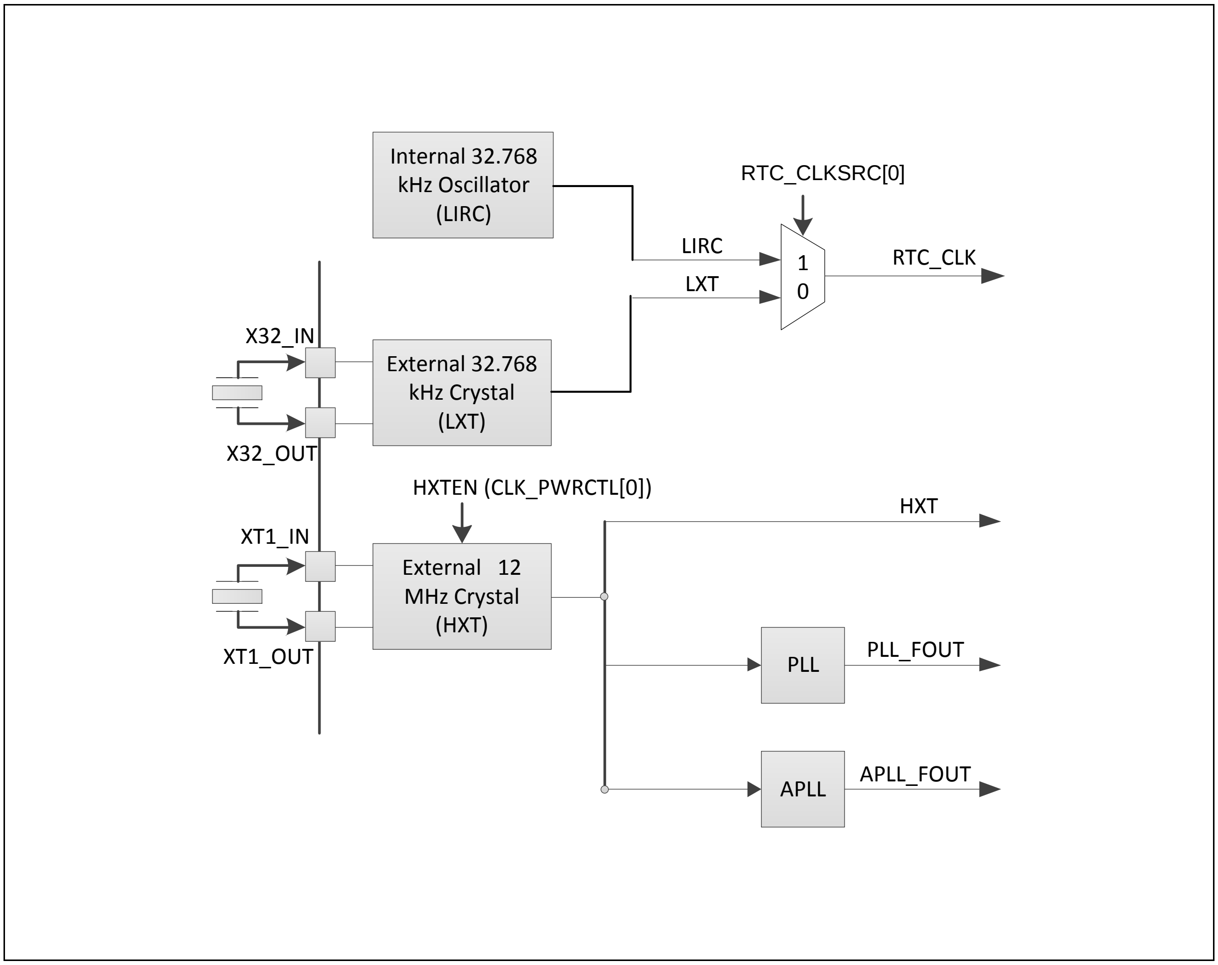


Figure 6.3-2 Clock Generator Block Diagram

The external crystal oscillator and two capacitors are connected to the pad “XT1_IN / X32_IN” and pad “XT1_OUT / X32_OUT”. The capacitance value of the two capacitors may be changed for differential crystal oscillator from different vender. The load capacitance values and resistance values must be adjusted according to the selected oscillator. The recommended load capacitance values and resistance values as

| Crystal Oscillator | Capacitance Values | Resistance Values |
|---|---|---|

| | | |
|---|---|---|
| 12 MHz | 20pF | 1 MΩ |
| 32.768 kHz | 33pF | 10 MΩ |

Table 6.3-1 Recommended Load Capacitance Values and Resistance Values.

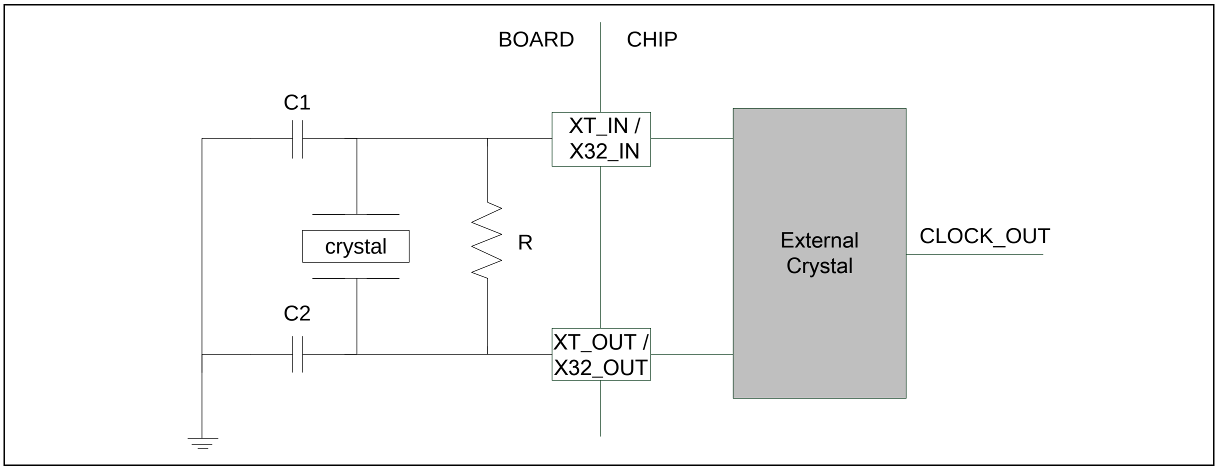


Figure 6.3-3 Crystal Oscillator Circuit

### 6.3.4 Power-down Mode Clock

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

The clocks which still keep active are listed below:

- Clock Generator
  - 32.768 kHz internal low speed RC oscillator (LIRC) clock
  - 32.768 kHz external low speed crystal oscillator (LXT) clock

In Power-down mode, If the woke-up even occurred, the disabled clocks will be regenerated after PDWKPSC (CLK_PWRCTL[23:8]) x 256 HXT cycle.

### 6.3.5 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CLKCTRL Base Address:**<br>**CLK_BA = 0x4000_0200** | | | | |
| **CLK_PWRCTL** | CLK_BA+0x00 | R/W | System Power-down Control Register | 0x00FF_FF03 |
| **CLK_AHBCLK** | CLK_BA+0x04 | R/W | AHB Devices Clock Enable Control Register | 0x0000_000F |
| **CLK_APBCLK** | CLK_BA+0x08 | R/W | APB Devices Clock Enable Control Register | 0x0000_0000 |
| **CLK_CLKDIV0** | CLK_BA+0x10 | R/W | Clock Divider Number Control Register 0 | 0x0001_0000 |
| **CLK_CLKDIV1** | CLK_BA+0x14 | R/W | Clock Divider Number Control Register 1 | 0x0000_0000 |
| **CLK_CLKDIV2** | CLK_BA+0x18 | R/W | Clock Divider Number Control Register 2 | 0x0000_0000 |
| **CLK_CLKDIV3** | CLK_BA+0x1C | R/W | Clock Divider Number Control Register 3 | 0x0000_0000 |
| **CLK_PLLCTL** | CLK_BA+0x20 | R/W | PLL Control Register | 0x0000_20CF |
| **CLK_APLLCTL** | CLK_BA+0x28 | R/W | APLL Control Register | 0x0003_0000 |
| **CLK_CLKDIV4** | CLK_BA+0x30 | R/W | Clock Divider Number Control Register 4 | 0x0000_0000 |
| **CLK_CLKDIV5** | CLK_BA+0x34 | R/W | Clock Divider Number Control Register 5 | 0x0000_0000 |

### 6.3.6 Register Description

**System Power-down Control Register （CLK_PWRCTL）**

The chip clock source is from an external crystal. The crystal oscillator can be control on/off by the register HXTEN. When turn off the crystal, the chip into power down state. To avoid outputting an unstable clock to system, clock controller implements a PDWKPSC counter. After the clock counter count PDWKPSC (CLK_PWRCTL[23:8]) x 256 crystal cycle, the clock controller starts to output the clock to system.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CLK_PWRCTL** | CLK_BA+0x00 | R/W | System Power-down Control Register | 0x00FF_FF03 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | PDWTCPU |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PDWKPSC | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PDWKPSC | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | PDWKIEN | PDWKIF | HXTCTL | HXTEN |

| Bits | Description | |
|---|---|---|
| [31:25] | **Reserved** | Reserved. |
| [24] | **PDWTCPU** | **Control Power-down Entry Condition**<br>0 = Chip enters Power-down mode when the HXTEN bit is set to 1.<br>1 = Chip enters Power-down mode when the both PDWTCPU and HXTEN bits are set to 1 and CPU run WFI instruction. |
| [23:8] | **PDWKPSC** | **PDWKPSC Counter**<br>Assuming the HXT is stable after the PDWKPSC x 256 HXT cycles, Clock controller would not output clock to system before the counter reaches (PDWKPSC x 256). |
| [7:4] | **Reserved** | Reserved. |
| [3] | **PDWKIEN** | **Power-down Mode Wake-up Interrupt Enable Control**<br>0 = Power-down Mode Wake-up Interrupt Disabled.<br>1 = Power-down Mode Wake-up Interrupt Enabled.<br>**Note:** The interrupt will occur when both PDWKIF and PDWKIEN are set |
| [2] | **PDWKIF** | **Power-down Mode Wake-up Interrupt Flag**<br>Set by “power down wake-up event” indicates that resume from Power-down mode”<br>The flag is set if the GPIO, USBH, USBD, UART, TIMER, WDT, RTC or I$^2$C wake-up occurred.<br>**Note1:** Write 1 to clear the bit to zero.<br>**Note2:** This bit works only if PDWKIEN (CLK_PWRCTL[24]) set to 1 |

|  |  |  |
|---|---|---|
| [1] | **HXTCTL** | **Power-down Mode Wake-up Pre-divider Counter Enable Control**<br>The HXT pre-divider controls wake-up time from Power-down mode. The chip will delay (PDWKPSC*256) cycles to wait until the HXT is stable after the reset signal.<br>0 = PDWKPSC counter Disabled.<br>1 = PDWKPSC counter Enabled. |
| [0] | **HXTEN** | **Crystal (Power Down) Control**<br>0 = Crystal off (Power down).<br>1 = Crystal on (Normal operation). |

| Register/Instruction Mode | HXTEN | PDWKIEN | PDWTCPU | CPU Run WFI Instruction | Clock Disable |
|---|---|---|---|---|---|
| **Normal Operation** | 1 | 0 | 0 | NO | All clocks are controlled by control register. |
| **Power-down Mode** | 0 | 1 | 1 | YES | Most clocks are disabled except LIRC/LXT, and only RTC/WDT/Timer peripheral clocks still enable if their clock sources are selected as LIRC/LXT. |

Table 6.3-2 Power-down Mode Control Table

When the chip enters Power-down mode, user can wake up chip by some interrupt sources. User should enable the related interrupt sources and NVIC IRQ enable bits (NVIC_ISER) before setting the HXTEN bit in CLK_PWRCTL[0] to ensure chip can enter Power-down and wake up successfully.

### AHB Devices Clock Enable Control Register （CLK_AHBCLK）

These register bits are used to enable/disable clock for AMBA clock, AHB engine and peripheral.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CLK_AHBCLK** | CLK_BA+0x04 | R/W | AHB Devices Clock Enable Control Register | 0x0000_000F |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| Reserved | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | | | | | | USBHCKEN | Reserved |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Reserved | USBDCKEN | SDHCKEN | Reserved | SPIMCKEN | ROMCKEN | SRAM23CKEN | SRAM01CKEN |

| Bits | Description | |
|---|---|---|
| [31:10] | **Reserved** | Reserved. |
| [9] | **USBHCKEN** | **USB Host Clock Enable Control**<br>0 = USB Host Clock Disabled.<br>1 = USB Host Clock Enabled. |
| [8:7] | **Reserved** | Reserved. |
| [6] | **USBDCKEN** | **USB Device Clock Enable Control**<br>0 = USB Device Clock Disabled.<br>1 = USB Device Clock Enabled. |
| [5] | **SDHCKEN** | **SDH Clock Enable Control**<br>0 = SDH Clock Disabled.<br>1 = SDH Clock Enabled. |
| [4] | **Reserved** | Reserved. |
| [3] | **SPIMCKEN** | **SPIM Clock Enable Control**<br>0 = SPIM Clock Disabled.<br>1 = SPIM Clock Enabled. |
| [2] | **ROMCKEN** | **ROM Clock Enable Control**<br>0 = ROM Clock Disabled.<br>1 = ROM Clock Enabled. |
| [1] | **SRAM23CKEN** | **SRAM#2 Clock Enable Control**<br>0 = SRAM#2 Clock Disabled.<br>1 = SRAM#2 Clock Enabled. |

|  |  |  |
|---|---|---|
| [0] | **SRAM01CKEN** | **SRAM#1 Clock Enable Control**<br>0 = SRAM#1 Clock Disabled.<br>1 = SRAM#1 Clock Enabled. |

## APB Devices Clock Enable Control Register （CLK_APBCLK）

These register bits are used to enable/disable clock for APB engine and peripheral.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CLK_APBCLK** | CLK_BA+0x08 | R/W | APB Devices Clock Enable Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **ADCCKEN** | **I2SCKEN** | **UART2CKEN** | **UART1CKEN** | **UART0CKEN** | **SPI1CKEN** | **SPI0CKEN** | **PWMCKEN** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **RTCCKEN** | **I2C1CKEN** | **I2C0CKEN** | **WDTCKEN** | **TMR3CKEN** | **TMR2CKEN** | **TMR1CKEN** | **TMR0CKEN** |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15] | **ADCCKEN** | **ADC Clock Enable Control**<br>0 = ADC Clock Disabled.<br>1 = ADC Clock Enabled. |
| [14] | **I2SCKEN** | **I$^2$S Clock Enable Control**<br>0 = I$^2$S Clock Disabled.<br>1 = I$^2$S Clock Enabled. |
| [13] | **UART2CKEN** | **UART2 Clock Enable Control**<br>0 = UART2 Clock Disabled.<br>1 = UART2 Clock Enabled. |
| [12] | **UART1CKEN** | **UART1 Clock Enable Control**<br>0 = UART1 Clock Disabled.<br>1 = UART1 Clock Enabled. |
| [11] | **UART0CKEN** | **UART0 Clock Enable Control**<br>0 = UART0 Clock Disabled.<br>1 = UART0 Clock Enabled. |
| [10] | **SPI1CKEN** | **SPI1 Clock Enable Control**<br>0 = SPI1 Clock Disabled.<br>1 = SPI1 Clock Enabled. |
| [9] | **SPI0CKEN** | **SPI0 Clock Enable Control**<br>0 = SPI0 Clock Disabled.<br>1 = SPI0 Clock Enabled. |

| | | |
|---|---|---|
| [8] | **PWMCKEN** | **PWM Clock Enable Control**<br>0 = PWM Clock Disabled.<br>1 = PWM Clock Enabled. |
| [7] | **RTCCKEN** | **RTC Clock Enable Control**<br>0 = RTC Clock Disabled.<br>1 = RTC Clock Enabled. |
| [6] | **I2C1CKEN** | **I2C1 Clock Enable Control**<br>0 = I2C1 Clock Disabled.<br>1 = I2C1 Clock Enabled. |
| [5] | **I2C0CKEN** | **I2C0 Clock Enable Control**<br>0 = I2C0 Clock Disabled.<br>1 = I2C0 Clock Enabled. |
| [4] | **WDTCKEN** | **Watchdog Timer Clock Enable Control**<br>0 = Watchdog Timer Clock Disabled.<br>1 = Watchdog Timer Clock Enabled. |
| [3] | **TMR3CKEN** | **TIMER3 Clock Enable Control**<br>0 = TIMER3 Clock Disabled.<br>1 = TIMER3 Clock Enabled. |
| [2] | **TMR2CKEN** | **TIMER2 Clock Enable Control**<br>0 = TIMER2 Clock Disabled.<br>1 = TIMER2 Clock Enabled. |
| [1] | **TMR1CKEN** | **TIMER1 Clock Enable Control**<br>0 = TIMER1 Clock Disabled.<br>1 = TIMER1 Clock Enabled. |
| [0] | **TMR0CKEN** | **TIMER0 Clock Enable Control**<br>0 = TIMER0 Clock Disabled.<br>1 = TIMER0 Clock Enabled. |

### Clock Divider Number Control Register 0 (CLK_CLKDIV0)

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CLK_CLKDIV0** | CLK_BA+0x10 | R/W | Clock Divider Number Control Register 0 | 0x0001_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | **USBHDIV** | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **USBDSEL** | **Reserved** | | **USBDDIV** | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | **PCLKDIV** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **HCLKSEL** | **Reserved** | | | **HCLKDIV** | | | |

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27:24] | **USBHDIV** | **Defines the Clock Divider Number for USB Host**<br>The actual clock divider number is (USBHDIV+1). So,<br>USBH_CLK = USBH_SrcCLK / (USBHDIV+1). |
| [23] | **USBDSEL** | **USB Device Source Clock Select (USBD_SrcCLK)**<br>0 = USB Device Clock source from HXT.<br>1 = USB Device Clock source from PLL_FOUT. |
| [22:21] | **Reserved** | Reserved. |
| [20:16] | **USBDDIV** | **Defines the Clock Divider Number for USB Device**<br>The actual clock divider number is (USBDDIV+1). So,<br>USBD_CLK = USBD_SrcCLK / (USBDDIV+1). |
| [15:12] | **Reserved** | Reserved. |
| [11:8] | **PCLKDIV** | **Defines the Clock Divider Number for APB_CLK**<br>The actual clock divider number is (PCLKDIV+1). So,<br>APB_CLK = HCLK / (PCLKDIV+1). |
| [7] | **HCLKSEL** | **System Source Clock Select (SYS_SrcCLK)**<br>0 = System Clock source from HXT.<br>1 = System Clock source from PLL_FOUT. |
| [6:4] | **Reserved** | Reserved. |
| [3:0] | **HCLKDIV** | **Defines the Clock Divider Number for SYS_CLK**<br>The actual clock divider number is (HCLKDIV+1). So,<br>SYS_CLK = SYS_SrcCLK / (HCLKDIV+1). |

## Clock Divider Number Control Register 1 (CLK_CLKDIV1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CLK_CLKDIV1** | CLK_BA+0x14 | R/W | Clock Divider Number Control Register 1 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30] | **SDHSEL** | **SDH Clock Select (SDH_SrcCLK)**<br>0 = SDH Clock source from HXT.<br>1 = SDH Clock source from PLL_FOUT. |
| [29] | **Reserved** | Reserved. |
| [28] | **ADCSEL** | **ADC Clock Select (ADC_SrcCLK)**<br>0 = ADC Clock source from HXT.<br>1 = ADC Clock source from PLL_FOUT. |
| [27] | **Reserved** | Reserved. |
| [26:16] | **SDHDIV** | **Defines the Clock Divider Number for SDH**<br>The actual clock divider number is (SDHDIV+1). So,<br>SDH_CLK = SDH_SrcCLK / (SDHDIV+1). |
| [15:8] | **STICKDIV** | **Defines the Clock Divider Number for SYS_TICK**<br>The actual clock divider number is (STICKDIV+1). So,<br>SYS_TICK_CLK = SYS_TICK_SrcCLK / (STICKDIV+1). |
| [7:0] | **ADCDIV** | **Defines the Clock Divider Number for ADC**<br>The actual clock divider number is (ADCDIV+1). So,<br>ADC_CLK = ADC_SrcCLK / (ADCDIV+1). |

### Clock Divider Number Control Register 2 (CLK_CLKDIV2)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CLK_CLKDIV2** | CLK_BA+0x18 | R/W | Clock Divider Number Control Register 2 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | **SPI1SEL** | **SPI0SEL** | **Reserved** | | **I2SSEL** | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **I2SDIV** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:30] | **Reserved** | Reserved. |
| [29] | **SPI1SEL** | **SPI1 Engine Clock Select (SPI1SEL)**<br>0 = SPI1 Engine Clock source from HXT.<br>1 = SPI1 Engine Clock source from PLL_FOUT. |
| [28] | **SPI0SEL** | **SPI0 Engine Clock Select (SPI0SEL)**<br>0 = SPI0 Engine Clock source from HXT.<br>1 = SPI0 Engine Clock source from PLL_FOUT. |
| [27:26] | **Reserved** | Reserved. |
| [25:24] | **I2SSEL** | **I$^{2}$S Clock Select (I2S_SrcCLK)**<br>00 = I$^{2}$S Clock source from HXT.<br>01 = I$^{2}$S Clock source from PLL_FOUT.<br>10 = Reserved.<br>11 = I$^{2}$S Clock source from APLL_FOUT. |
| [23:8] | **Reserved** | Reserved. |
| [7:0] | **I2SDIV** | **Defines the Clock Divider Number for I$^{2}$S**<br>The actual clock divider number is (I2SDIV+1). So,<br>I2S_CLK = I2S_SrcCLK / (I2SDIV+1). |

## Clock Divider Number Control Register 3 (CLK_CLKDIV3)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CLK_CLKDIV3** | CLK_BA+0x1C | R/W | Clock Divider Number Control Register 3 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | **UART2SEL** | **UART2DIV** | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | **UART1SEL** | **UART1DIV** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | **UART0SEL** | **UART0DIV** | | | |

| Bits | Description | |
|---|---|---|
| [31:21] | **Reserved** | Reserved. |
| [20] | **UART2SEL** | **UART2 Source Clock Select (UART2_SrcCLK)**<br>0 = UART2 Source Clock source from HXT.<br>1 = UART2 Source Clock source from PLL_FOUT. |
| [19:16] | **UART2DIV** | **Defines the Clock Divider Number for UART2**<br>The actual clock divider number is (UART2DIV +1). So,<br>UART2_CLK = UART2_SrcCLK / (UART2DIV +1). |
| [15:13] | **Reserved** | Reserved. |
| [12] | **UART1SEL** | **UART1 Source Clock Select (UART1_SrcCLK)**<br>0 = UART1 Source Clock source from HXT.<br>1 = UART1 Source Clock source from PLL_FOUT. |
| [11:8] | **UART1DIV** | **Defines the Clock Divider Number for UART1**<br>The actual clock divider number is (UART1DIV +1). So,<br>UART1_CLK = UART1_SrcCLK / (UART1DIV +1). |
| [7:5] | **Reserved** | Reserved. |
| [4] | **UART0SEL** | **UART0 Source Clock Select (UART0_SrcCLK)**<br>0 = UART0 Source Clock source from HXT.<br>1 = UART0 Source Clock source from PLL_FOUT. |
| [3:0] | **UART0DIV** | **Defines the Clock Divider Number for UART0**<br>The actual clock divider number is (UART0DIV +1). So,<br>UART0_CLK = UART0_SrcCLK / (UART0DIV +1). |

### PLL Control Register （CLK_PLLCTL）

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CLK_PLLCTL** | CLK_BA+0x20 | R/W | PLL Control Register | 0x0000_20CF |

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

| Bits | Description | |
|---|---|---|
| [31:18] | **Reserved** | Reserved. |
| [17] | **PD** | **Power-down Mode**<br>0 = PLL in Normal mode.<br>1 = PLL in Power-down mode (Default). |
| [16] | **BP** | **PLL Bypass Control**<br>0 = PLL at Normal mode.<br>1 = Bypass Fin (i.e. Fout = XIN). |
| [15:13] | **OUTDIV** | **Output Divider Control (P)**<br>Set the output divider factor from 1 to 8. |
| [12:7] | **INDIV** | **Reference Input Divider (M)**<br>Set the input reference clock divider factor from 1 to 64. |
| [6:0] | **FBDIV** | **Feedback Divider Control (N)**<br>Set the feedback divider factor from 1 to 128 |

## Frequency calculation

The relationship between the PLL input clock frequency and the output frequency depends on the configuration of the internal dividers. The dividing ratio of the dividers is as follows.

| | | |
|---|---|---|
| **M** | **INDIV[5:0]+1** | **Input Divider** |
| **N** | FBDIV[6:0]+1 | Feedback divider |
| **P** | OUTDIV[2:0]+1 | Output divider |

In Locked Mode, the frequency relationship of the PLL blocks is defined by the following equations:

$$\mathrm{F}_{PLL\_clko} = \mathrm{F}_{HXT} * \frac{N}{M} * \frac{1}{P}$$

**Constraints:**

1. $5\mathrm{MHz} \leq \frac{\mathrm{F}_{HXT}}{M} \leq 80\mathrm{MHz}$
2. $300\mathrm{MHz} \leq \mathrm{F}_{PLL\_clko} * \mathrm{P} \leq 1000\mathrm{MHz}$
3. $\mathrm{M} \geq 1\,, \mathrm{N} \geq 1$

Here $\mathrm{F}_{\mathrm{PLL\_clko}}$ is the output clock frequency of PLL. $\mathrm{F}_{\mathrm{HXT}}$ is the input clock frequency.

### APLL Control Register (CLK_APLLCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CLK_APLLCTL** | CLK_BA+0x28 | R/W | APLL Control Register | 0x0003_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:20] | **FRAC** | **Sigma-delta Modulator Control Pins**<br>Set the fractional number of the Feedback divider.<br>$Franction\_number = FRAC[11:0]/2^{12}$ |
| [19] | **Reserved** | Reserved. |
| [18] | **MODE** | **Mode Select**<br>0 = Integer mode;.<br>1 = Fraction mode. |
| [17] | **Reserved** | Reserved. |
| [16] | **PD** | **Power Down Enable Control**<br>0 = Power down Disabled.<br>1 = Power down Enabled. |
| [15:13] | **OUTDIV** | **Output Divider Control**<br>Set the Output divider factor from 1 to 8.<br>P = OUTDIV[2:0]+1. |
| [12:6] | **FBDIV** | **Feedback Divider Control**<br>Set the Feedback divider factor from 1 to 128.<br>N = FBDIV[6:0]+1. |
| [5:0] | **INDIV** | **Reference Input Divider Control**<br>Set the reference divider factor from 1 to 64.<br>M = INDIV[5:0]+1. |

## Frequency calculation in Integer Mode

The relationship between the APLL input clock frequency and the output frequency depends on the configuration of the internal dividers. The dividing ratio of the dividers is:

| **M** | **INDIV[5:0]+1** | **Input Divider** |
|---|---|---|
| N | FBDIV[6:0]+1 | Feedback divider |
| P | OUTDIV[2:0]+1 | Output divider |

In Locked Mode, the frequency relationship of the APLL blocks is illustrated by the following equations:

$$\mathrm{F}_{APLL\_clko} = \mathrm{F}_{HXT} * \frac{N}{M} * \frac{1}{P}$$

**Constrain:**

1. $2.5\mathrm{MHz} \leq \frac{\mathrm{F}_{HXT}}{M} \leq 80\mathrm{MHz}$
2. $200\mathrm{MHz} \leq \mathrm{F}_{APLL\_clko} * \mathrm{P} \leq 500\mathrm{MHz}$
3. $\mathrm{M} \geq 1\,, \mathrm{N} \geq 1$

Here $\mathrm{F}_{\mathrm{APLL\_clko}}$ is the output clock frequency of PLL. $\mathrm{F}_{\mathrm{HXT}}$ is the input clock frequency.

### Frequency calculation in Fraction Mode

In this mode, the configuration of the internal dividers is the same as that in Integer Mode:

| **M** | **INDIV[5:0]+1** | **Input Divider** |
|---|---|---|
| N | FBDIV[6:0]+1 | Feedback divider |
| P | OUTDIV[2:0]+1 | Output divider |

The difference is that by setting FRAC[11:0], one can get a fractional divider number N.X rather than N in Integer Mode, where:

X=FRAC[11:0]/ $2^{12}$ ; Fractional number

N.X=N+X ; Fractional divider

The frequency relationship of the PLL blocks and the output clock can be calculated by the following equations:

$$\mathrm{F}_{APLL\_clko} = \mathrm{F}_{HXT} * \frac{N.X}{M} * \frac{1}{P}$$

**Example:**

1. $\mathrm{F}_{\mathrm{HXT}}$ = 12MHz, $\mathrm{F}_{\mathrm{APLL\_clko}}$ *P= 301.5MHz;
2. Calculation: (12MHz / 2) * 50.25 / 1=301.5MHz

The equation means that $\mathrm{F}_{\mathrm{HXT}}$/M=6MHz, M= 2, N.X=50.25, P=1

To get fractional divider number 50.25, use the equation below:

50.25 = 50+($2^{10}$ / $2^{12}$)

So N = 50, X=0.25, FRAC=$2^{10}$

3. Configuration:

   INDIV[5:0] = 000001,

   FBDIV[6:0] = 0110001,

   OUTDIV[2:0] = 000,

   FRAC[11:0] = 010000000000.

4. Frequency Step: Adjust FRAC from $2^{10}$ to $2^{10}$+1, and the fractional divider number changes to 50.250244. Then $F_{APLL\_clko}$ changes to 6MHz * 50.250244 =301.501464MHz. Thus the frequency step is 301.501464 MHz – 301.5 MHz = 1.464 kHz.

### Clock Divider Number Control Register 4 (CLK_CLKDIV4)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CLK_CLKDIV4** | CLK_BA+0x30 | R/W | Clock Divider Number Control Register 4 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | **TMR2SEL** | **TMR1SEL** | **TMR0SEL** |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **TMR2DIV** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **TMR1DIV** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **TMR0DIV** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26] | **TMR2SEL** | **Timer2 Engine Clock Select (TMR2_SrcCLK)**<br>0 = Timer2 Engine Clock source from RTC_CLK.<br>1 = Timer2 Engine Clock source from HXT. |
| [25] | **TMR1SEL** | **Timer1 Engine Clock Select (TMR1_SrcCLK)**<br>0 = Timer1 Engine Clock source from RTC_CLK.<br>1 = Timer1 Engine Clock source from HXT. |
| [24] | **TMR0SEL** | **Timer0 Engine Clock Select (TMR0_SrcCLK)**<br>0 = Timer0 Engine Clock source from RTC_CLK.<br>1 = Timer0 Engine Clock source from HXT. |
| [23:16] | **TMR2DIV** | **Defines the Clock Divider Number for TMR2**<br>The actual clock divider number is (TMR2DIV+1). So,<br>TMR2_CLK = TMR2_SrcCLK / (TMR2CLK_N+1). |
| [15:8] | **TMR1DIV** | **Defines the Clock Divider Number for TMR1**<br>The actual clock divider number is (TMR1DIV+1). So,<br>TMR1_CLK = TMR1_SrcCLK / (TMR1DIV+1). |
| [7:0] | **TMR0DIV** | **Defines the Clock Divider Number for TMR0**<br>The actual clock divider number is (TMR0DIV+1). So,<br>TMR0_CLK = TMR0_SrcCLK / (TMR0DIV+1). |

### Clock Divider Number Control Register 5 (CLK_CLKDIV5)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CLK_CLKDIV5** | CLK_BA+0x34 | R/W | Clock Divider Number Control Register 5 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | PWMSEL | WDTSEL | TMR3SEL |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PWMDIV | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| WDTDIV | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TMR3DIV | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:27] | **Reserved** | Reserved. |
| [26] | **PWMSEL** | **PWM Engine Clock Select (PWM_SrcCLK)**<br>0 = PWM Engine Clock source from HXT.<br>1 = PWM Engine Clock source from PLL_FOUT. |
| [25] | **WDTSEL** | **WDT Engine Clock Select (WDT_SrcCLK)**<br>0 = WDT Engine Clock source from RTC_CLK.<br>1 = WDT Engine Clock source from HXT. |
| [24] | **TMR3SEL** | **Timer3 Engine Clock Select (TMR3_SrcCLK)**<br>0 = Timer3 Engine Clock source from RTC_CLK.<br>1 = Timer3 Engine Clock source from HXT. |
| [23:16] | **PWMDIV** | **Define the Clock Divider Number for PWM**<br>The actual clock divider number is (PWMDIV+1). So,<br>PWM_CLK = PWM_SrcCLK / (PWMDIV+1). |
| [15:8] | **WDTDIV** | **Define the Clock Divider Number for WDT**<br>The actual clock divider number is (WDTDIV+1). So,<br>WDT_CLK = WDT_SrcCLK / (WDTDIV+1). |
| [7:0] | **TMR3DIV** | **Define the Clock Divider Number for TMR3**<br>The actual clock divider number is (TM3DIV+1). So,<br>TMR3_CLK = TMR3_SrcCLK / (TMR3DIV+1). |

## 6.4 General Purpose I/O (GPIO)

### 6.4.1 Overview

The NUC505 series has up to 52 General Purpose I/O pins to be shared with other function pins depending on the chip configuration. The 52 pins are arranged in 4 ports named as PA, PB, PC, and PD. PA and PB have 16 pins on port, PC has 15 pins on port, and PD has 5 pins on port. Each of the 52 pins is independent and has the corresponding register bits to control the pin mode function and data.

The I/O type of each I/O pins can be configured by software individually as Input or Push-pull output mode. After the chip is reset, the I/O mode of all pins is input mode with no pull-up and pull-down enable (except PB.2, it is pull-up enable). Each I/O pin has an individual pull-up and pull-down resistor which is about 40 kΩ ~ 50 kΩ for VDD and Vss. User can set Px_PUEN to control I/O pins to pull-up or pull-down.

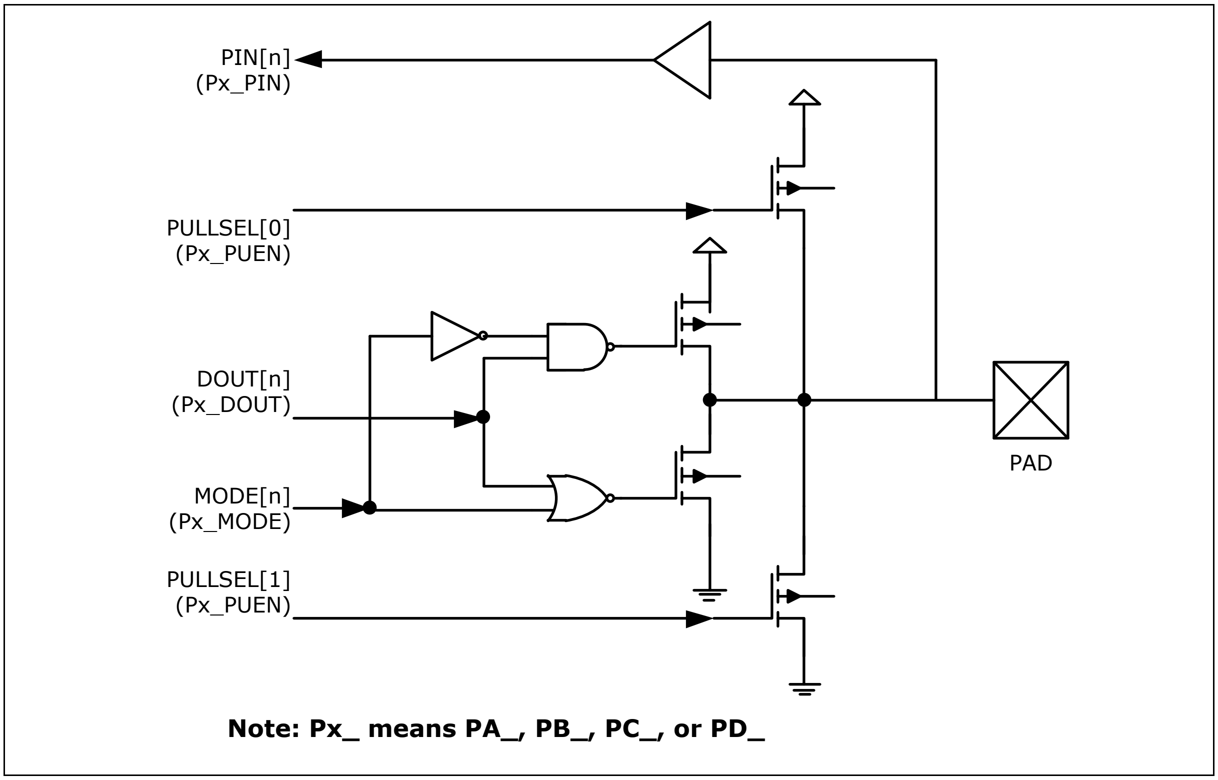


Figure 6.4-1 I/O Pin Block Diagram

### 6.4.2 Features

- Two I/O modes:
  - Push-Pull Output mode
  - Input only with high impendence mode
- CMOS/Schmitt trigger input selectable (refers SYS_GPAIBE register on section 6.2.9)
- I/O pin can be configured as interrupt source with edge setting

- I/O pin has individual internal pull-up resistor and pull-down resistor
- Enabling the pin interrupt function will also enable the wake-up function

### 6.4.3 Basic Configuration

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

### 6.4.4 Functional Description

Figure 6.4-1Figure 6.4-1 shows the structure of GPIO pad. It supports input mode and push-pull output mode. It has individual pull-up and pull-down resistor.

#### *6.4.4.1 Input Mode*

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

#### *6.4.4.2 Push-pull Output Mode*

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

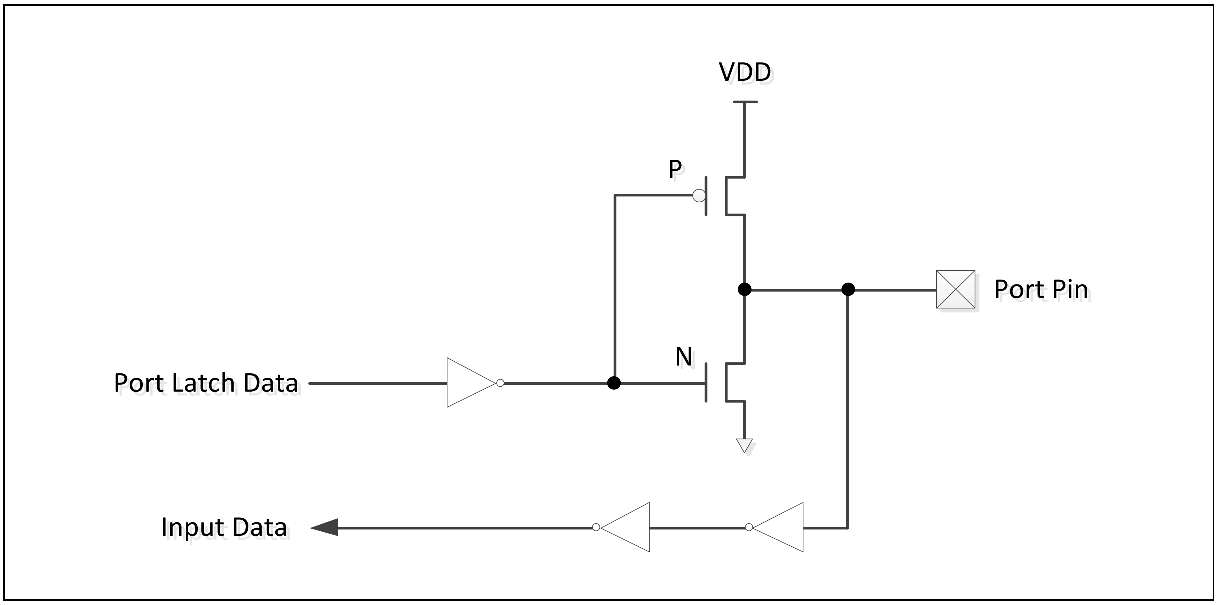


Figure 6.4-2 Push-Pull Output

### 6.4.5 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GPIO Base Address:**<br>**GPIO_BA = 0x400E_3000** | | | | |
| **PA_MODE** | GPIO_BA+0x00 | R/W | PA I/O Mode Control | 0x0000_0000 |
| **PA_PUEN** | GPIO_BA+0x04 | R/W | PA I/O Pull-up/down Resistor Control | 0x0000_0000 |
| **PA_DOUT** | GPIO_BA+0x08 | R/W | PA Data Output Value | 0x0000_0000 |
| **PA_PIN** | GPIO_BA+0x0C | R | PA Pin Value | 0x0000_XXXX |
| **PB_MODE** | GPIO_BA+0x10 | R/W | PB I/O Mode Control | 0x0000_0000 |
| **PB_PUEN** | GPIO_BA+0x14 | R/W | PB I/O Pull-up/down Resistor Control | 0x0000_0010 |
| **PB_DOUT** | GPIO_BA+0x18 | R/W | PB Data Output Value | 0x0000_0000 |
| **PB_PIN** | GPIO_BA+0x1C | R | PB Pin Value | 0x0000_XXXX |
| **PC_MODE** | GPIO_BA+0x20 | R/W | PC I/O Mode Control | 0x0000_0000 |
| **PC_PUEN** | GPIO_BA+0x24 | R/W | PC I/O Pull-up/down Resistor Control | 0x0000_0000 |
| **PC_DOUT** | GPIO_BA+0x28 | R/W | PC Data Output Value | 0x0000_0000 |
| **PC_PIN** | GPIO_BA+0x2C | R | PC Pin Value | 0x0000_XXXX |
| **PD_MODE** | GPIO_BA+0x30 | R/W | PD I/O Mode Control | 0x0000_0000 |
| **PD_PUEN** | GPIO_BA+0x34 | R/W | PD I/O Pull-up/down Resistor Control | 0x0000_0000 |
| **PD_DOUT** | GPIO_BA+0x38 | R/W | PD Data Output Value | 0x0000_0000 |
| **PD_PIN** | GPIO_BA+0x3C | R | PD Pin Value | 0x0000_00XX |
| **GPIO_DBCTL** | GPIO_BA+0x70 | R/W | Interrupt Event (EINT) De-bounce Control | 0x0000_0000 |
| **PA_INTSRCGP** | GPIO_BA+0x80 | R/W | PA Interrupt Event (EINT) Source Grouping | 0x0000_0000 |
| **PB_INTSRCGP** | GPIO_BA+0x84 | R/W | PB Interrupt Event (EINT) Source Grouping | 0x5555_5555 |
| **PC_INTSRCGP** | GPIO_BA+0x88 | R/W | PC Interrupt Event (EINT) Source Grouping | 0x2AAA_AAAA |
| **PD_INTSRCGP** | GPIO_BA+0x8C | R/W | PD Interrupt Event (EINT) Source Grouping | 0x0000_03FF |
| **PA_INTEN** | GPIO_BA+0x90 | R/W | PA Interrupt Enable Control Register | 0x0000_0000 |
| **PB_INTEN** | GPIO_BA+0x94 | R/W | PB Interrupt Enable Control Register | 0x0000_0000 |
| **PC_INTEN** | GPIO_BA+0x98 | R/W | PC Interrupt Enable Control Register | 0x0000_0000 |
| **PD_INTEN** | GPIO_BA+0x9C | R/W | PD Interrupt Enable Control Register | 0x0000_0000 |
| **GPIO_INTCTL** | GPIO_BA+0xA0 | R/W | Interrupt Latch Trigger Selection Register | 0x0000_0000 |
| **PA_LATCHDAT** | GPIO_BA+0xA4 | R | PA Interrupt Latch Value | 0x0000_0000 |
| **PB_LATCHDAT** | GPIO_BA+0xA8 | R | PB Interrupt Latch Value | 0x0000_0000 |
| **PC_LATCHDAT** | GPIO_BA+0xAC | R | PC Interrupt Latch Value | 0x0000_0000 |

NUC505 SERIES TECHNICAL REFERENCE MANUAL

| | | | | |
|---|---|---|---|---|
| **PD_LATCHDAT** | GPIO_BA+0xB0 | R | PD Interrupt Latch Value | 0x0000_0000 |
| **GPIO_INTSTSA_B** | GPIO_BA+0xB4 | R/W | EINT0~3 Interrupt Trigger Source Indicator from PA and PB | 0x0000_0000 |
| **GPIO_INTSTSC_D** | GPIO_BA+0xB8 | R/W | EINT0~3 Interrupt Trigger Source Indicator from PC and PD | 0x0000_0000 |

### 6.4.6 Register Description

**Port A-D I/O Mode Control (Px_MODE)**

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PA_MODE** | GPIO_BA+0x00 | R/W | PA I/O Mode Control | 0x0000_0000 |
| **PB_MODE** | GPIO_BA+0x10 | R/W | PB I/O Mode Control | 0x0000_0000 |
| **PC_MODE** | GPIO_BA+0x20 | R/W | PC I/O Mode Control | 0x0000_0000 |
| **PD_MODE** | GPIO_BA+0x30 | R/W | PD I/O Mode Control | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [n]<br>n = 0,1..15 | **MODE[n]** | **Port A-d I/O Pin[N] Mode Control**<br>Determine each I/O mode of Px.n pins.<br>0 = Px.n is in Input mode.<br>1 = Px.n is in Push-pull Output mode.<br>**Note:**<br>Max. n=15 for port A/B.<br>Max. n=14 for port C.<br>Max. n=4 for port D. |

### Port A-D Pin Pull-up/down Resistor Control (Px_PUEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PA_PUEN** | GPIO_BA+0x04 | R/W | PA I/O Pull-up/down Resistor Control | 0x0000_0000 |
| **PB_PUEN** | GPIO_BA+0x14 | R/W | PB I/O Pull-up/down Resistor Control | 0x0000_0010 |
| **PC_PUEN** | GPIO_BA+0x24 | R/W | PC I/O Pull-up/down Resistor Control | 0x0000_0000 |
| **PD_PUEN** | GPIO_BA+0x34 | R/W | PD I/O Pull-up/down Resistor Control | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PULLSEL15 | | PULLSEL14 | | PULLSEL3 | | PULLSEL2 | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PULLSEL11 | | PULLSEL10 | | PULLSEL9 | | PULLSEL8 | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PULLSEL7 | | PULLSEL6 | | PULLSEL5 | | PULLSEL4 | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PULLSEL3 | | PULLSEL2 | | PULLSEL1 | | PULLSEL0 | |

| Bits | Description | |
|---|---|---|
| [2n+1:2n]<br>n = 0,1..15 | **PULLSELn** | **Port A-d Pin Pull-up Resistor or Pull-down Resistor Enable Control**<br>00 = Px.n pull-up or pull-down resistors are all Disabled.<br>01 = Px.n pull-up resistor Enabled.<br>10 = Px.n pull-down resistor Enabled.<br>11 = Reserved.<br>**Note:**<br>Max. n=15 for port A/B.<br>Max. n=14 for port C.<br>Max. n=4 for port D.<br>**Note2:** Refer to Figure 6.4-1 |

### Port A-D Data Output Value (Px_DOUT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PA_DOUT** | GPIO_BA+0x08 | R/W | PA Data Output Value | 0x0000_0000 |
| **PB_DOUT** | GPIO_BA+0x18 | R/W | PB Data Output Value | 0x0000_0000 |
| **PC_DOUT** | GPIO_BA+0x28 | R/W | PC Data Output Value | 0x0000_0000 |
| **PD_DOUT** | GPIO_BA+0x38 | R/W | PD Data Output Value | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [n]<br>n = 0,1..15 | **DOUT[n]** | **Port A-d Pin[N] Output Value**<br>Each of these bits controls the status of a Px.n pin when the Px.n is configured as Push-pull output mode.<br>0 = Px.n will drive Low if the Px.n pin is configured as Push-pull output mode.<br>1 = Px.n will drive High if the Px.n pin is configured as Push-pull output mode.<br>**Note:**<br>Max. n=15 for port A/B.<br>Max. n=14 for port C.<br>Max. n=7 for port D. |

### Port A-D Pin Value (Px _PIN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PA_PIN** | GPIO_BA+0x0C | R | PA Pin Value | 0x0000_XXXX |
| **PB_PIN** | GPIO_BA+0x1C | R | PB Pin Value | 0x0000_XXXX |
| **PC_PIN** | GPIO_BA+0x2C | R | PC Pin Value | 0x0000_XXXX |
| **PD_PIN** | GPIO_BA+0x3C | R | PD Pin Value | 0x0000_00XX |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [n]<br>n = 0,1..15 | **PIN[n]** | **Port A-d Pin[N] Pin Value**<br>Each bit of the register reflects the actual status of the respective Px.n pin. If the bit is 1, it indicates the corresponding pin status is high; else the pin status is low.<br>**Note:**<br>Max. n=15 for port A/B.<br>Max. n=14 for port C.<br>Max. n=4 for port D. |

### Interrupt Event (EINT) De-bounce Control (GPIO_DBCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GPIO_DBCTL** | GPIO_BA+0x70 | R/W | Interrupt Event (EINT) De-bounce Control | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:4] | **DBCLKSEL** | **De-bounce Sampling Cycle Selection**<br>0000 = Sample interrupt input once per 1 APB clocks.<br>0001 = Sample interrupt input once per 2 APB clocks.<br>0010 = Sample interrupt input once per 4 APB clocks.<br>0011 = Sample interrupt input once per 8 APB clocks.<br>0100 = Sample interrupt input once per 16 APB clocks.<br>0101 = Sample interrupt input once per 32 APB clocks.<br>0110 = Sample interrupt input once per 64 APB clocks.<br>0111 = Sample interrupt input once per 128 APB clocks.<br>1000 = Sample interrupt input once per 256 APB clocks.<br>1001 = Sample interrupt input once per 512 APB clocks.<br>1010 = Sample interrupt input once per 1024 APB clocks.<br>1011 = Sample interrupt input once per 2048 APB clocks.<br>1100 = Sample interrupt input once per 4096 APB clocks.<br>1101 = Sample interrupt input once per 8192 APB clocks.<br>1110 = Sample interrupt input once per 16384 APB clocks.<br>1111 = Sample interrupt input once per 32768 APB clocks. |
| [n]<br>n = 0,1,2,3 | **DBEN[n]** | **EINT (EINT0~EINT3) De-bounce Enable Control**<br>The DBEN[n] bit is used to enable the de-bounce function for each corresponding EINT. If the input signal pulse width cannot be sampled by continuous two de-bounce sample cycle, the input signal transition is regarded as the signal bounce and will not trigger the interrupt. The de-bounce sample cycle period is controlled by DBCLKSEL (GPIO_DBCTL [7:4]).<br>0 = EINTn de-bounce function Disabled.<br>1 = EINTn de-bounce function Enabled. |

### Port A-D EINT Source Grouping (Px_INTSRCGP)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PA_INTSRCGP** | GPIO_BA+0x80 | R/W | PA Interrupt Event (EINT) Source Grouping | 0x0000_0000 |
| **PB_INTSRCGP** | GPIO_BA+0x84 | R/W | PB Interrupt Event (EINT) Source Grouping | 0x5555_5555 |
| **PC_INTSRCGP** | GPIO_BA+0x88 | R/W | PC Interrupt Event (EINT) Source Grouping | 0x2AAA_AAAA |
| **PD_INTSRCGP** | GPIO_BA+0x8C | R/W | PD Interrupt Event (EINT) Source Grouping | 0x0000_03FF |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| INTSEL15 | | INTSEL14 | | INTSEL13 | | INTSEL12 | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| INTSEL11 | | INTSEL10 | | INTSEL9 | | INTSEL8 | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| INTSEL7 | | INTSEL6 | | INTSEL5 | | INTSEL4 | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| INTSEL3 | | INTSEL2 | | INTSEL1 | | INTSEL0 | |

| Bits | Description | |
|---|---|---|
| [2n+1:2n]<br>n=0,1..15 | **INTSELn** | **Selection for Px.N As One of Interrupt Sources to EINT0, EINT1, EINT2, or EINT3**<br>00 = Px.n pin is selected as one of interrupt sources to EINT0.<br>01 = Px.n pin is selected as one of interrupt sources to EINT1.<br>10 = Px.n pin is selected as one of interrupt sources to EINT2.<br>11 = Px.n pin is selected as one of interrupt sources to EINT3.<br>**Note:**<br>Max. n=15 for port A/B.<br>Max. n=14 for port C.<br>Max. n=4 for port D. |

## Port A-D Interrupt Enable Control Register (Px_INTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PA_INTEN** | GPIO_BA+0x90 | R/W | PA Interrupt Enable Control Register | 0x0000_0000 |
| **PB_INTEN** | GPIO_BA+0x94 | R/W | PB Interrupt Enable Control Register | 0x0000_0000 |
| **PC_INTEN** | GPIO_BA+0x98 | R/W | PC Interrupt Enable Control Register | 0x0000_0000 |
| **PD_INTEN** | GPIO_BA+0x9C | R/W | PD Interrupt Enable Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [n+16]<br>n = 0,1..15 | **REIEN[n]** | **Port A-d Pin[N] Control Rising Edge of Input Px.N Pin to Trigger the Interrupt**<br>The RHIEN (Px_INTEN[n+16]) bit is used to enable the interrupt for each of the corresponding input Px.n pin. Set bit to 1 also enable the pin wake-up function.<br>When setting the RHIEN (Px_INTEN[n+16]) bit to 1 :<br>The input Px.n pin will generate the interrupt while this pin state changed from low to high.<br>0 = Px.n low to high interrupt Disabled.<br>1 = Px.n low to high interrupt Enabled.<br>**Note:**<br>Max. n=15 for port A/B.<br>Max. n=14 for port C.<br>Max. n=4 for port D. |
| [n]<br>n = 0,1..15 | **FEIEN[n]** | **Port A-d Pin[N] Control Falling Edge of Input Px.N Pin to Trigger the Interrupt**<br>The FLIEN (Px_INTEN[n]) bit is used to enable the interrupt for each of the corresponding input Px.n pin. Set bit to 1 also enable the pin wake-up function.<br>When setting the FLIEN (Px_INTEN[n]) bit to 1 :<br>The input Px.n pin will generate the interrupt while this pin state changed from high to low.<br>0 = Px.n high to low interrupt Disabled.<br>1 = Px.n high to low interrupt Enabled.<br>**Note:**<br>Max. n=15 for port A/B.<br>Max. n=14 for port C.<br>Max. n=4 for port D. |

**Note1:** Only edge-trigger interrupt is supported.

**Note2:** In Normal Operation mode, for each pin, FEIENn and REIENn can be both set to detect both rising and falling edge.

**Note3:** When a pin is used as Power-down wake up source, the setting of edges must be explained as level trigger. For example, if one pin is set for rising, user must keep this pin low while start to enter Power-down; a high level will make Power-down entrance be ignored. After entering Power-down, a high level at this pin will make chip leave Power-down.

**Note4:** When a pin is used as Power-down wake up source, if both edges are set, the high level will be set as wake-up level.

### Interrupt Latch Trigger Selection Register (GPIO_INTCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GPIO_INTCTL** | GPIO_BA+0xA0 | R/W | Interrupt Latch Trigger Selection Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | | **INTCTL** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **WKEN3** | **WKEN2** | **WKEN1** | **WKEN0** | **INTLHEN3** | **INTLHEN2** | **INTLHEN1** | **INTLHEN0** |

| Bits | Description | |
|---|---|---|
| [31:9] | **Reserved** | Reserved. |
| [8] | **INTCTL** | **Interrupt Request Source Control**<br>0 = When the GPIO interrupt occurs, the GPIO interrupt controller generates 1 APB clock pulse to the NVIC.<br>1 = When the GPIO interrupt occurs, the interrupt from GPIO to NVIC will keep till the CPU clear the interrupt trigger source. (GPIO_INTSTSA_B, GPIO_INTSTSC_D) |
| [n+4]<br>n = 0,1,2,3 | **WKENn** | **GPIO Interrupt Wake Up System Enable Control**<br>0 = No effect.<br>1 = EINTn can wake up the chip from Idle and Power-down mode. |
| [n]<br>n = 0,1,2,3 | **INTLHENn** | **Enable Latch PA.N/PB.N/PC.N/PD.N Value When EINTn Happen**<br>0 = No effect.<br>1 = When EINTn interrupt has happened, PA_LATCHDAT, PB_LATCHDAT, PC_LATCHDAT, and PD_LATCHDAT registers will latch PA, PB, PC, and PD port values within EINTn group. |

### Port A-D Interrupt Latch Value (Px_LATCHDAT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PA_LATCHDAT** | GPIO_BA+0xA4 | R | PA Interrupt Latch Value | 0x0000_0000 |
| **PB_LATCHDAT** | GPIO_BA+0xA8 | R | PB Interrupt Latch Value | 0x0000_0000 |
| **PC_LATCHDAT** | GPIO_BA+0xAC | R | PC Interrupt Latch Value | 0x0000_0000 |
| **PD_LATCHDAT** | GPIO_BA+0xB0 | R | PD Interrupt Latch Value | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **DAT15** | **DAT14** | **DAT13** | **DAT12** | **DAT11** | **DAT10** | **DAT9** | **DAT8** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **DAT7** | **DAT6** | **DAT5** | **DAT4** | **DAT3** | **DAT2** | **DAT1** | **DAT0** |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [n]<br>n = 0,1..15 | **DATn** | **Latch Px.N Interrupt Value**<br>Latched value of Px.n while the EINT (EINT0~EINT3) selected by GPIO_INTCTL is active.<br>**Note:**<br>Max. n=15 for port A/B.<br>Max. n=14 for port C.<br>Max. n=4 for port D. |

When a latched pin value is ‘0’, it indicates:

1. The pin is Input mode and is recognized as LOW
2. The pin is Output mode, so the input value is masked as ‘0’.

### EINT0~3 Interrupt Trigger Source Indicator from PA and PB (GPIO_INTSTSA_B)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| GPIO_INTSTSA_B | GPIO_BA+0xB4 | R/W | EINT0~3 Interrupt Trigger Source Indicator from PA and PB | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [n+16]<br>n = 0,1..15 | **PBIFn** | **Port B Pin[N] Interrupt Source Flag**<br>Write Operation:<br>0 = No action.<br>1 = Clear the corresponding pending interrupt.<br>Read Operation:<br>0 = No interrupt at PB.n1 = PB.n generates an interrupt. |
| [n]<br>n = 0,1..15 | **PAIFn** | **Port a Pin[N] Interrupt Source Flag**<br>Write Operation:<br>0 = No action.<br>1 = Clear the corresponding pending interrupt.<br>Read Operation:<br>0 = No interrupt at PA.n.<br>1 = PA.n generates an interrupt.<br>When this bit is read as “1”, it indicates that PA.n is a trigger source to generate the interrupt.<br>**Note:** Write 1 to clear the correspond interrupt source |

The trigger source will be latched when the corresponding rising or falling trigger enable is setup and the pin state toggle is recognized (through de-bounce or without de-bounce), no matter whether the source is an input or output pin.

### EINT0~3 Interrupt Trigger Source Indicator from PC and PD (GPIO_INTSTSC_D)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **GPIO_INTSTSC_D** | GPIO_BA+0xB8 | R/W | EINT0~3 Interrupt Trigger Source Indicator from PC and PD | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:21] | **Reserved** | Reserved. |
| [n+16]<br>n = 0,1..4 | **PDIFn** | **Port D Pin[N] Interrupt Source Flag**<br>Write Operation:<br>0 = No action.<br>1 = Clear the corresponding pending interrupt.<br>Read Operation:<br>0 = No interrupt at PD.n.<br>1 = PD.n generates an interrupt. |
| [15] | **Reserved** | Reserved. |
| [n]<br>n = 0,1..14 | **PCIFn** | **Port C Pin[N] Interrupt Source Flag**<br>Write Operation:<br>0 = No action.<br>1 = Clear the corresponding pending interrupt.<br>Read Operation:<br>0 = No interrupt at PC.n.<br>1 = PC.n generates an interrupt. |

The trigger source will be latched when the corresponding rising or falling trigger enable is set and the pin state toggle is recognized (through de-bounce or without de-bounce), no matter whether the source is an input or output pin.

## 6.5 Timer Controller (TIMER)

### 6.5.1 Overview

The Timer Controller includes four 32-bit timers, TIMER0 ~ TIMER3, allowing user to easily implement a timer control for applications. The timer can perform functions, such as frequency measurement, delay timing, clock generation, and event counting by external input pins, and interval measurement by external capture pins.

### 6.5.2 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) * CMPDAT (TIMERx_CMP[23:0])
- 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 TIMERx_CNT (Timer Data Register)
- Supports event counting function to count the event from external pin (TM0_CNT_OUT~TM3_CNT_OUT)
- Supports external capture pin (TM0_EXT~TM3_EXT) for interval measurement
- Supports external capture pin (TM0_EXT~TM3_EXT) to reset 24-bit up counter
- Supports chip wake-up from Idle mode, Power-down mode and Deep Power-down mode, if a timer interrupt signal is generated

### 6.5.3 Block Diagram

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

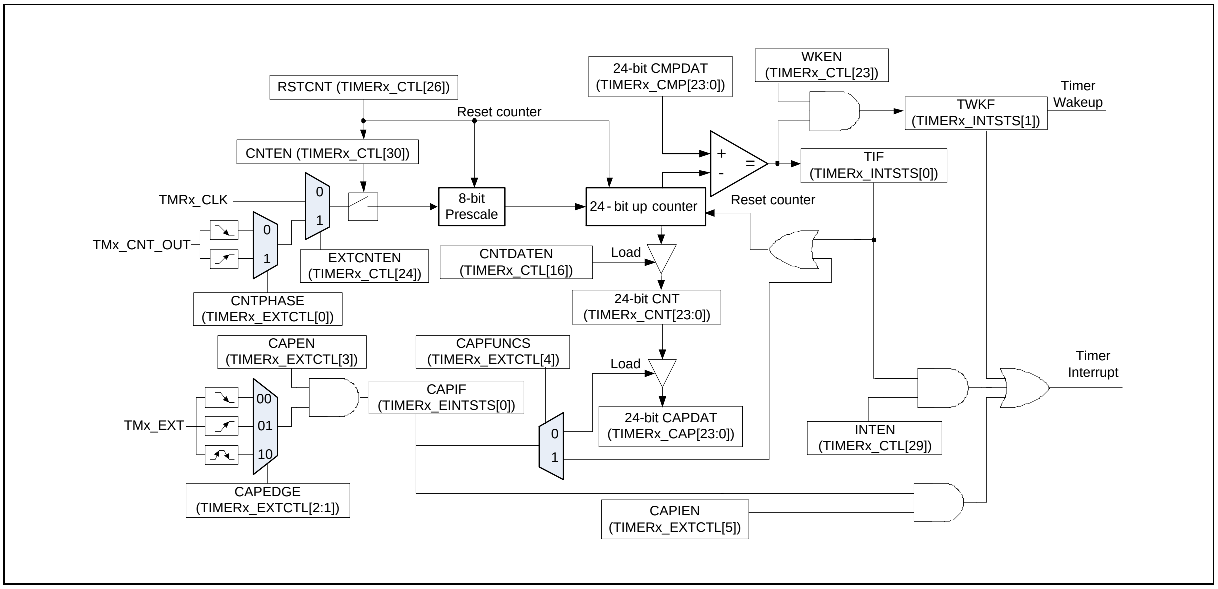


Figure 6.5-1 Timer Controller Block Diagram

### 6.5.4 Functional Description

The 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.

#### *6.5.4.1 Timer Interrupt Flag*

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

#### *6.5.4.2 One–shot Mode*

If the timer controller is configured in One-shot mode (TIMERx_CTL[28:27] is 00) and CNTEN (TIMERx_CTL[30]) bit is set, the timer counter starts up counting. Once the TIMERx_CNT value reaches TIMERx_CMP value, the TIF flag will be set to 1, TIMERx_CNT value and CNTEN bit is cleared by timer controller then timer counting operation stops. In the meantime, if the INTEN (TIMERx_CTL[29]) bit is enabled, the timer interrupt signal is generated and sent to NVIC to inform CPU also.

In this operating mode, when TIF (TIMERx_INTSTS[0])had set to 1, timer counting operation stops and the timer counter value (TIMERx_CNT value) goes back to counting initial value then CNTEN (TIMERx_CTL[30]) is cleared to 0 by timer controller automatically. That is to say, timer operates timer counting and compares with TIMERx_CMP value function only one time after programming the timer compare register (TIMERx_CMP) value and CNTEN (TIMERx_CTL[30]) is set to 1. Accordingly, this operating mode is called “One-shot mode”.

#### 6.5.4.3 Periodic Mode

If the timer is operated in Period mode (TIMERx_CTL[28:27] is 01) and CNTEN (TIMERx_CTL[30]) is set to 1, the timer counter starts up counting. Once the TIMERx_CNT value reaches CMPDAT value, the TIF (TIMERx_INTSTS[0]) will set to 1. If the INTEN (TIMERx_CTL[29]) is set to 1, and TIF (TIMERx_INTSTS[0]) is 1 then the interrupt signal is generated and sent to NVIC to inform CPU for indicating that the timer counting overflow happens. If the INTEN (TIMERx_CTL[29]) is set to 0, no interrupt signal is generated.

In this operating mode, once the TIMERx_CNT value reaches TIMERx_CMP value, TIF (TIMERx_INTSTS[0]) will set to 1, the timer counter value (TIMERx_CNT value) goes back to counting initial value and CNTEN (TIMERx_CTL[30] timer enable bit) is kept at 1 (counting enable continuously) and timer counter operates up counting again. If TIF (TIMERx_INTSTS[0]) is cleared by software, once the TIMERx_CNT value reaches TIMERx_CMP value again, TIF (TIMERx_INTSTS[0]) will set to 1 also. That is to say, timer operates timer counting and compares with TIMERx_CMP value function periodically. The timer counting operation does not stop until the CNTEN (TIMERx_CTL[30]) is set to 0. The interrupt signal is also generated periodically. Thus, this operating mode is called “Periodic mode”.

#### 6.5.4.4 Toggle-Output Mode

If the timer is operated in Toggle-out mode (TIMERx_CTL[28:27] is 10) and CNTEN (TIMERx_CTL[30]) is set to 1, the timer counter starts up counting. Once the TIMERx_CNT value reaches TIMERx_CMP value, the TIF (TIMERx_INTSTS[0]) will set to 1. If the INTEN (TIMERx_CTL[29]) is set to 1, and TIF (TIMERx_INTSTS[0]) is 1 then the interrupt signal is generated and sent to NVIC to inform CPU for indicating that the timer counting overflow happens. If the INTEN (TIMERx_CTL[29]) is set to 0, no interrupt signal is generated.

In this operating mode, once the timer TIMERx_CNT value reaches TIMERx_CMP value, TIF (TIMERx_INTSTS[0]) will set to 1, toggle-out signal (TM0_CNT_OUT~TM3_CNT_OUT pin) is set to 1, the TIMERx_CNT value goes back to counting initial value and CNTEN (TIMERx_CTL[30]) is kept at 1 and timer counter operates up counting again. If TIF (TIMERx_INTSTS[0]) is cleared by software, once the TIMERx_CNT value reaches TIMERx_CMP value again, TIF (TIMERx_INTSTS[0]) will set to 1 also and toggle-out signal (TM0_CNT_OUT~TM3_CNT_OUT pin) is set to 0. The timer counting operation does not stop until the CNTEN (TIMERx_CTL[30]) is set to 0. Thus, the toggle-out signal (TM0_CNT_OUT~TM3_CNT_OUT pin) is changing back and forth with 50% duty cycle. Thus, this operating mode is called Toggle-out mode.

#### 6.5.4.5 Continuous Counting Mode

If the timer is operated in Continuous Counting mode (TIMERx_CTL[28:27] is 11) and CNTEN (TIMERx_CTL[30]) is set to 1, the timer counter starts up counting. Once the TIMERx_CNT value reaches TIMERx_CMP value, the TIF (TIMERx_INTSTS[0]) will set to 1. If the INTEN (TIMERx_CTL[29]) is set to 1, and TIF (TIMERx_INTSTS[0]) is 1 then the interrupt signal is generated and sent to NVIC to inform CPU for indicating that the timer counting overflow happens. If the INTEN (TIMERx_CTL[29]) is set to 0, no interrupt signal is generated.

In this operating mode, once the TIMERx_CNT value reaches TIMERx_CMP value, TIF (TIMERx_INTSTS[0]) will set to 1 and CNTEN (TIMERx_CTL[30]) is kept at 1 and timer counter continuous counting without reload the timer counter value (TIMERx_CNT value) to counting initial value. User can change different timer compare register (TIMERx_CMP) value immediately without disabling timer counter and restarting timer counter counting.

For example, the TIMERx_CMP value is set as 80, first. Once the TIMERx_CNT value reaches to 80, TIF will set to 1 and CNTEN (TIMERx_CTL[30]) is kept at 1 and TIMERx_CNT value will not goes back to 0, it continues to count 81, 82, 83,⋯ to $2^{24}$ -1, 0, 1, 2, 3, ⋯ to $2^{24}$ -1 again and again. Next, if user programs TIMERx_CMP value as 200 and the TIF (TIMERx_INTSTS[0]) is cleared to 0, then TIF (TIMERx_INTSTS[0]) will set to 1 again when TIMERx_CNT value reaches to 200. At last, user programs TIMERx_CMP value as 500 and clears TIF (TIMERx_INTSTS[0]) to 0, then TIF (TIMERx_INTSTS[0]) will set to 1 again when TIMERx_CNT value reaches to 500. In this mode, the TIMERx_CNT value is keeping up counting always even if TIF (TIMERx_INTSTS[0]) is 1. Thus, this operation mode is called “Continuous Counting mode”.

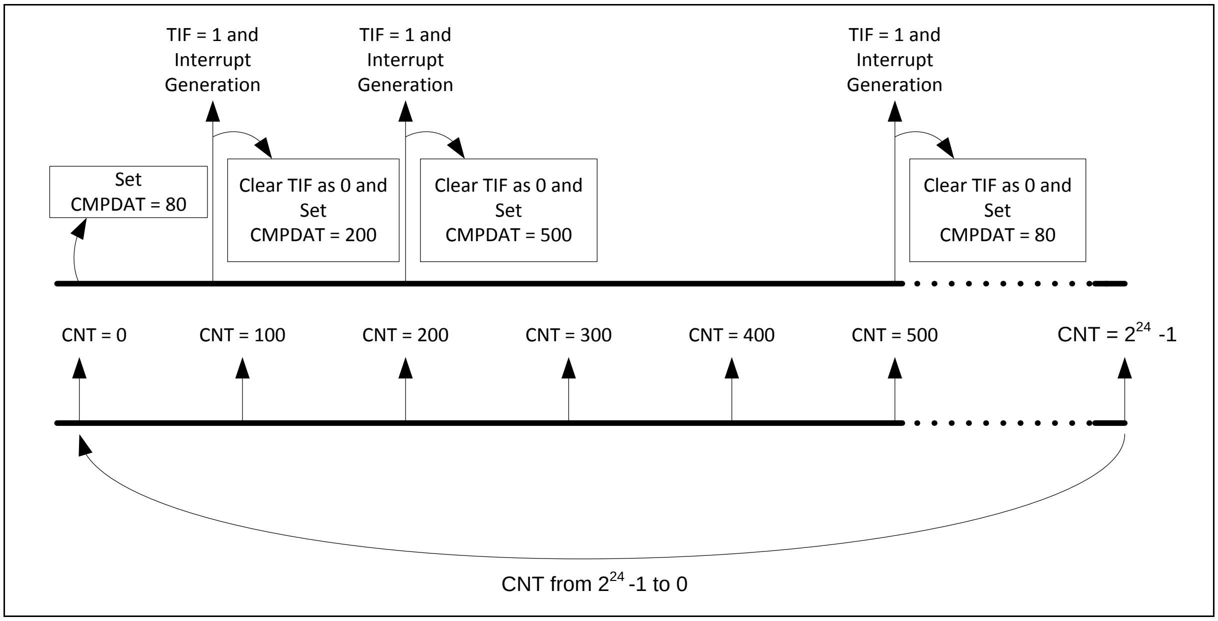

TIF = 1 and Interrupt Generation
TIF = 1 and Interrupt Generation
TIF = 1 and Interrupt Generation
Set CMPDAT = 80
Clear TIF as 0 and Set CMPDAT = 200
Clear TIF as 0 and Set CMPDAT = 500
Clear TIF as 0 and Set CMPDAT = 80
CNT = 0
CNT = 100
CNT = 200
CNT = 300
CNT = 400
CNT = 500
CNT = $2^{24}$ -1
CNT from $2^{24}$ -1 to 0

Figure 6.5-2 Continuous Counting Mode

### 6.5.4.6 Event Counting Mode

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

Software can enable or disable TMx_CNT_OUT pin de-bounce circuit by CNTDBEN (TIMERx_EXTCTL[7]) bit. The input event frequency should be less than 1/3 HXT if TMx_CNT_OUT (x= 0~3) pin de-bounce disabled or less than 1/8 HXT if TMx_CNT_OUT (x= 0~3) pin de-bounce enabled to assure the returned TIMERx_CNT value is incorrect, and software can also select edge detection phase of TMx_CNT_OUT pin by CNTPHASE (TIMERx_EXTCTL[0]) bit.

In Event Counting mode, the timer counting operation mode can be selected as one-shot, periodic and continuous counting mode to counts the TIMERx_CNT value by input event from TMx_CNT_OUT (x= 0~3) pin.

### 6.5.4.7 External Capture Mode

The external capture function is used to capture TIMERx_CNT value to TIMERx_CAP value while edge transition detected on TMx_EXT (x= 0~3) pin. In this mode, CAPFUNCS (TIMERx_EXTCTL[4]) bit should be as 0 for select TMx_EXT transition is using as the external capture function and the timer peripheral clock source should be set as HXT.

Software can enable or disable TMx_EXT pin de-bounce circuit by CAPDBEN (TIMERx_EXTCTL[6]) bit. The transition frequency of TMx_EXT pin should be less than 1/3 HXT if TMx_EXT pin de-bounce disabled or less than 1/8 HXT if TMx_EXT pin de-bounce enabled to assure the capture function can be work normally, and software can also select edge transition detection of TMx_EXT pin by CAPEDGE (TIMERx_EXTCTL[2:1]) bits.

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

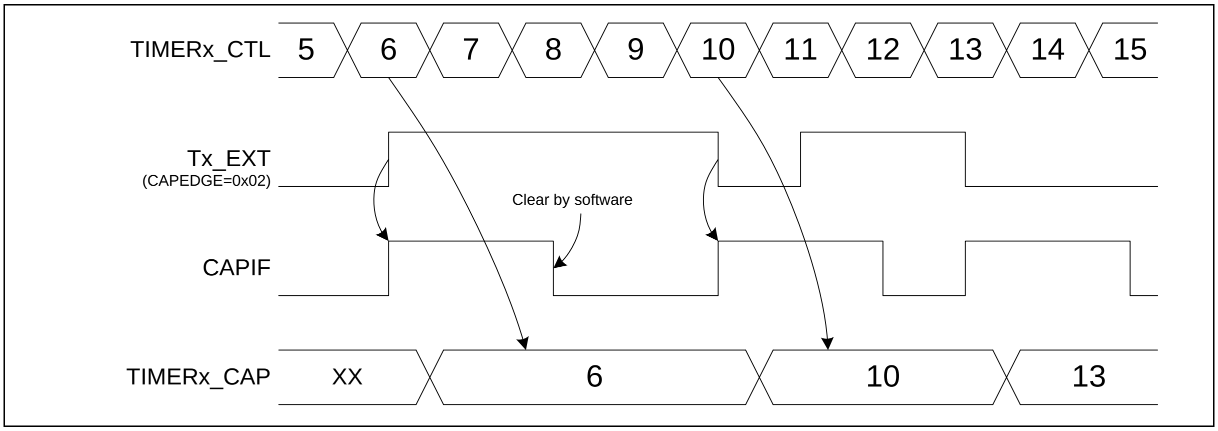


Figure 6.5-3 External Capture Mode

### 6.5.4.8 External Reset Counter Mode

The timer controller also provides reset counter function to reset the CNT (TIMERx_CNT[23:0]) value while edge transition detected on TX_EXT (x= 0~3). In this mode, most of the settings are the same as Event Capture mode except CAPFUNCS (TIMERx_EXTCTL[4]) which should be as 1 for selecting the TX_EXT transition used to trigger reset counter value.

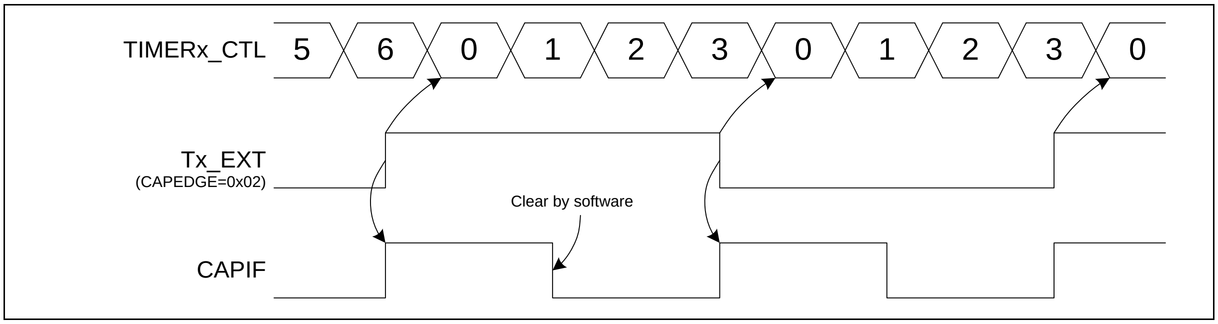


Figure 6.5-4 External Reset Counter Mode

### 6.5.5 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **TIMER Base Address:**<br>**TIMER_BA01 = 0x400E_A000**<br>**TIMER_BA23 = 0x400E_B000** | | | | |
| **TIMER0_CTL** | TIMER_BA01+0x00 | R/W | Timer0 Control and Status Register | 0x0000_0005 |
| **TIMER0_CMP** | TIMER_BA01+0x04 | R/W | Timer0 Compare Register | 0x0000_0000 |
| **TIMER0_INTSTS** | TIMER_BA01+0x08 | R/W | Timer0 Interrupt Status Register | 0x0000_0000 |
| **TIMER0_CNT** | TIMER_BA01+0x0C | R | Timer0 Data Register | 0x0000_0000 |
| **TIMER0_CAP** | TIMER_BA01+0x10 | R | Timer0 Capture Data Register | 0x0000_0000 |
| **TIMER0_EXTCTL** | TIMER_BA01+0x14 | R/W | Timer0 External Control Register | 0x0000_0000 |
| **TIMER0_EINTSTS** | TIMER_BA01+0x18 | R/W | Timer0 External Interrupt Status Register | 0x0000_0000 |
| **TIMER1_CTL** | TIMER_BA01+0x20 | R/W | Timer1 Control and Status Register | 0x0000_0005 |
| **TIMER1_CMP** | TIMER_BA01+0x24 | R/W | Timer1 Compare Register | 0x0000_0000 |
| **TIMER1_INTSTS** | TIMER_BA01+0x28 | R/W | Timer1 Interrupt Status Register | 0x0000_0000 |
| **TIMER1_CNT** | TIMER_BA01+0x2C | R | Timer1 Data Register | 0x0000_0000 |
| **TIMER1_CAP** | TIMER_BA01+0x30 | R | Timer1 Capture Data Register | 0x0000_0000 |
| **TIMER1_EXTCTL** | TIMER_BA01+0x34 | R/W | Timer1 External Control Register | 0x0000_0000 |
| **TIMER1_EINTSTS** | TIMER_BA01+0x38 | R/W | Timer1 External Interrupt Status Register | 0x0000_0000 |
| **TIMER2_CTL** | TIMER_BA23+0x00 | R/W | Timer2 Control and Status Register | 0x0000_0005 |
| **TIMER2_CMP** | TIMER_BA23+0x04 | R/W | Timer2 Compare Register | 0x0000_0000 |
| **TIMER2_INTSTS** | TIMER_BA23+0x08 | R/W | Timer2 Interrupt Status Register | 0x0000_0000 |
| **TIMER2_CNT** | TIMER_BA23+0x0C | R | Timer2 Data Register | 0x0000_0000 |
| **TIMER2_CAP** | TIMER_BA23+0x10 | R | Timer2 Capture Data Register | 0x0000_0000 |
| **TIMER2_EXTCTL** | TIMER_BA23+0x14 | R/W | Timer2 External Control Register | 0x0000_0000 |
| **TIMER2_EINTSTS** | TIMER_BA23+0x18 | R/W | Timer2 External Interrupt Status Register | 0x0000_0000 |
| **TIMER3_CTL** | TIMER_BA23+0x20 | R/W | Timer3 Control and Status Register | 0x0000_0005 |
| **TIMER3_CMP** | TIMER_BA23+0x24 | R/W | Timer3 Compare Register | 0x0000_0000 |
| **TIMER3_INTSTS** | TIMER_BA23+0x28 | R/W | Timer3 Interrupt Status Register | 0x0000_0000 |
| **TIMER3_CNT** | TIMER_BA23+0x2C | R | Timer3 Data Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **TIMER3_CAP** | TIMER_BA23+0x30 | R | Timer3 Capture Data Register | 0x0000_0000 |
| **TIMER3_EXTCTL** | TIMER_BA23+0x34 | R/W | Timer3 External Control Register | 0x0000_0000 |
| **TIMER3_EINTSTS** | TIMER_BA23+0x38 | R/W | Timer3 External Interrupt Status Register | 0x0000_0000 |

### 6.5.6 Register Description

#### Timer Control and Status Register (TIMERx_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **TIMER0_CTL** | TIMER_BA01+0x00 | R/W | Timer0 Control and Status Register | 0x0000_0005 |
| **TIMER1_CTL** | TIMER_BA01+0x20 | R/W | Timer1 Control and Status Register | 0x0000_0005 |
| **TIMER2_CTL** | TIMER_BA23+0x00 | R/W | Timer2 Control and Status Register | 0x0000_0005 |
| **TIMER3_CTL** | TIMER_BA23+0x20 | R/W | Timer3 Control and Status Register | 0x0000_0005 |

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

| Bits | Description | |
|---|---|---|
| [31] | **ICEDEBUG** | **ICE Debug Mode Acknowledge Disable Control**<br>0 = ICE debug mode acknowledgement effects TIMER counting.<br>Timer counter will be held while CPU is held by ICE.<br>1 = ICE debug mode acknowledgement Disabled.<br>Timer counter will keep going no matter CPU is held by ICE or not. |
| [30] | **CNTEN** | **Timer Enable Control**<br>0 = Stops/Suspends counting.<br>1 = Starts counting.<br>**Note1:** In stop status, and then set CNTEN to 1 will enable the 24-bit up counter to keep counting from the last stop counting value.<br>**Note2:** This bit is auto-cleared by hardware in One-shot mode (TIMERx_CTL [28:27] = 00) when the timer interrupt flag TIF (TIMERx_INTSTS[0]) is generated. |
| [29] | **INTEN** | **Interrupt Enable Control**<br>0 = Timer Interrupt Disabled.<br>1 = Timer Interrupt Enabled.<br>**Note:** If this bit is enabled, when the timer interrupt flag TIF is set to 1, the timer interrupt signal is generated and inform to CPU. |
| [28:27] | **OPMODE** | **Timer Operation Mode**<br>00 = Timer controller is operated in One-shot mode. |

| | | |
|---|---|---|
| | | 01 = Timer controller is operated in Periodic mode.<br>10 = Timer controller is operated in Toggle-output mode.<br>11 = Timer controller is operated in Continuous Counting mode. |
| [26] | **RSTCNT** | **Timer Reset Control**<br>Setting this bit will reset the 24-bit up counter value (TIMERx_CNT) and also force CNTEN (TIMERx_CTL[30]) to 0 if ACTSTS (TIMERx_CTL[25]) is 1.<br>0 = No effect.<br>1 = Reset 8-bit PSC counter, 24-bit up counter value and CNTEN bit. |
| [25] | **ACTSTS** | **Timer Active Status Control (Read Only)**<br>This bit indicates the 24-bit up counter status.<br>0 = 24-bit up counter is not active.<br>1 = 24-bit up counter is active. |
| [24] | **EXTCNTEN** | **Counter Mode Enable Control**<br>This bit is for external counting pin function to be enabled. When timer is used as an event counter, this bit should be set to 1 and HXT selected as timer clock source.<br>0 = External counter mode Disabled.<br>1 = External counter mode Enabled. |
| [23] | **WKEN** | **Wake-up Enable Control**<br>If this bit is set to 1, while timer interrupt flag TIF (TIMERx_INTSTS[0]) is 1 and INTEN (TIMERx_CTL[29]) is enabled, the timer interrupt signal will generate a wake-up trigger event to CPU.<br>0 = Wake-up trigger event Disabled if timer interrupt signal generated.<br>1 = Wake-up trigger event Enabled if timer interrupt signal generated. |
| [22:17] | **Reserved** | Reserved. |
| [16] | **CNTDATEN** | **Data Load Enable Control**<br>When this bit is set, timer counter value (TIMERx_CNT) will be updated continuously to monitor internal 24-bit up counter value as the counter is counting.<br>0 = Timer Data Register update Disabled.<br>1 = Timer Data Register update Enabled while timer counter is active. |
| [15:8] | **Reserved** | Reserved. |
| [7:0] | **PSC** | **PSC Counter**<br>Timer input clock source is divided by (PSC+1) before it is fed to the timer up counter. If this field is 0 (PSC = 0), then there is no scaling. |

### Timer Compare Register (TIMERx_CMP)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **TIMER0_CMP** | TIMER_BA01+0x04 | R/W | Timer0 Compare Register | 0x0000_0000 |
| **TIMER1_CMP** | TIMER_BA01+0x24 | R/W | Timer1 Compare Register | 0x0000_0000 |
| **TIMER2_CMP** | TIMER_BA23+0x04 | R/W | Timer2 Compare Register | 0x0000_0000 |
| **TIMER3_CMP** | TIMER_BA23+0x24 | R/W | Timer3 Compare Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **CMPDAT** | **Timer Compared Value**<br>CMPDAT is a 24-bit compared value register. When the internal 24-bit up counter value is equal to CMPDAT value, the TIF (TIMERx_INTSTS[0] timer interrupt flag) will set to 1.<br>Time-out period = (Period of timer clock input) * (8-bit PSC +1) * (24-bit CMPDAT).<br>**Note1:** Never write 0x0 or 0x1 in the CMPDAT field, or the timer will run into unknown state.<br>**Note2:** When timer is operating in Continuous Counting mode, the 24-bit up counter will keep counting continuously even if software writes a new value into CMPDAT field. But if timer is operating at other modes, the 24-bit up counter will restart counting and using the newest CMPDAT value to be the timer compared value if software writes a new value to the CMPDAT field. |

### Timer Interrupt Status Register (TIMERx_INTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **TIMER0_INTSTS** | TIMER_BA01+0x08 | R/W | Timer0 Interrupt Status Register | 0x0000_0000 |
| **TIMER1_INTSTS** | TIMER_BA01+0x28 | R/W | Timer1 Interrupt Status Register | 0x0000_0000 |
| **TIMER2_INTSTS** | TIMER_BA23+0x08 | R/W | Timer2 Interrupt Status Register | 0x0000_0000 |
| **TIMER3_INTSTS** | TIMER_BA23+0x28 | R/W | Timer3 Interrupt Status Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1] | **TWKF** | **Timer Wake-up Flag**<br>This bit indicates the interrupt wake-up flag status of timer.<br>0 = Timer does not cause CPU wake-up.<br>1 = CPU wake-up from Power-down mode and Deep Power-down mode, if timer time-out interrupt signal generated.<br>**Note:** This bit is cleared by writing 1 to it. |
| [0] | **TIF** | **Timer Interrupt Flag**<br>This bit indicates the interrupt flag status of Timer while TIMERx_CNT value reaches to CMPDAT value.<br>0 = No effect.<br>1 = TIMERx_CNT value matches the CMPDAT value.<br>**Note:** This bit is cleared by writing 1 to it. |

### Timer Data Register (TIMERx_CNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **TIMER0_CNT** | TIMER_BA01+0x0C | R | Timer0 Data Register | 0x0000_0000 |
| **TIMER1_CNT** | TIMER_BA01+0x2C | R | Timer1 Data Register | 0x0000_0000 |
| **TIMER2_CNT** | TIMER_BA23+0x0C | R | Timer2 Data Register | 0x0000_0000 |
| **TIMER3_CNT** | TIMER_BA23+0x2C | R | Timer3 Data Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **CNT** | **Timer Data Register**<br>1. EXTCNTEN (TIMERx_CTL[24]) = 0 : CNT is 24- bit counter value.<br>User can read TIMERx_CNT for getting current 24- bit counter value if TIMERx_CTL[24] is set to 0<br>2. EXTCNTEN (TIMERx_CTL[24]) = 1 : CNT is 24- bit event counter value.<br>**Note:** User can read CNT for getting the current 24- bit event counter value if TIMERx_CTL[24] is 1 |

### Timer Capture Data Register (TIMERx_CAP)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **TIMER0_CAP** | TIMER_BA01+0x10 | R | Timer0 Capture Data Register | 0x0000_0000 |
| **TIMER1_CAP** | TIMER_BA01+0x30 | R | Timer1 Capture Data Register | 0x0000_0000 |
| **TIMER2_CAP** | TIMER_BA23+0x10 | R | Timer2 Capture Data Register | 0x0000_0000 |
| **TIMER3_CAP** | TIMER_BA23+0x30 | R | Timer3 Capture Data Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **CAPDAT** | **Timer Capture Data Register**<br>When the CAPEN (TIMERx_EXTCTL[3]) bit is set, CAPFUNCS (TIMERx_EXTCTL[4]) bit is 0, and a transition on TMx_EXT pin matched the CAPEDGE (TIMERx_EXTCTL[2:1]) setting, CAPIF (TIMERx_EINTSTS[0]) will be set to 1 and the current timer counter value (TIMERx_CNT value) will be auto-loaded into the CAPDAT field. |

### Timer External Control Register (TIMERx_EXTCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **TIMER0_EXTCTL** | TIMER_BA01+0x14 | R/W | Timer0 External Control Register | 0x0000_0000 |
| **TIMER1_EXTCTL** | TIMER_BA01+0x34 | R/W | Timer1 External Control Register | 0x0000_0000 |
| **TIMER2_EXTCTL** | TIMER_BA23+0x14 | R/W | Timer2 External Control Register | 0x0000_0000 |
| **TIMER3_EXTCTL** | TIMER_BA23+0x34 | R/W | Timer3 External Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7] | **CNTDBEN** | **Timer Counter Pin De-bounce Enable Control**<br>0 = TMx_CNT_OUT (x= 0~3) pin de-bounce Disabled.<br>1 = TMx_CNT_OUT (x= 0~3) pin de-bounce Enabled.<br>**Note:** If this bit is enabled, the edge detection of TMx_CNT_OUT pin is detected with de-bounce circuit. |
| [6] | **CAPDBEN** | **Timer External Capture Pin De-bounce Enable Control**<br>0 = TMx_EXT (x= 0~3) pin de-bounce Disabled.<br>1 = TMx_EXT (x= 0~3) pin de-bounce Enabled.<br>**Note:** If this bit is enabled, the edge detection of TMx_EXT pin is detected with de-bounce circuit. |
| [5] | **CAPIEN** | **Timer External Interrupt Enable Control**<br>0 = TMx_EXT (x= 0~3) pin detection Interrupt Disabled.<br>1 = TMx_EXT (x= 0~3) pin detection Interrupt Enabled.<br>**Note:** CAPIEN is used to enable timer external interrupt. If CAPIEN enabled, timer will rise an interrupt when CAPIF = 1.<br>For example, while CAPIEN = 1, CAPEN = 1, and CAPEDGE = 00, a 1 to 0 transition on the TMx_EXT pin will cause the CAPIF(TIMERx_EINTSTS[0]) interrupt flag to be set then the interrupt signal is generated and sent to NVIC to inform CPU. |
| [4] | **CAPFUNCS** | **Timer External Reset Counter / Capture Mode Select**<br>0 = Transition on TMx_EXT (x= 0~3) pin is using to save the 24-bit timer counter value.<br>(TIMERx_CNT value) to timer capture value (TIMERx_CAP value) if CAPIF |

| | | |
|---|---|---|
| | | (TIMERx_EINTSTS[0]) is set to 1<br>1 = Transition on TMx_EXT (x= 0~3) pin is using to reset the 24-bit timer counter value. |
| [3] | **CAPEN** | **Timer External Pin Enable Control**<br>This bit enables the CAPFUNCS (TIMERx_EXTCTL[4]) function on the TMx_EXT pin.<br>0 = CAPFUNCS function of TMx_EXT (x= 0~3) pin will be ignored.<br>1 = CAPFUNCS function of TMx_EXT (x= 0~3) pin is active. |
| [2:1] | **CAPEDGE** | **Timer External Pin Edge Detect**<br>00 = A 1 to 0 transition on TMx_EXT (x= 0~3) pin will be detected.<br>01 = A 0 to 1 transition on TMx_EXT (x= 0~3) pin will be detected.<br>10 = Either 1 to 0 or 0 to 1 transition on TMx_EXT (x= 0~3) pin will be detected.<br>11 = Reserved. |
| [0] | **CNTPHASE** | **Timer External Count Phase**<br>This bit indicates the detection phase of external counting pin.<br>0 = A falling edge of external counting pin will be counted.<br>1 = A rising edge of external counting pin will be counted. |

### Timer External Interrupt Status Register (TIMERx_EINTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **TIMER0_EINTSTS** | TIMER_BA01+0x18 | R/W | Timer0 External Interrupt Status Register | 0x0000_0000 |
| **TIMER1_EINTSTS** | TIMER_BA01+0x38 | R/W | Timer1 External Interrupt Status Register | 0x0000_0000 |
| **TIMER2_EINTSTS** | TIMER_BA23+0x18 | R/W | Timer2 External Interrupt Status Register | 0x0000_0000 |
| **TIMER3_EINTSTS** | TIMER_BA23+0x38 | R/W | Timer3 External Interrupt Status Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **CAPIF** | **Timer External Interrupt Flag**<br>This bit indicates the timer external interrupt flag status.<br>When CAPEN (TIMERx_EXTCTL[3]) bit is set, CAPFUNCS (TIMERx_EXTCTL[4]) bit is 0, and a transition on TMx_EXT (x= 0~3) pin matched the CAPEDGE (TIMERx_EXTCTL[2:1]) setting, this bit will set to 1 by hardware.<br>0 = TMx_EXT (x= 0~3) pin interrupt did not occur.<br>1 = TMx_EXT (x= 0~3) pin interrupt occurred.<br>**Note:** This bit is cleared by writing 1 to it. |

## 6.6 PWM Generator and Capture Timer (PWM)

### 6.6.1 Overview

The NUC505 series has one PWM generator that can support four channels PWM output or four channels input capture sharing the same pins (PWM_CH0/ PWM_CH1/PWM_CH2/PWM_CH3).

The PWM generator has a 16-bit PWM counter and comparator, and the PWM generator supports two standard PWM output modes: Independent output mode and Complementary output mode with 8-bit Dead-time generator. Each mode can be used as a timer and issues interrupt independently. In addition, It also has an 8-bit prescaler and clock divider with 5 divided frequencies (1, 1/2, 1/4, 1/8, 1/16) to support wide range clock frequency of PWM counter. For PWM output control unit, it supports polarity output function.

The PWM generator also supports input capture function. It supports latch PWM counter value to corresponding register when input channel has a rising transition, falling transition or both transition is happened.

After the capture feature is enabled, the capture always latches PWM-counter to RCAPDATn when input channel has a rising transition and latched PWM-counter to FCAPDATn when input channel has a falling transition. Capture channel 0 interrupt is programmable by setting CRLIEN0 (PWM_CAPCTL01[1]) (Rising latch Interrupt enable) and CFLIEN0 (PWM_CAPCTL01[2]) (Falling latch Interrupt enable) to determine the condition of interrupt occur. Capture channel 1 has the same feature by setting CRLIEN1 (PWM_CAPCTL01[17]) and CFLIEN1 (PWM_CAPCTL01[18]). The capture channel 2 & 3 has the same feature by setting CRLIEN2 (PWM_CAPCTL23[1]),CFLIEN2 (PWM_CAPCTL23[2]) and CRLIEN3 (PWM_CAPCTL23[17]), CFLIEN3 (PWM_CAPCTL23[18]) respectively. Whenever Capture issues Interrupt 0/1/2/3, the PWM counter 0/1/2/3 will be reload at this moment.

There are only four interrupts from PWM. PWM 0 and Capture 0 share the same interrupt; PWM 1 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.

### 6.6.2 Features

#### *6.6.2.1 PWM function features*

- Supports 4 PWM output channels with 16-bit resolution
- Supports 8-bit prescaler and clock divider
- Supports 4 PWM interrupts
- Supports One-shot or Auto-reload PWM counter operation mode
- Supports 8-bit Dead-time

#### *6.6.2.2 Capture function features*

- Supports 4 capture input channels with 16-bit resolution
- Supports rising or falling capture condition
- Supports 4 Capture interrupts

### 6.6.3 Block Diagram

Figure 6.6-1 and Figure 6.6-2 illustrate the architecture of PWM in pair (e.g. PWM-Timer 0/1 are in one pair, PWM-Timer 2/3 are in one pair).

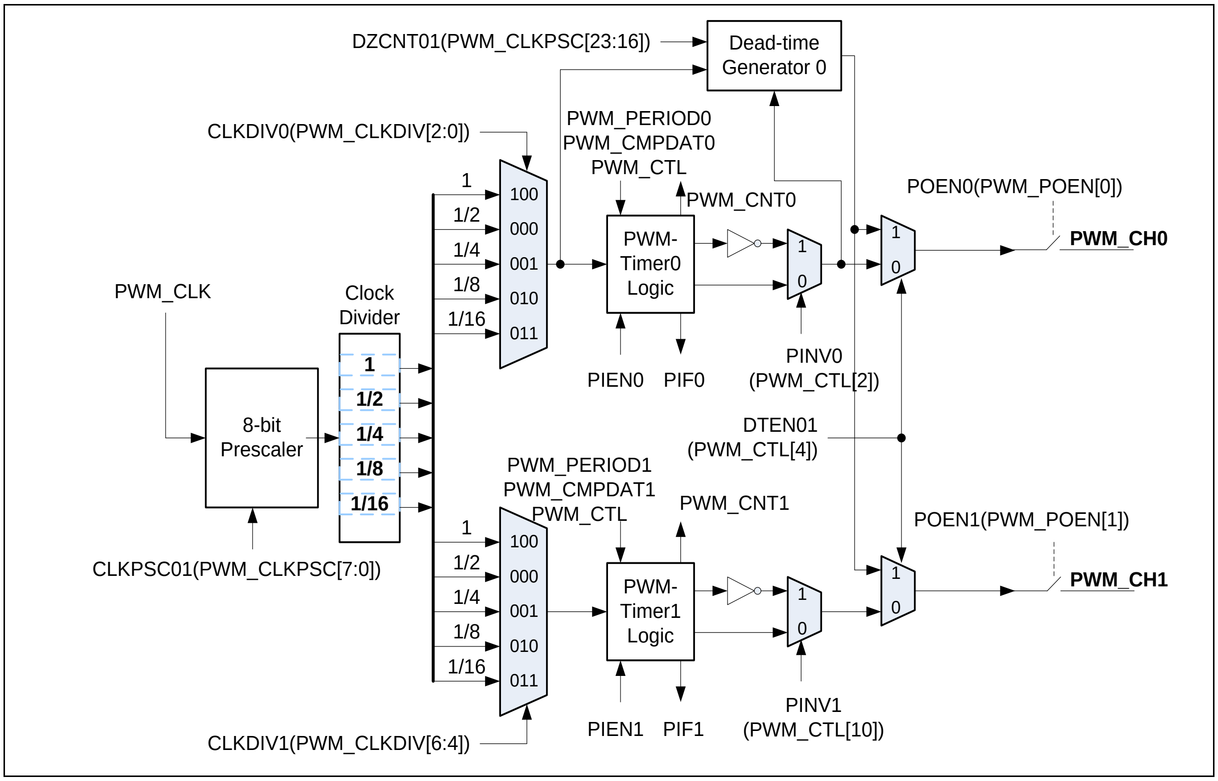


Figure 6.6-1 PWM Generator 0 Architecture Diagram

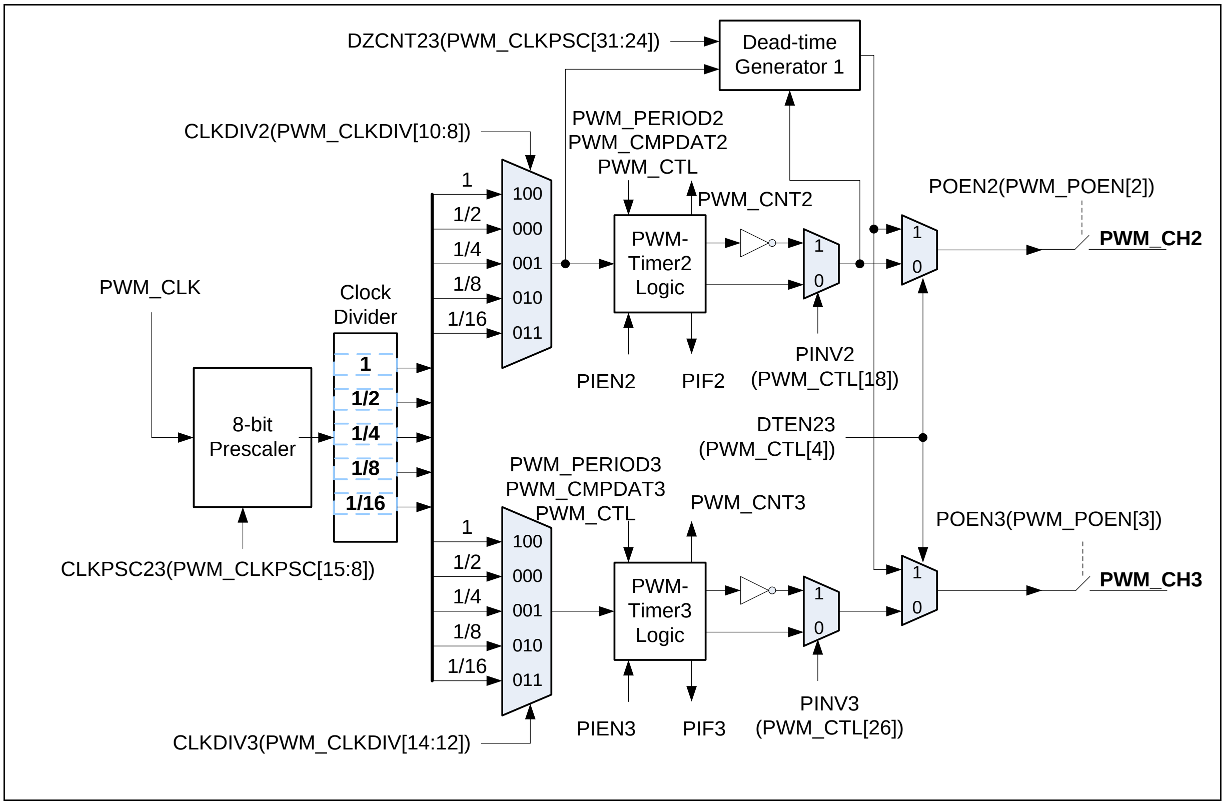


Figure 6.6-2 PWM Generator 1 Architecture Diagram

Each PWM generator has two clock source inputs. Each clock source can be selected from system clock as shown in Figure 6.6-3

To keep PWM output duty cycle when system clock is changed, user can select external Crystal (12 MHz). The maximum PWM_CLK frequency can be the same as PCLK frequency.

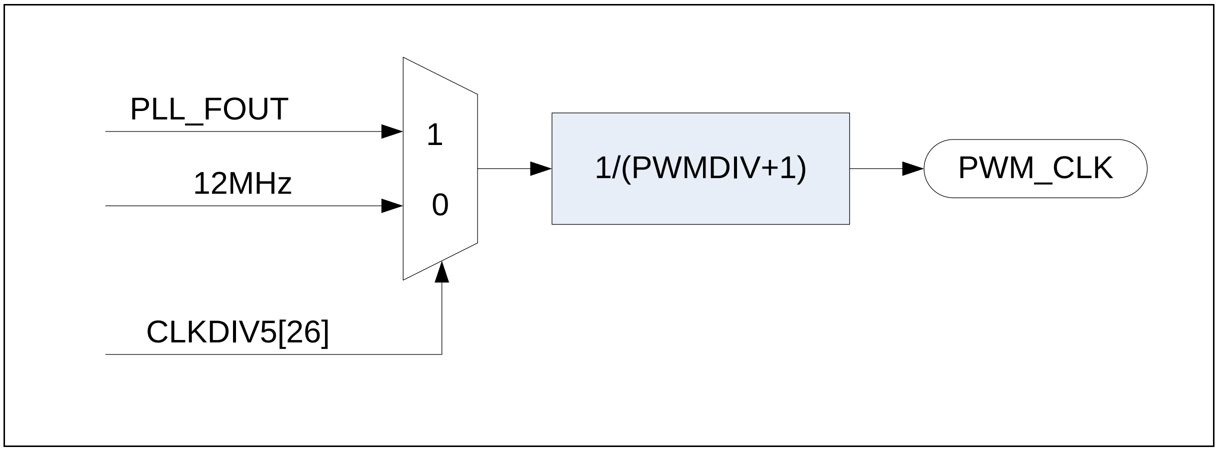


Figure 6.6-3 PWM System Clock Source Control

### 6.6.4 Functional Description

#### *6.6.4.1 Edge-aligned Type (Down-counter)*

In Edge-aligned type, the 16 bits PWM counter is a down counter and starts down-counting from PWM_PERIODn to zero to finish a PWM period. The value of PWM counter will be compared with PWM_CMPDATn at the PWM comparator unit. The PWM comparator unit will output low when the value of PWM counter is larger than PWM_CMPDATn and output high when the value of PWM counter is equal or smaller than PWM_CMPDATn. Base on this operating mechanism, the period and duty of PWM can be controlled by PWM_PERIODn and PWM_CMPDATn. The PWM follows the formula below and the legend of PWM waveform of Edge-aligned type is shown in the following Figure 6.6-4.

- PWM clock frequency = PWM_CLK/[(prescale+1)* (clock divider)] depends on selected PWM channel.
- Duty ratio = (PWM_CMPDATn + 1)/( PWM_PERIODn + 1)
- PWM_CMPDATn >= PWM_PERIODn: PWM output is always high
- PWM_CMPDATn < PWM_PERIODn: PWM low width= (PWM_PERIODn - PWM_CMPDATn) unit; PWM high width = (PWM_CMPDATn + 1) unit
- PWM_CMPDATn = 0: PWM low width = (PWM_PERIODn) unit; PWM high width = 1 unit
- 1 unit = one PWM output clock cycle

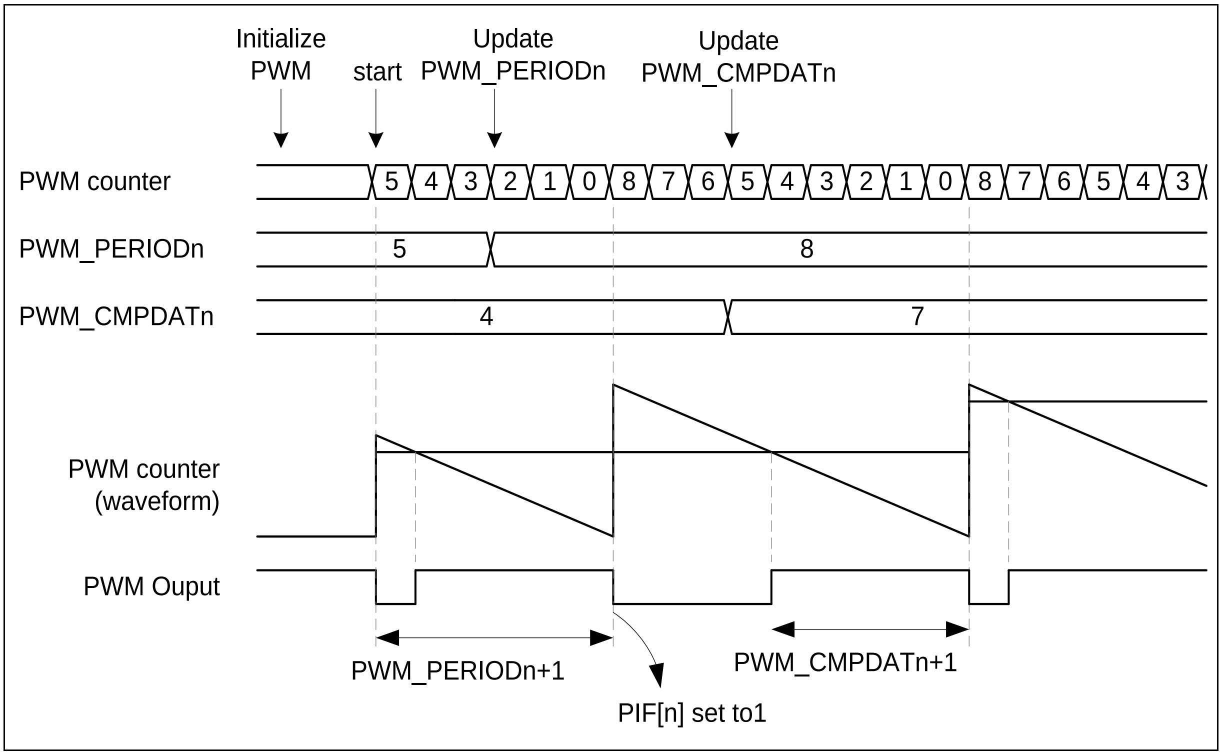


Figure 6.6-4 PWM Waveform of Edge-aligned Type

#### *6.6.4.1 PWM Counter Operation Mode*

The PWM counter supports two operation modes: One-shot mode and Auto-reload mode. PWM counter will operate in One-shot mode if CNTMODE0 (PWM_CTL[3]) bit is set to 0, and operate in Auto-reload mode if CNTMODE0 (PWM_CTL[3]) bit is set to 1. It is recommended to configure PWM counter operation mode before setting PWM_PERIODn, PWM_CMPDATn and enable PWM counter running (set CNTENn bit as 1) because the content of PWM_PERIODn and PWM_CMPDATn will be cleared to zero when the PWM counter operation mode is changed.

In One-shot mode, PWM_CMPDATn and PWM_PERIODn should be written first and then the CNTENn bit should be set to 1 to enable PWM counter to start running. After PWM counter counts down from PWM_PERIODn to zero at Edge-aligned type, PWM_PERIODn and PWM_CMPDATn will be cleared to zero by hardware and PWM counter will be held.

A new value of PWM_CMPDATn and PWM_PERIODn needs to be written through software to set the next one-shot period and duty. When re-starting the next one-shot operation, the PWM_CMPDATn should be written first because PWM counter will auto re-start counting when PWM_PERIODn is written a non-zero value.

In Auto-reload mode, PWM_CMPDATn and PWM_PERIODn should be written first and then the CNTENn bit should be set to 1 to enable PWM counter to start running. The value of PWM_PERIODn will be reloaded to PWM counter when down count reaches zero. If PWM_PERIODn is set to zero, PWM counter will be held.

### *6.6.4.2 PWM Double Buffering*

The PWM has double buffering function for PWM period and duty. When software changes PWM period (by writing PWM_PERIODn) or PWM duty (by writing PWM_CMPDATn), the period and duty of PWM will reload new value at the start of next period without affecting the current timer operation. The double buffing timing waveform is shown below.

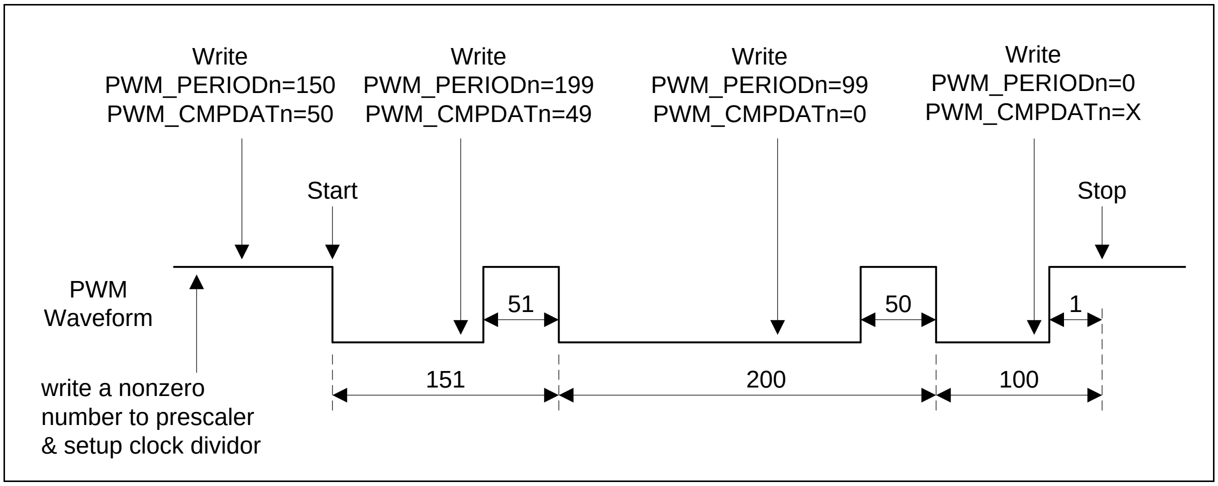


Figure 6.6-5 PWM Double Buffering Timing Waveform

### *6.6.4.3 PWM Output Control Unit*

The PWM output control unit includes a dead-time generator, output mode control, and polarity control.

### *6.6.4.3.1 Dead-time Generator*

The dead-time generator inserts an “off” period called “dead-time” to one of the complementary paired pins which is turned on when another is turned off. The complementary output pair mode has an 8-bit down counter used to produce the dead-time insertion. The complementary outputs are delayed until the timer counts down to zero.

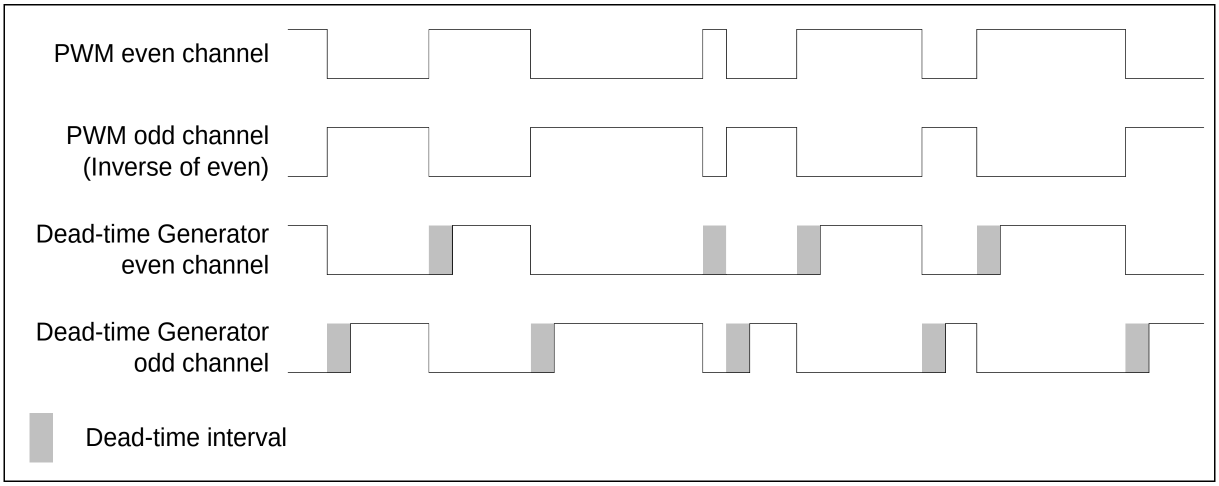


Figure 6.6-6 PWM Paired-output with Dead-time Generation Operation

### *6.6.4.3.2 Polarity Control*

Each PWM port, from PWM_CH0 to PWM_CH3, has an independent polarity control to configure the polarity of the active state of the PWM output. By default, the PWM output is active high. This implies the PWM OFF state is low and ON state is high. This definition is variable through setting the PWM Output Polarity Inverse Enable PINV bit of PWM_CTL, for each individual PWM channel.

### *6.6.4.4 Capture Operation*

The channel of capture input and PWM output share the same pin and PWM counter. The capture function always latches PWM counter to RCAPDATn when input channel has a rising transition and latches PWM counter to FCAPDATn when input channel has a falling transition. Capture interrupt is programmable by setting CRLIE[n] (Rising latch Interrupt enable) and CFLIE[n] (Falling latch Interrupt enable) to determine the condition of interrupt occurrence. Whenever the capture controller issues a capture interrupt, the corresponding PWM counter will be reloaded with PWM_PERIODn at this moment. Note that the corresponding GPIO pins must be configured as capture function (CAPINEN[n] enabled) for the corresponding capture channel.

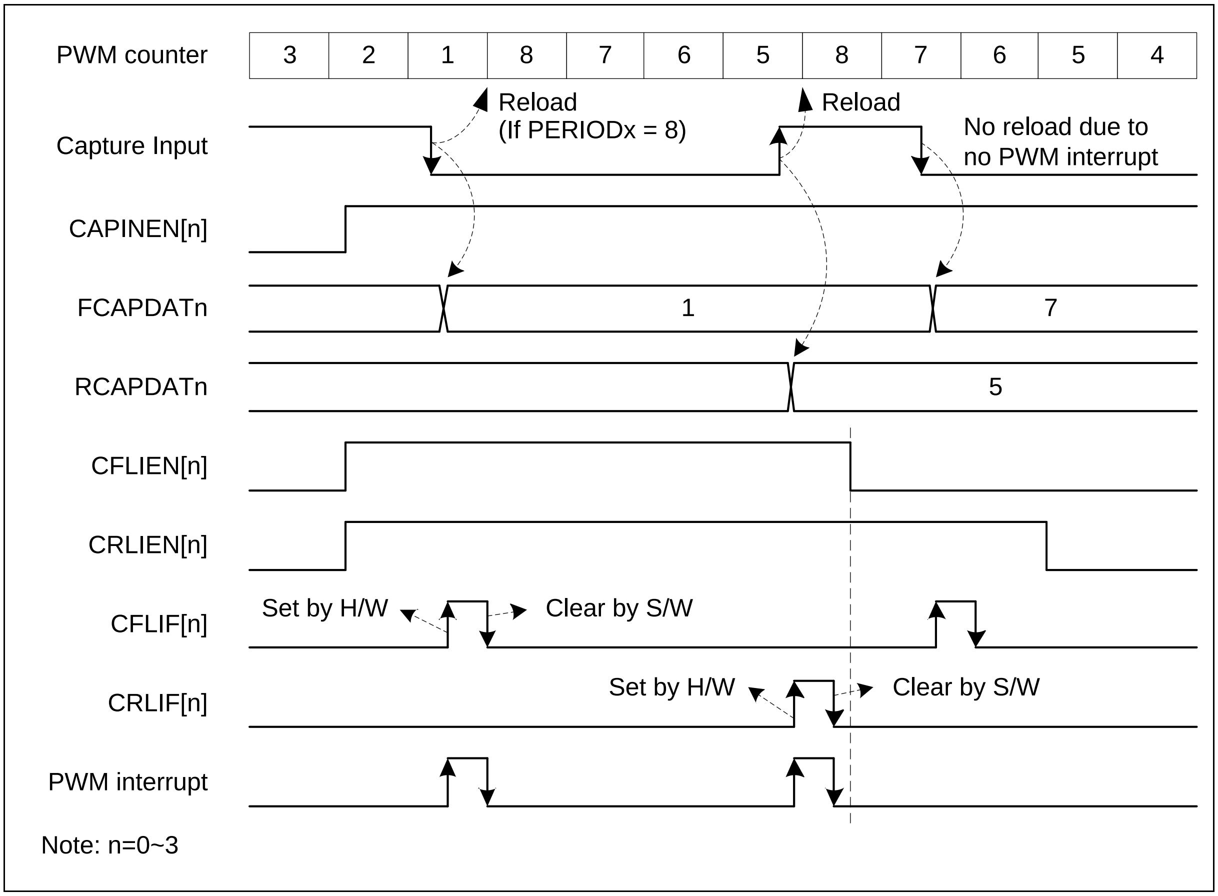


Figure 6.6-7 Capture Operation Timing

In this case, the PERIOD is 8:

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

### 6.6.4.5 PWM Start Procedure

The following procedure is recommended for starting a PWM drive.

1. Set clock source divider select register (PWM_CLKDIV)
2. Set prescaler (PWM_CLKPSC)
3. Set PWM counter Auto-reload/One-shot operation mode, PWM output polarity(PWM_CTL)
4. Set Dead-time generator on/off (PWM_CTL)
5. Set comparator register (PWM_CMPDATn) for setting PWM duty.
6. Set PWM counter register (PWM_PERIODn) for setting PWM period.
7. Set interrupt Enable Control Register (PWM_INTEN) (option)

8. Setup PWM output enable (PWM_POEN)

| | | | | |
|---|---|---|---|---|
| PWM_ POEN[0] enable | -----> | PWM0 output | -----> | PB.10/PC.9 |
| PWM_ POEN[1] enable | -----> | PWM1 output | -----> | PB.11/PC.10 |
| PWM_ POEN[2] enable | -----> | PWM2 output | -----> | PB.12/PC.11 |
| PWM_ POEN[3] enable | -----> | PWM3 output | -----> | PB.13/PC.12 |

9. Set the corresponding GPIO pins as PWM function for the corresponding PWM channel.

   Enable PWM timer start running (PWM_CTL).

### *6.6.4.6 PWM Re-Start Procedure in One-shot mode*

After PWM waveform is generated once in PWM One-shot mode, PWM counter will be stopped automatically. The following procedure is recommended for re-starting PWM at One-shot mode.

1. Set comparator register (PWM_CMPDATn) for setting PWM duty.
2. Set PWM counter register (PWM_PERIODn) for setting PWM period. After setup PWM_PERIODn, PWM wave will be generated

### *6.6.4.7 PWM Stop Procedure*

**Method 1:**

Set 16-bit down counter (PWM_PERIODn) as 0, and monitor PWM_CNTn (current value of 16-bit down-counter). When PWM_CNTn reaches 0, disable the PWM-Timer CNTENn bit. ***(Recommended)***

**Method 2:**

Set 16-bit down counter (PWM_PERIODn) as 0. When interrupt request happened, disable the PWM-Timer CNTENn bit. ***(Recommended)***

**Method 3:**

Disable PWM-Timer directly (CNTENn bit). ***(Not recommended)***

The reason why method 3 is not recommended is that disable (CNTENn bit) 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

### *6.6.4.8 Capture Start Procedure*

1. Set clock source divider select register (PWM_CLKDIV)
2. Set prescaler (PWM_CLKPSC)
3. Set capture input inverter on/off and capture function enable (PWM_CAPCTL)
4. Set PWM counter as Auto-reload mode (PWM_CTL)

5. Set PWM period register (PWM_PERIODn)
6. Enable capture (PWM_CAPCTL)
7. Enable PWM timer start running (Set (CNTENn = 1)

#### *6.6.4.9 PWM interrupt Generator*

There are four independent interrupts for PWM block as shown in Figure 6.6-8

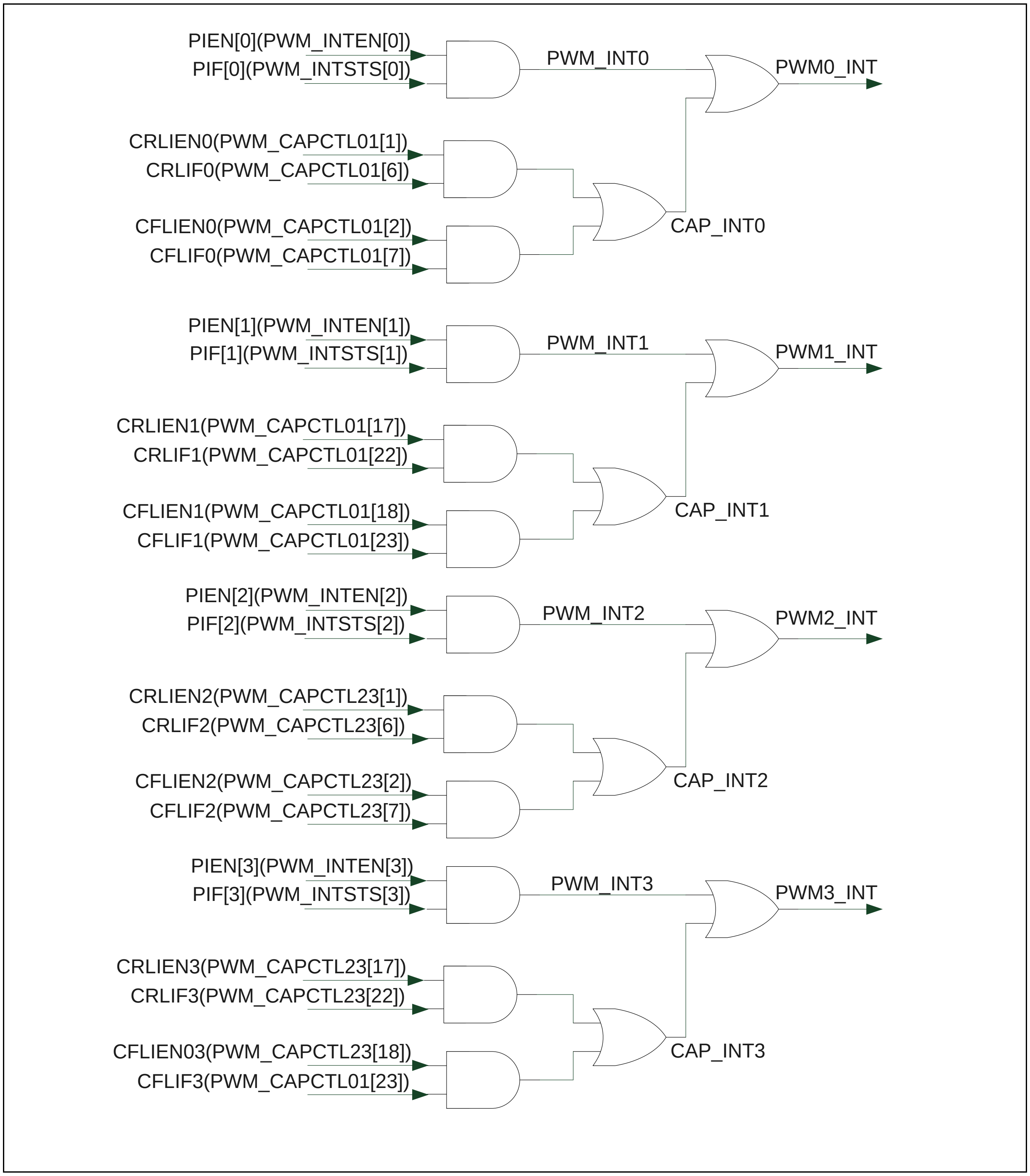


Figure 6.6-8 PWM Interrupt Generator

### 6.6.5 Register Map

**R**: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 0 can be written

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM Base Address:**<br>**PWM_BA = 0x400E_6000** | | | | |
| **PWM_CLKPSC** | PWM_BA+0x00 | R/W | PWM Pre-scale Register | 0x0000_0000 |
| **PWM_CLKDIV** | PWM_BA+0x04 | R/W | PWM Clock Select Register | 0x0000_0000 |
| **PWM_CTL** | PWM_BA+0x08 | R/W | PWM Control Register | 0x0000_0000 |
| **PWM_PERIOD0** | PWM_BA+0x0C | R/W | PWM Period Register 0 | 0x0000_0000 |
| **PWM_CMPDAT0** | PWM_BA+0x10 | R/W | PWM Comparator Register 0 | 0x0000_0000 |
| **PWM_CNT0** | PWM_BA+0x14 | R | PWM Data Register 0 | 0x0000_0000 |
| **PWM_PERIOD1** | PWM_BA+0x18 | R/W | PWM Period Register 1 | 0x0000_0000 |
| **PWM_CMPDAT1** | PWM_BA+0x1C | R/W | PWM Comparator Register 1 | 0x0000_0000 |
| **PWM_CNT1** | PWM_BA+0x20 | R | PWM Data Register 1 | 0x0000_0000 |
| **PWM_PERIOD2** | PWM_BA+0x24 | R/W | PWM Period Register 2 | 0x0000_0000 |
| **PWM_CMPDAT2** | PWM_BA+0x28 | R/W | PWM Comparator Register 2 | 0x0000_0000 |
| **PWM_CNT2** | PWM_BA+0x2C | R | PWM Data Register 2 | 0x0000_0000 |
| **PWM_PERIOD3** | PWM_BA+0x30 | R/W | PWM Period Register 3 | 0x0000_0000 |
| **PWM_CMPDAT3** | PWM_BA+0x34 | R/W | PWM Comparator Register 3 | 0x0000_0000 |
| **PWM_CNT3** | PWM_BA+0x38 | R | PWM Data Register 3 | 0x0000_0000 |
| **PWM_INTEN** | PWM_BA+0x40 | R/W | PWM Interrupt Enable Register | 0x0000_0000 |
| **PWM_INTSTS** | PWM_BA+0x44 | R/W | PWM Interrupt Indication Register | 0x0000_0000 |
| **PWM_CAPCTL01** | PWM_BA+0x50 | R/W | Capture Control Register 0 | 0x0000_0000 |
| **PWM_CAPCTL23** | PWM_BA+0x54 | R/W | Capture Control Register 1 | 0x0000_0000 |
| **PWM_RCAPDAT0** | PWM_BA+0x58 | R/W | Capture Rising Latch Register (Channel 0) | 0x0000_0000 |
| **PWM_FCAPDAT0** | PWM_BA+0x5C | R/W | Capture Falling Latch Register (Channel 0) | 0x0000_0000 |
| **PWM_RCAPDAT1** | PWM_BA+0x60 | R/W | Capture Rising Latch Register (Channel 1) | 0x0000_0000 |
| **PWM_FCAPDAT1** | PWM_BA+0x64 | R/W | Capture Falling Latch Register (Channel 1) | 0x0000_0000 |
| **PWM_RCAPDAT2** | PWM_BA+0x68 | R/W | Capture Rising Latch Register (Channel 2) | 0x0000_0000 |
| **PWM_FCAPDAT2** | PWM_BA+0x6C | R/W | Capture Falling Latch Register (Channel 2) | 0x0000_0000 |
| **PWM_RCAPDAT3** | PWM_BA+0x70 | R/W | Capture Rising Latch Register (Channel 3) | 0x0000_0000 |
| **PWM_FCAPDAT3** | PWM_BA+0x74 | R/W | Capture Falling Latch Register (Channel 3) | 0x0000_0000 |
| **PWM_CAPINEN** | PWM_BA+0x78 | R/W | Capture Input Enable Register | 0x0000_0000 |
| **PWM_POEN** | PWM_BA+0x7C | R/W | PWM Output Enable Register | 0x0000_0000 |

### 6.6.6 Register Description

**PWM Pre-scale Register (PWM_CLKPSC)**

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_CLKPSC** | PWM_BA+0x00 | R/W | PWM Pre-scale Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| DZCNT23 | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| DZCNT01 | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| CLKPSC23 | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| CLKPSC01 | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **DZCNT23** | **Dead-time Interval Register 1**<br>These 8-bit determine Dead-time length.<br>Dead-time = (DTCNT23[7:0]+1) * PWM_CLK period. |
| [23:16] | **DZCNT01** | **Dead-time Interval Register 0**<br>These 8-bit determine Dead-time length.<br>Dead-time = (DTCNT01[7:0]+1) * PWM_CLK period. |
| [15:8] | **CLKPSC23** | **Clock Pre-scale 1 for PWM Counter 2 & 3**<br>Clock input is divided by (CLKPSC23+1) before it is fed to the counter 2 & 3.<br>If CLKPSC23=0, then the pre-scale 1 output clock will be stopped. |
| [7:0] | **CLKPSC01** | **Clock Pre-scale 0 for PWM Counter 0 & 1**<br>Clock input is divided by (CLKPSC01+1) before it is fed to the counter 0 & 1.<br>If CLKPSC01=0, then the pre-scale 0 output clock will be stopped. |

NUC505 SERIES TECHNICAL REFERENCE MANUAL

### PWM Clock Select Register (PWM_CLKDIV)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_CLKDIV** | PWM_BA+0x04 | R/W | PWM Clock Select Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | **CLKDIV3** | | | **Reserved** | **CLKDIV2** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **CLKDIV1** | | | **Reserved** | **CLKDIV0** | | |

| Bits | Description | |
|---|---|---|
| [31:15] | **Reserved** | Reserved. |
| [14:12] | **CLKDIV3** | **PWM Counter 3 Clock Source Selection**<br>Select clock input for timer 3.<br>000 = PWM_CLK/2.<br>001 = PWM_CLK/4.<br>010 = PWM_CLK/8.<br>011 = PWM_CLK/16.<br>100 = PWM_CLK/1. |
| [11] | **Reserved** | Reserved. |
| [10:8] | **CLKDIV2** | **PWM Counter 2 Clock Source Selection**<br>Select clock input for PWM Counter 2.<br>(Table is the same as CLKDIV3) |
| [7] | **Reserved** | Reserved. |
| [6:4] | **CLKDIV1** | **PWM Counter 1 Clock Source Selection**<br>Select clock input for PWM Counter 1.<br>(Table is the same as CLKDIV3) |
| [3] | **Reserved** | Reserved. |
| [2:0] | **CLKDIV0** | **PWM Counter 0 Clock Source Selection**<br>Select clock input for PWM Counter 0.<br>(Table is the same as CLKDIV3) |

### PWM Control Register (PWM_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_CTL** | PWM_BA+0x08 | R/W | PWM Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | **CNTMODE3** | **PINV3** | **Reserved** | **CNTEN3** |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | **CNTMODE2** | **PINV2** | **Reserved** | **CNTEN2** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | **CNTMODE1** | **PINV1** | **Reserved** | **CNTEN1** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | **DTEN23** | **DTEN01** | **CNTMODE0** | **PINV0** | **Reserved** | **CNTEN0** |

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27] | **CNTMODE3** | **PWM Counter 3 Auto-reload Mode/One-shot Mode**<br>0 = One-Shot mode.<br>1 = Auto-Reload mode.<br>**Note:** If there is a rising transition at this bit, it will cause PWM_PERIOD3 and PWM_CMPDAT3 be cleared. |
| [26] | **PINV3** | **PWM Counter 3 Inverter ON/OFF**<br>0 = Inverter OFF.<br>1 = Inverter ON. |
| [25] | **Reserved** | Reserved. |
| [24] | **CNTEN3** | **PWM Counter 3 Enable Control**<br>0 = PWM Counter and clock prescaler stops running.<br>1 = PWM Counter and clock prescaler starts running. |
| [23:20] | **Reserved** | Reserved. |
| [19] | **CNTMODE2** | **PWM Counter 2 Auto-reload Mode/One-shot Mode**<br>0 = One-shot mode.<br>1 = Auto-reload mode.<br>**Note:** If there is a rising transition at this bit, it will cause PWM_PERIOD2 and PWM_CMPDAT2 be cleared. |
| [18] | **PINV2** | **PWM Counter 2 Inverter ON/OFF**<br>0 = Inverter OFF.<br>1 = Inverter ON. |
| [17] | **Reserved** | Reserved. |
| [16] | **CNTEN2** | **PWM Counter 2 Enable Control** |

| | | |
|---|---|---|
| | | 0 = PWM Counter and clock prescaler stops running.<br>1 = PWM Counter and clock prescaler starts running. |
| [15:12] | **Reserved** | Reserved. |
| [11] | **CNTMODE1** | **PWM Counter 1 Auto-reload Mode/One-shot Mode**<br>0 = One-shot mode.<br>1 = Auto-reload mode.<br>**Note:** If there is a rising transition at this bit, it will cause PWM_PERIOD1 and PWM_CMPDAT1 be cleared. |
| [10] | **PINV1** | **PWM Counter 1 Inverter ON/OFF**<br>0 = Inverter OFF.<br>1 = Inverter ON. |
| [9] | **Reserved** | Reserved. |
| [8] | **CNTEN1** | **PWM Counter 1 Enable Control**<br>0 = PWM Counter and clock prescaler stops running.<br>1 = PWM Counter and clock prescaler starts running. |
| [7:6] | **Reserved** | Reserved. |
| [5] | **DTEN23** | **Dead-time 1 Generator Enable Control**<br>0 = Dead-time generator stops running.<br>1 = Dead-time generator starts running. |
| [4] | **DTEN01** | **Dead-time 0 Generator Enable Control**<br>0 = Dead-time generator stops running.<br>1 = Dead-time generator starts running. |
| [3] | **CNTMODE0** | **PWM Counter 0 Auto-reload Mode/One-shot Mode**<br>0 = One-shot mode.<br>1 = Auto-reload mode.<br>If there is a rising transition at this bit, it will cause PWM_PERIOD0 and PWM_CMPDAT0 be cleared. |
| [2] | **PINV0** | **PWM Counter 0 Inverter ON/OFF**<br>0 = Inverter OFF.<br>1 = Inverter ON. |
| [1] | **Reserved** | Reserved. |
| [0] | **CNTEN0** | **PWM Counter 0 Enable Control**<br>0 = PWM Counter and clock prescaler stops running.<br>1 = PWM Counter and clock prescaler starts running. |

### PWM Period Register 3~0 (PWM_PERIOD3~0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_PERIOD0** | PWM_BA+0x0C | R/W | PWM Period Register 0 | 0x0000_0000 |
| **PWM_PERIOD1** | PWM_BA+0x18 | R/W | PWM Period Register 1 | 0x0000_0000 |
| **PWM_PERIOD2** | PWM_BA+0x24 | R/W | PWM Period Register 2 | 0x0000_0000 |
| **PWM_PERIOD3** | PWM_BA+0x30 | R/W | PWM Period Register 3 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **PERIOD** | **PWM Period Register**<br>PERIOD determines the PWM period.<br>PWM frequency = PWM_CLK/[(prescale+1)*(clock divider)*(PERIOD+1)].<br>● Duty ratio = (CMP+1)/(PERIOD+1).<br>● CMP >= PERIOD: PWM output is always high.<br>● CMP < PERIOD: PWM low width = (PERIOD-CMP) unit; PWM high width = (CMP+1) unit.<br>● CMP = 0: PWM low width = (PERIOD) unit; PWM high width = 1 unit.<br>**Note:** Any write to PERIOD will take effect in the next PWM cycle. |

## PWM Comparator Register 3~0 (PWM_CMPDAT3~0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_CMPDAT0** | PWM_BA+0x10 | R/W | PWM Comparator Register 0 | 0x0000_0000 |
| **PWM_CMPDAT1** | PWM_BA+0x1C | R/W | PWM Comparator Register 1 | 0x0000_0000 |
| **PWM_CMPDAT2** | PWM_BA+0x28 | R/W | PWM Comparator Register 2 | 0x0000_0000 |
| **PWM_CMPDAT3** | PWM_BA+0x34 | R/W | PWM Comparator Register 3 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **CMP** | **PWM Compare Register**<br>CMP determines the PWM output duty ratio.<br>PWM frequency = PWM_CLK/[(prescale+1)*(clock divider)*(PERIOD+1)].<br>● Duty ratio = (CMP+1)/(PERIOD+1).<br>● CMP >= PERIOD: PWM output is always high.<br>● CMP < PERIOD: PWM low width = (PERIOD-CMP) unit; PWM high width = (CMP+1) unit.<br>● CMP = 0: PWM low width = (PERIOD) unit; PWM high width = 1 unit.<br>**Note:** Any write to CMP will take effect in the next PWM cycle. |

### PWM Data Register 3~0 (PWM_CNT3~0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_CNT0** | PWM_BA+0x14 | R | PWM Data Register 0 | 0x0000_0000 |
| **PWM_CNT1** | PWM_BA+0x20 | R | PWM Data Register 1 | 0x0000_0000 |
| **PWM_CNT2** | PWM_BA+0x2C | R | PWM Data Register 2 | 0x0000_0000 |
| **PWM_CNT3** | PWM_BA+0x38 | R | PWM Data Register 3 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **CNT** | **PWM Data Register**<br>User can monitor CNT to know current value in 16-bit down counter. |

### PWM Interrupt Enable Register (PWM_INTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_INTEN** | PWM_BA+0x40 | R/W | PWM Interrupt Enable Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:4] | **Reserved** | Reserved. |
| [3:0] | **PIEN** | **PWM Period Interrupt Enable Control**<br>0 = Period interrupt Disabled.<br>1 = Period interrupt Enabled.<br>**Note:** Each bit controls the corresponding PWM channel. |

### PWM Interrupt Indication Register (PWM_INTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_INTSTS** | PWM_BA+0x44 | R/W | PWM Interrupt Indication Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:4] | **Reserved** | Reserved. |
| [3:0] | **PIF** | **PWM Timer Interrupt Flag**<br>0 = Interrupt Flag OFF.<br>1 = Interrupt Flag ON.<br>**Note1:** Each bit controls the corresponding PWM channel.<br>**Note2:** User can clear each interrupt flag by writing a one to corresponding bit |

### Capture Control Register 0 (PWM_CAPCTL01)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_CAPCTL01** | PWM_BA+0x50 | R/W | Capture Control Register 0 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| CFLIF1 | CRLIF1 | Reserved | CAPIF1 | CAPEN1 | CFLIEN1 | CRLIEN1 | CAPINV1 |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| CFLIF0 | CRLIF0 | Reserved | CAPIF0 | CAPEN0 | CFLIEN0 | CRLIEN0 | CAPINV0 |

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23] | **CFLIF1** | **Capture Falling Latch Interrupt Flag**<br>0 = No capture falling latch condition happened.<br>1 = Capture falling latch condition happened, and this flag will be set to high.<br>**Note:** This bit must be cleared by writing 1 to it. |
| [22] | **CRLIF1** | **Capture Rising Latch Interrupt Flag**<br>0 = No capture rising latch condition happened.<br>1 = Capture rising latch condition happened, and this flag will be set to high.<br>**Note:** This bit must be cleared by writing 1 to it. |
| [21] | **Reserved** | Reserved. |
| [20] | **CAPIF1** | **Capture 1 Interrupt Indication**<br>0 = Interrupt Flag OFF.<br>1 = Interrupt Flag ON.<br>**Note:** If this bit is “1”, PWM-counter 1 will not reload when the next capture interrupt occurs. Write “1” to clear. |
| [19] | **CAPEN1** | **Capture Channel 1 Function Enable Control**<br>0 = Capture function Disabled.<br>1 = Capture function Enabled.<br>**Note1:** When Enabled, Capture latched the PMW-counter 1 and saved to PWM_RCAPDAT1 (Rising latch) and PWM_FCAPDAT1 (Falling latch).<br>**Note2:** When Disabled, Capture does not update PWM_RCAPDAT1 and PWM_FCAPDAT1, and disable Channel 1 Interrupt. |
| [18] | **CFLIEN1** | **Channel1 Falling Interrupt Enable Control**<br>0 = Channel1 Falling Interrupt Disabled.<br>1 = Channel1 Falling Interrupt Enabled.<br>**Note:** When Enabled, if Capture detects Channel 1 has falling transition, Capture |

|  |  |  |
|---|---|---|
|  |  | issues an Interrupt. |
| [17] | **CRLIEN1** | **Channel 1 Rising Interrupt Enable Control**<br>0 = Channel 1 Rising Interrupt Disabled.<br>1 = Channel 1 Rising Interrupt Enabled.<br>**Note:** When Enabled, if Capture detects Channel 1 has rising transition, Capture issues an Interrupt. |
| [16] | **CAPINV1** | **Capture 1 Inverter Enable Control**<br>0 = Capture source inverter Disabled.<br>1 = Capture source inverter Enabled. |
| [15:8] | **Reserved** | Reserved. |
| [7] | **CFLIF0** | **Capture Falling Latch Interrupt Flag**<br>0 = No capture falling latch condition happened.<br>1 = Capture falling latch condition happened, and this flag will be set to high.<br>**Note:** This bit must be cleared by writing 1 to it. |
| [6] | **CRLIF0** | **Capture Rising Latch Interrupt Flag**<br>0 = No capture rising latch condition happened.<br>1 = Capture rising latch condition happened, and this flag will be set to high.<br>**Note:** This bit must be cleared by writing 1 to it. |
| [5] | **Reserved** | Reserved. |
| [4] | **CAPIF0** | **Capture 0 Interrupt Indication**<br>0 = Interrupt Flag OFF.<br>1 = Interrupt Flag ON.<br>**Note:** If this bit is “1”, PWM-counter 0 will not reload when the next capture interrupt occur. Write “1” clear. |
| [3] | **CAPEN0** | **Capture Channel 0 Function Enable Control**<br>0 = Capture function Disabled.<br>1 = Capture function Enabled.<br>**Note1:** When Enabled, Capture latched the PWM-counter value and saved to PWM_RCAPDAT0 (Rising latch) and PWM_FCAPDAT0 (Falling latch).<br>**Note2:** When Disabled, Capture does not update PWM_RCAPDAT0 and PWM_FCAPDAT0, and disable Channel 0 Interrupt. |
| [2] | **CFLIEN0** | **Channel 0 Falling Interrupt Enable Control**<br>0 = Channel 0 Falling Interrupt Disabled.<br>1 = Channel 0 Falling Interrupt Enabled.<br>**Note:** When Enabled, if Capture detects Channel 0 has falling transition, Capture issues an Interrupt. |
| [1] | **CRLIEN0** | **Channel 0 Rising Interrupt Enable Control**<br>0 = Channel 0 Rising Interrupt Enable Disabled.<br>1 = Channel 0 Rising Interrupt Enable Enabled.<br>**Note:** When Enabled, if Capture detects Channel 0 has rising transition, Capture issues an Interrupt. |
| [0] | **CAPINV0** | **Capture 0 Inverter Enable Control**<br>0 = Capture source inverter Disabled.<br>1 = Capture source inverter Enabled. Reverse the input signal from GPIO. |

## Capture Control Register 1 (PWM_CAPCTL23)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_CAPCTL23** | PWM_BA+0x54 | R/W | Capture Control Register 1 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| CFLIF3 | CRLIF3 | Reserved | CAPIF3 | CAPEN3 | CFLIEN3 | CRLIEN3 | CAPINV3 |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| CFLIF2 | CRLIF2 | Reserved | CAPIF2 | CAPEN2 | CFLIEN2 | CRLIEN2 | CAPINV2 |

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23] | **CFLIF3** | **Capture Falling Latch Interrupt Flag**<br>0 = No capture falling latch condition happened.<br>1 = Capture falling latch condition happened, this flag will be set to high.<br>**Note:** This bit must be cleared by writing 1 to it. |
| [22] | **CRLIF3** | **Capture Rising Latch Interrupt Flag**<br>0 = No capture rising latch condition happened.<br>1 = Capture rising latch condition happened, this flag will be set to high.<br>**Note:** This bit must be cleared by writing 1 to it. |
| [21] | **Reserved** | Reserved. |
| [20] | **CAPIF3** | **Capture 3 Interrupt Indication**<br>0 = Interrupt Flag OFF.<br>1 = Interrupt Flag ON.<br>**Note:** If this bit is “1”, PWM-counter 3 will not reload when next capture interrupt occur. |
| [19] | **CAPEN3** | **Capture Channel 3 Function Enable Control**<br>0 = Capture function Disabled.<br>1 = Capture function Enabled.<br>**Note:** When Enabled, Capture latched the PMW-counter and saved to PWM_RCAPDAT3 (Rising latch) and PWM_FCAPDAT3 (Falling latch). When Disabled, Capture does not update PWM_RCAPDAT3 and PWM_FCAPDAT3, and disable Channel 3 Interrupt. |
| [18] | **CFLIEN3** | **Channel 3 Falling Interrupt Enable Control**<br>0 = Channel 3 Falling Interrupt Disabled.<br>1 = Channel 3 Falling Interrupt Enabled.<br>**Note:** When Enabled, if Capture detects Channel 3 has falling transition, Capture issues |

| | | |
|---|---|---|
| | | an Interrupt. |
| [17] | **CRLIEN3** | **Channel 3 Rising Interrupt Enable Control**<br>0 = Channel 3 Rising Interrupt Disabled.<br>1 = Channel 3 Rising Interrupt Enabled.<br>**Note:** When Enabled, if Capture detects Channel 3 has rising transition, Capture issues an Interrupt. |
| [16] | **CAPINV3** | **Capture 3 Inverter Enable Control**<br>0 = Capture source inverter Disabled.<br>1 = Capture source inverter Enabled. Reverse the input signal from GPIO |
| [15:8] | **Reserved** | Reserved. |
| [7] | **CFLIF2** | **Capture Falling Latch Interrupt Flag**<br>0 = No capture falling latch condition happened.<br>1 = Capture falling latch condition happened, this flag will be set to high.<br>**Note:** This bit must be cleared by writing 1 to it. |
| [6] | **CRLIF2** | **Capture Rising Latch Interrupt Flag**<br>0 = No capture rising latch condition happened.<br>1 = Capture rising latch condition happened, this flag will be set to high.<br>**Note:** This bit must be cleared by writing 1 to it. |
| [5] | **Reserved** | Reserved. |
| [4] | **CAPIF2** | **Capture 2 Interrupt Indication**<br>0 = Interrupt Flag OFF.<br>1 = Interrupt Flag ON.<br>**Note:** If this bit is “1”, PWM-counter 2 will not reload when next capture interrupt occur. |
| [3] | **CAPEN2** | **Capture Channel 2 Function Enable Control**<br>0 = Capture function Disabled.<br>1 = Capture function Enabled.<br>**Note:** When Enabled, Capture latched the PMW-counter value and saved to PWM_RCAPDAT2 (Rising latch) and PWM_FCAPDAT2 (Falling latch). When Disabled, Capture does not update PWM_RCAPDAT2 and PWM_FCAPDAT2, and disable Channel 2 Interrupt. |
| [2] | **CFLIEN2** | **Channel 2 Falling Interrupt Enable Control**<br>0 = Channel 2 Falling Interrupt Disabled.<br>1 = Channel 2 Falling Interrupt Enabled.<br>**Note:** When Enabled, if Capture detects Channel 2 has falling transition, Capture issues an Interrupt. |
| [1] | **CRLIEN2** | **Channel 2 Rising Interrupt Enable Control**<br>0 = Channel 2 Rising Interrupt Disabled.<br>1 = Channel 2 Rising Interrupt Enabled.<br>**Note:** When Enabled, if Capture detects Channel 2 has rising transition, Capture issues an Interrupt. |
| [0] | **CAPINV2** | **Capture 2 Inverter Enable Control**<br>0 = Capture source inverter Disabled.<br>1 = Capture source inverter Enabled. Reverse the input signal from GPIO |

### Capture Rising Latch Register3-0 (PWM_RCAPDAT3-0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_RCAPDAT0** | PWM_BA+0x58 | R/W | Capture Rising Latch Register (Channel 0) | 0x0000_0000 |
| **PWM_RCAPDAT1** | PWM_BA+0x60 | R/W | Capture Rising Latch Register (Channel 1) | 0x0000_0000 |
| **PWM_RCAPDAT2** | PWM_BA+0x68 | R/W | Capture Rising Latch Register (Channel 2) | 0x0000_0000 |
| **PWM_RCAPDAT3** | PWM_BA+0x70 | R/W | Capture Rising Latch Register (Channel 3) | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **RCAPDAT** | **Capture Rising Latch Register**<br>Latch the PWM counter when Channel 0/1/2/3 has rising transition. |

### Capture Falling Latch Register3-0 (PWM_FCAPDAT3-0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_FCAPDAT0** | PWM_BA+0x5C | R/W | Capture Falling Latch Register (Channel 0) | 0x0000_0000 |
| **PWM_FCAPDAT1** | PWM_BA+0x64 | R/W | Capture Falling Latch Register (Channel 1) | 0x0000_0000 |
| **PWM_FCAPDAT2** | PWM_BA+0x6C | R/W | Capture Falling Latch Register (Channel 2) | 0x0000_0000 |
| **PWM_FCAPDAT3** | PWM_BA+0x74 | R/W | Capture Falling Latch Register (Channel 3) | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **FCAPDAT** | **Capture Falling Latch Register**<br>Latch the PWM counter when Channel 0/1/2/3 has Falling transition. |

### Capture Input Enable Register (PWM_CAPINEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_CAPINEN** | PWM_BA+0x78 | R/W | Capture Input Enable Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:4] | **Reserved** | Reserved. |
| [n]=0,1,2,3 | **CAPINEN[n]** | **Capture Input Enable Control**<br>0 = PWM Channel capture input path Disabled. The input of PWM channel capture function is always regarded as 0<br>1 = PWM Channel capture input path Enabled. The input of PWM channel capture function comes from correlative multifunction pin<br>n = 0, Capture channel 0 is from PB.10 or PC.9.<br>n = 1, Capture channel 1 is from PB.11 or PC.10.<br>n = 2, Capture channel 2 is from PB.12 or PC.11.<br>n = 3, Capture channel 3 is from PB.13 or PC.12. |

### PWM Output Enable Register (PWM_POEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **PWM_POEN** | PWM_BA+0x7C | R/W | PWM Output Enable Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:4] | **Reserved** | Reserved. |
| [3:0] | **POEN** | **PWM Counter Output Enable Control**<br>0 = PWM Counter Output Disabled.<br>1 = PWM Counter Output Enabled.<br>**Note:** Each bit controls the corresponding PWM channel. |

## 6.7 Watchdog Timer (WDT)

### 6.7.1 Overview

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

### 6.7.2 Features

- 18-bit free running up counter for Watchdog Timer time-out interval.
- Selectable time-out interval ($2^4$ ~ $2^{18}$) WDT_CLK cycle and the time-out interval period is 32.5 ms ~ 8.224 s if WDT_CLK = 32 kHz.
- System kept in reset state for a period of (1 / WDT_CLK) * 63
- Supports selectable Watchdog Timer reset delay period, including 1026、130 、18 or 3 WDT_CLK reset delay period.
- Supports Watchdog Timer time-out wake-up function when Watchdog Timer clock source is selected as 32 kHz low-speed oscillator.

### 6.7.3 Block Diagram

The Watchdog Timer clock control and block diagram are shown in Figure 6.7-1 and Figure 6.7-2 respectively.

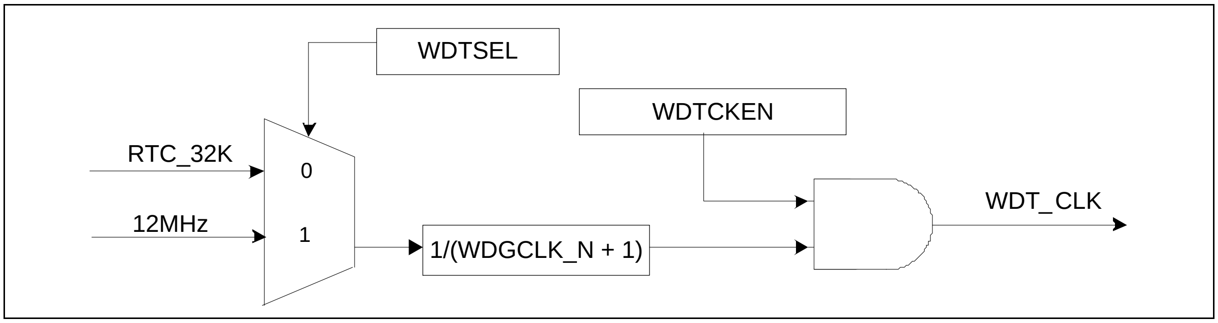
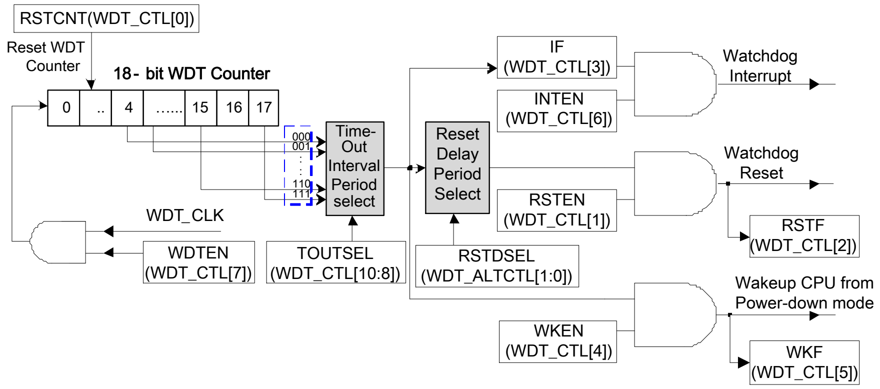


Figure 6.7-1 Watchdog Timer Clock Control Diagram



**Note1:** Watchdog Timer resets CPU and lasts 63 WDT_CLK.

**Note2:** If user intends to use Watchdog Timer to wake up Power-down mode, it is recommended that CPU clock source is set the same as Watchdog Timer clock source before CPU enters Power-down mode.

**Note3:** The Watchdog Timer reset delay period can be selected as 3/18/130/1026 WDT_CLK.

Figure 6.7-2 Watchdog Timer Block Diagram

### 6.7.4 Functional Description

The Watchdog Timer (WDT) includes an 18-bit free running up counter with programmable time-out intervals. The Table 6.7-1 shows the Watchdog Timer time-out interval period selection and the Figure 6.7-3 shows the Watchdog Timer time-out interval and reset period timing.

### Watchdog Timer Time-out Interrupt

Setting WDTEN (WDT_CTL[7]) bit to 1 will enable the Watchdog Timer function and the Watchdog Timer counter to start counting up. There are eight time-out interval period can be selected by setting TOUTSEL (WDT_CTL[10:8]). When the Watchdog Timer up counter reaches the TOUTSEL settings, Watchdog Timer time-out interrupt will occur then IF (WDT_CTL[3]) flag will be set to 1 immediately.

### Watchdog Timer Reset Delay Period and Reset System

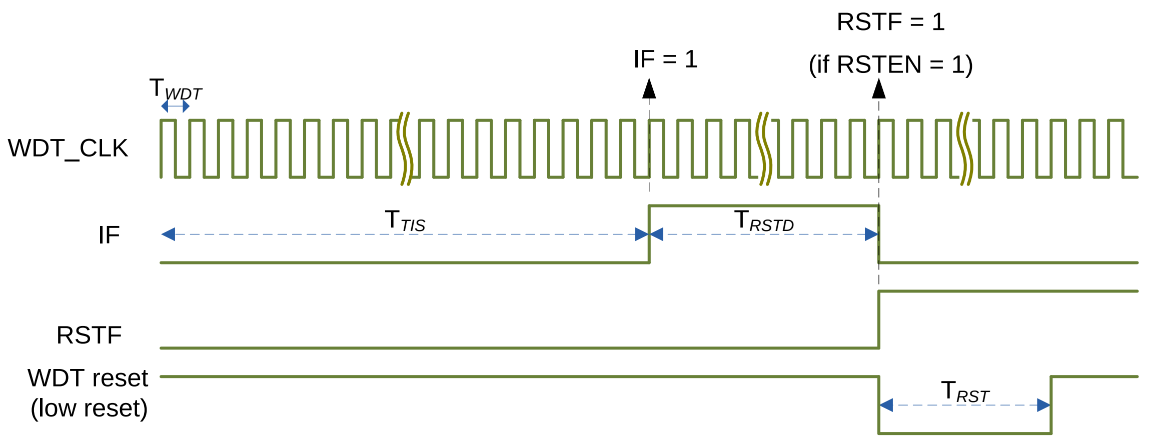
There is a specified $T_{RSTD}$ delay period follows the IF flag is setting to 1. User must enabled RSTCNT (WDT_CTL[0]) bit to reset the 18-bit Watchdog Timer up counter value to avoid generate Watchdog Timer time-out reset signal before the $T_{RSTD}$ delay period expires. If the Watchdog Timer up counter value has not been cleared after the specific $T_{RSTD}$ delay period expires, the Watchdog Timer control will set RSTF (WDT_RSTSTS[2]) flag to 1 if RSTEN (WDT_CTL[1]) bit is enabled, then chip enters to reset state immediately. Refer to Figure 6.7-3, the $T_{RST}$ reset period will keep last 63 Watchdog Timer clocks then chip restart executing program from reset vector (0x0000_0000). The RSTF flag will keep 1 after Watchdog Timer time-out reset the chip, user can check RSTF flag by software to recognize the system has been reset by Watchdog Timer time-out reset or not.

### Watchdog Timer Wake-up

If Watchdog Timer clock source is selected to 32 kHz, system can be woken up from Power-down mode while Watchdog Timer time-out interrupt signal is generated and WKEN (WDT_CTL[4]) bit enabled. In the meanwhile, the WKF (WDT_CTL[5]) flag will set to 1 automatically, user can check WKF flag by software to recognize if the system has been woken up by Watchdog Timer time-out interrupt.

| WTIS | Time-Out Interval Selection<br>$T_{TIS}$ | Interrupt Period<br>$T_{INT}$ | Reset Delay Period<br>$T_{RSTD}$ |
|---|---|---|---|
| 000 | $2^4 * T_{WDT}$ | $1024 * T_{WDT}$ | $(3/18/130/1026) * T_{WDT}$ |
| 001 | $2^6 * T_{WDT}$ | $1024 * T_{WDT}$ | $(3/18/130/1026) * T_{WDT}$ |
| 010 | $2^8 * T_{WDT}$ | $1024 * T_{WDT}$ | $(3/18/130/1026) * T_{WDT}$ |
| 011 | $2^{10} * T_{WDT}$ | $1024 * T_{WDT}$ | $(3/18/130/1026) * T_{WDT}$ |
| 100 | $2^{12} * T_{WDT}$ | $1024 * T_{WDT}$ | $(3/18/130/1026) * T_{WDT}$ |
| 101 | $2^{14} * T_{WDT}$ | $1024 * T_{WDT}$ | $(3/18/130/1026) * T_{WDT}$ |
| 110 | $2^{16} * T_{WDT}$ | $1024 * T_{WDT}$ | $(3/18/130/1026) * T_{WDT}$ |
| 111 | $2^{18} * T_{WDT}$ | $1024 * T_{WDT}$ | $(3/18/130/1026) * T_{WDT}$ |

Table 6.7-1 Watchdog Timer Interval Selection



- $T_{WDT}$ *: Watchdog Clock Time Period*
- $T_{TIS}$ *: Watchdog Time-out Interval Period ( (*$2^4$ *~* $2^{18}$*) ** $T_{WDT}$ *)*
- $T_{RSTD}$ *: Watchdog Reset Delay Period*
  *- Selectable 3/18/130/1026 ** $T_{WDT}$ *delay period controlled by RSTDSEL (WDT_ALTCTL[1:0])*
- $T_{RST}$ *: Watchdog Reset Period ( 63 ** $T_{WDT}$ *)*

Figure 6.7-3 Timing of Interrupt and Reset Signal

### 6.7.5 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **WDT Base Address:**<br>**WDT_BA = 0x400E_F000** | | | | |
| **WDT_CTL** | WDT_BA+0x00 | R/W | Watchdog Timer Control Register | 0x0000_0700 |
| **WDT_ALTCTL** | WDT_BA+0x04 | R/W | Watchdog Timer Alternative Control Register | 0x0000_0000 |
| **WDT_RSTSTS** | WDT_BA+0x10 | R | Watchdog Timer Reset Status Register | 0x0000_0700 |

### 6.7.6 Register Description

#### Watchdog Timer Control Register (WDT_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **WDT_CTL** | WDT_BA+0x00 | R/W | Watchdog Timer Control Register | 0x0000_0700 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **ICEDEBUG** | **Reserved** | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | **TOUTSEL** | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **WDTEN** | **INTEN** | **WKF** | **WKEN** | **IF** | **RSTFC** | **RSTEN** | **RSTCNT** |

| Bits | Description | |
|---|---|---|
| [31] | **ICEDEBUG** | **ICE Debug Mode Acknowledge Disable Control**<br>0 = ICE debug mode acknowledgement affects Watchdog Timer counting.<br>Watchdog Timer counter will be held while CPU is held by ICE.<br>1 = ICE debug mode acknowledgement Disabled.<br>Watchdog Timer counter will keep going no matter CPU is held by ICE or not. |
| [30:11] | **Reserved** | Reserved. |
| [10:8] | **TOUTSEL** | **Watchdog Timer Time-out Interval Selection**<br>These three bits select the time-out interval period for the Watchdog Timer.<br>000 = $2^{4}$ *$T_{WDT}$.<br>001 = $2^{6}$ *$T_{WDT}$.<br>010 = $2^{8}$ *$T_{WDT}$.<br>011 = $2^{10}$ *$T_{WDT}$.<br>100 = $2^{12}$ *$T_{WDT}$.<br>101 = $2^{14}$ *$T_{WDT}$.<br>110 = $2^{16}$ *$T_{WDT}$.<br>111 = $2^{18}$ *$T_{WDT}$. |
| [7] | **WDTEN** | **Watchdog Timer Enable Control**<br>0 = Watchdog Timer Disabled (This action will reset the internal counter).<br>1 = Watchdog Timer Enabled. |
| [6] | **INTEN** | **Watchdog Timer Interrupt Enable Control**<br>If this bit is enabled, the Watchdog Timer time-out interrupt signal is generated and inform to CPU.<br>0 = Watchdog Timer interrupt Disabled.<br>1 = Watchdog Timer interrupt Enabled. |

| | | |
|---|---|---|
| [5] | **WKF** | **Watchdog Timer Wake-up Flag**<br>This bit indicates the interrupt wake-up flag status of Watchdog Timer<br>0 = Watchdog Timer does not cause chip wake-up.<br>1 = Chip wake-up from Power-down if Watchdog Timer time-out interrupt signal generated.<br>**Note:** This bit is cleared by writing 1 to it. |
| [4] | **WKEN** | **Watchdog Timer Wake-up Function Enable Control**<br>If this bit is set to 1, while Watchdog Timer interrupt flag IF (WDT_CTL[3]) is generated to 1 and INTEN (WDT_CTL[6] Watchdog Timer interrupt enable) is enabled, the Watchdog Timer time-out interrupt signal will generate a wake-up trigger event to chip.<br>0 = Wake-up trigger event Disabled if Watchdog Timer time-out interrupt signal generated.<br>1 = Wake-up trigger event Enabled if Watchdog Timer time-out interrupt signal generated.<br>**Note:** Chip can be woken-up by Watchdog Timer time-out interrupt signal generated only if Watchdog Timer clock source is selected to 32 kHz oscillator. |
| [3] | **IF** | **Watchdog Timer Interrupt Flag**<br>This bit will set to 1 while Watchdog Timer counter value reaches the selected Watchdog Timer time-out interval<br>0 = Watchdog Timer time-out interrupt did not occur.<br>1 = Watchdog Timer time-out interrupt occurred.<br>**Note:** This bit is cleared by writing 1 to it. |
| [2] | **RSTFC** | **Watchdog Timer Reset Flag Cleared**<br>Write 1 to clear the RSTF (WDT_RSTSTS [2]). |
| [1] | **RSTEN** | **Watchdog Timer Reset Enable Control**<br>Setting this bit will enable the Watchdog Timer time-out reset function If the Watchdog Timer counter value has not been cleared after the specific Watchdog Timer reset delay period expires.<br>0 = Watchdog Timer time-out reset function Disabled.<br>1 = Watchdog Timer time-out reset function Enabled. |
| [0] | **RSTCNT** | **Clear Watchdog Timer**<br>0 = No effect.<br>1 = Reset the internal 18-bit Watchdog Timer counter.<br>**Note:** This bit will be automatically cleared by hardware. |

### Watchdog Timer Alternative Control Register (WDT_ALTCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **WDT_ALTCTL** | WDT_BA+0x04 | R/W | Watchdog Timer Alternative Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1:0] | **RSTDSEL** | **Watchdog Timer Reset Delay Selection**<br>When Watchdog Timer time-out happened, software has a time named Watchdog Timer reset delay period to clear Watchdog Timer counter to prevent Watchdog Timer time-out reset happened. Software can select a suitable value of Watchdog Timer reset delay period for different Watchdog Timer time-out period.<br>00 = Watchdog Timer reset delay period is (1024+2) * WDT_CLK.<br>01 = Watchdog Timer reset delay period is (128+2) * WDT_CLK.<br>10 = Watchdog Timer reset delay period is (16+2) * WDT_CLK.<br>11 = Watchdog Timer reset delay period is (1+2) * WDT_CLK.<br>**Note:** This bit will be reset to 0 if Watchdog Timer time-out reset happened. |

### Watchdog Timer Reset Status Register (WDT_RSTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **WDT_RSTSTS** | WDT_BA+0x10 | R | Watchdog Timer Reset Status Register | 0x0000_0700 |

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

| Bits | Description | |
|---|---|---|
| [31:3] | **Reserved** | Reserved. |
| [2] | **RSTF** | **Watchdog Timer Reset Flag**<br>This bit indicates the system has been reset by Watchdog Timer time-out reset or not.<br>0 = Watchdog Timer time-out reset did not occur.<br>1 = Watchdog Timer time-out reset occurred.<br>**Note:** This bit is cleared by writing 1 to RSTFC (WDT_CTL [2]). |
| [1:0] | **Reserved** | Reserved. |

## 6.8 Window Watchdog Timer (WWDT)

### 6.8.1 Overview

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

### 6.8.2 Features

- 6-bit down counter CNTDAT (WWDT_CNT[5:0]) and 6-bit compare value CMPDAT (WWDT_CTL[21:16]) to make the window period flexible
- Selectable maximum 11-bit WWDT clock prescale PSCSEL (WWDT_CTL[11:8]) to make WWDT time-out interval variable

### 6.8.3 Block Diagram

The Window Watchdog Timer block diagram is shown as follows.

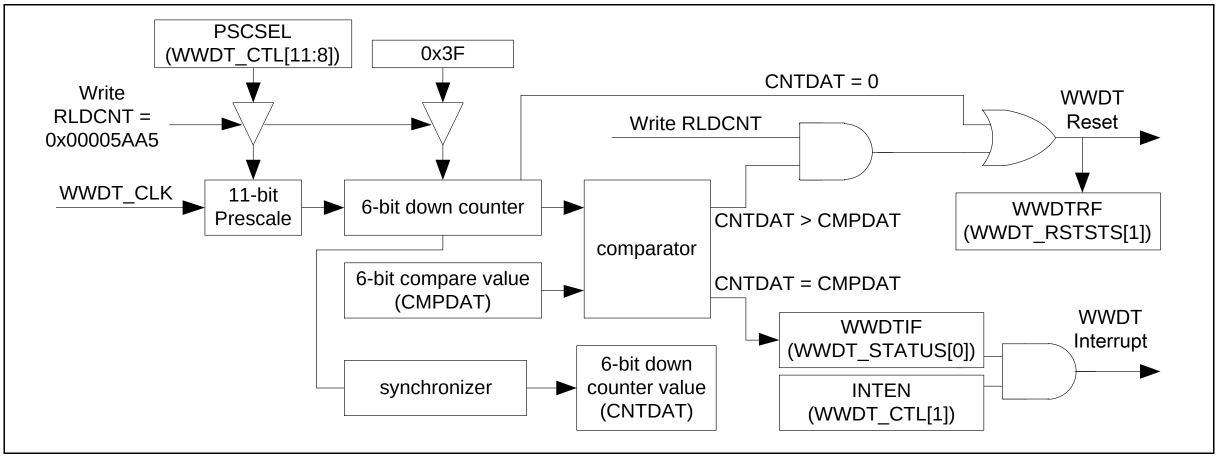


Figure 6.8-1 Window Watchdog Timer Block Diagram

### 6.8.4 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 RTC clock (32 kHz) with a programmable maximum  11-bit prescale value. Also, the programmable 11-bit prescale value is controlled by PSCSEL (WWDT_CTL[11:8] WWDT prescale period select) and the correlate of PSCSEL and prescale value are listed in the following table.

| PSCSEL | Prescaler Value | Time-Out Period | Time-Out Interval<br>(WWDT_CLK= 32 KHz) |
|---|---|---|---|
| 0000 | 1 | 1 * 64 * $T_{WWDT}$ | 2 ms |
| 0001 | 2 | 2 * 64 * $T_{WWDT}$ | 4 ms |

NUC505 SERIES TECHNICAL REFERENCE MANUAL

| | | | |
|---|---|---|---|
| 0010 | 4 | 4 * 64 * $T_{WWDT}$ | 8 ms |
| 0011 | 8 | 8 * 64 * $T_{WWDT}$ | 16 ms |
| 0100 | 16 | 16 * 64 * $T_{WWDT}$ | 32 ms |
| 0101 | 32 | 32 * 64 * $T_{WWDT}$ | 64 ms |
| 0110 | 64 | 64 * 64 * $T_{WWDT}$ | 128 ms |
| 0111 | 128 | 128 * 64 * $T_{WWDT}$ | 256 ms |
| 1000 | 192 | 192 * 64 * $T_{WWDT}$ | 384 ms |
| 1001 | 256 | 256 * 64 * $T_{WWDT}$ | 512 ms |
| 1010 | 384 | 384 * 64 * $T_{WWDT}$ | 768 ms |
| 1011 | 512 | 512 * 64 * $T_{WWDT}$ | 1024 ms |
| 1100 | 768 | 768 * 64 * $T_{WWDT}$ | 1536 ms |
| 1101 | 1024 | 1024 * 64 * $T_{WWDT}$ | 2048 ms |
| 1110 | 1536 | 1536 * 64 * $T_{WWDT}$ | 3072 ms |
| 1111 | 2048 | 2048 * 64 * $T_{WWDT}$ | 4096 ms |

Table 6.8-1 Window Watchdog Prescaler Value Selection

The Window Watchdog Timer can be enabled only once by software setting WWDTEN (WWDT_CTL[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 (WWDT_STATUS[0] WWDT compare match interrupt flag) is set to 1 if the WWDT counter value is equal to CMPDAT (WWDT_CTL [21:16] WWDT window compare register) value; if the INTEN (WWDT_CTL[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 0. Before WWDT counter down counting to 0, software can write **0x00005AA5** to WWDT_RLDCNT register to reload WWDT internal counter value to 0x3F to prevent WWDT time-out reset from happening when the current WWDT counter value (WWDT_CNT value) is equal to or less than CMPDAT value. If the current WWDT counter value (WWDT_CNT value) is greater than CMPDAT value and software writes **0x00005AA5** to the WWDT_RLDCNT register, WWDT reset signal will be generated to cause chip reset. Figure 6.8-2 shows the reset and reload behavior of WWDT.

To avoid program running to disable Window Watchdog Timer counter counting unexpectedly, the control register WWDT_CTL 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 (PSCSEL) or change window compare value (CMPDAT) while WWDTEN bit has been enabled by software unless chip is reset.

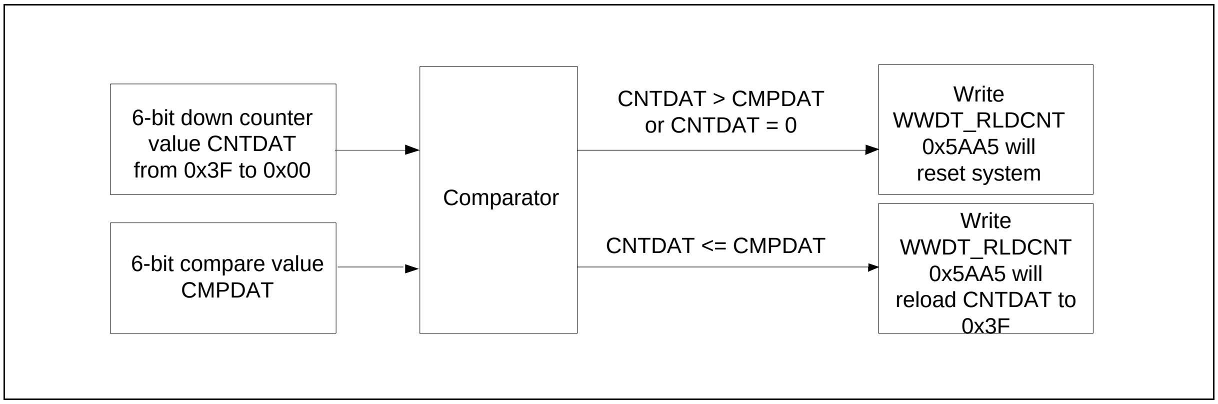

6-bit down counter value CNTDAT from 0x3F to 0x00
6-bit compare value CMPDAT
Comparator
CNTDAT > CMPDAT or CNTDAT = 0
CNTDAT <= CMPDAT
Write WWDT_RLDCNT 0x5AA5 will reset system
Write WWDT_RLDCNT 0x5AA5 will reload CNTDAT to 0x3F

Figure 6.8-2 Window Watchdog Timer Reset and Reload Behavior

When user writes **0x00005AA5** to WWDT_RLDCNT register to reload WWDT counter value to 0x3F, it needs 3 WWDT clocks to sync reload command to actually perform reload action. This means if user set PSCSEL (WWDT_CTL[11:8] : WWDT prescale period select) to 0000, the prescale value should be as 1, and the CMPDAT (WWDT_CTL[21:16] : WWDT window compare register) value must be larger than 2; otherwise, writing WWDT_RLDCNT to reload WWDT counter value to 0x3F is unavailable while WWDTIF (WWDT_STATUS[0]) is generated and WWDT reset system event always happened. The following table shows the limitation of CMPDAT.

| PSCSEL | Prescale Value | Valid CMPDAT Value |
|---|---|---|
| 0000 | 1 | 0x3 ~ 0x3F |
| 0001 | 2 | 0x2 ~ 0x3F |
| Others | Others | 0x0 ~ 0x3F |

Table 6.8-2 CMPDAT Setting Limitation

### 6.8.5 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **WWDT Base Address:**<br>**WWDT_BA = 0x400E_F100** | | | | |
| **WWDT_RLDCNT** | WWDT_BA+0x00 | W | Window Watchdog Timer Reload Counter Register | 0x0000_0000 |
| **WWDT_CTL** | WWDT_BA+0x04 | R/W | Window Watchdog Timer Control Register | 0x003F_0800 |
| **WWDT_STATUS** | WWDT_BA+0x08 | R/W | Window Watchdog Timer Status Register | 0x0000_0000 |
| **WWDT_CNT** | WWDT_BA+0x0C | R | Window Watchdog Timer Counter Value Register | 0x0000_003F |
| **WWDT_RSTSTS** | WWDT_BA+0x18 | R | Window Watchdog Timer Reset Status Register | 0x0000_0000 |

### 6.8.6 Register Description

**Window Watchdog Timer Reload Counter Register (WWDT_RLDCNT)**

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **WWDT_RLDCNT** | WWDT_BA+0x00 | W | Window Watchdog Timer Reload Counter Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **RLDCNT** | **WWDT Reload Counter Bits**<br>Writing 0x00005AA5 to this register will reload the Window Watchdog Timer counter value to 0x3F.<br>Software can only write RLDCNT to reload WWDT counter value when current WWDT counter value between 0 and CMPDAT. If software writes RLDCNT when current WWDT counter value is larger than CMPDAT, WWDT reset signal will generate immediately. |

### Window Watchdog Timer Control Register (WWDT_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **WWDT_CTL** | WWDT_BA+0x04 | R/W | Window Watchdog Timer Control Register | 0x003F_0800 |

This register can be write only one time after chip is powered on or reset.

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

| Bits | Description | |
|---|---|---|
| [31] | **ICEDEBUG** | **ICE Debug Mode Acknowledge Disable Control**<br>0 = ICE debug mode acknowledgement effects WWDT counting.<br>WWDT down counter will be held while CPU is held by ICE.<br>1 = ICE debug mode acknowledgement Disabled.<br>WWDT down counter will keep going no matter CPU is held by ICE or not. |
| [30:22] | **Reserved** | Reserved. |
| [21:16] | **CMPDAT** | **WWDT Window Compare Bits**<br>Set this register to adjust the valid reload window.<br>Software can only write RLDCNT to reload WWDT counter value when current WWDT counter value between 0 and CMPDAT. If Software writes RLDCNT when current WWDT counter value larger than CMPDAT, WWDT reset signal will generate immediately. |
| [15:12] | **Reserved** | Reserved. |
| [11:8] | **PSCSEL** | **WWDT Counter Prescale Period Selection**<br>0000 = Pre-scale is 1; Max time-out period is 1 * 64 * $T_{WWDT}$.<br>0001 = Pre-scale is 2; Max time-out period is 2 * 64 * $T_{WWDT}$.<br>0010 = Pre-scale is 4; Max time-out period is 4 * 64 * $T_{WWDT}$.<br>0011 = Pre-scale is 8; Max time-out period is 8 * 64 * $T_{WWDT}$.<br>0100 = Pre-scale is 16; Max time-out period is 16 * 64 * $T_{WWDT}$.<br>0101 = Pre-scale is 32; Max time-out period is 32 * 64 * $T_{WWDT}$.<br>0110 = Pre-scale is 64; Max time-out period is 64 * 64 * $T_{WWDT}$.<br>0111 = Pre-scale is 128; Max time-out period is 128 * 64 * $T_{WWDT}$.<br>1000 = Pre-scale is 192; Max time-out period is 192 * 64 * $T_{WWDT}$.<br>1001 = Pre-scale is 256; Max time-out period is 256 * 64 * $T_{WWDT}$.<br>1010 = Pre-scale is 384; Max time-out period is 384 * 64 * $T_{WWDT}$. |

NUC505 SERIES TECHNICAL REFERENCE MANUAL

|  |  |  |
|---|---|---|
|  |  | 1011 = Pre-scale is 512; Max time-out period is 512 * 64 * $T_{WWDT}$.<br>1100 = Pre-scale is 768; Max time-out period is 768 * 64 * $T_{WWDT}$.<br>1101 = Pre-scale is 1024; Max time-out period is 1024 * 64 * $T_{WWDT}$.<br>1110 = Pre-scale is 1536; Max time-out period is 1536 * 64 * $T_{WWDT}$.<br>1111 = Pre-scale is 2048; Max time-out period is 2048 * 64 * $T_{WWDT}$. |
| [7:2] | **Reserved** | Reserved. |
| [1] | **INTEN** | **WWDT Interrupt Enable Control**<br>If this bit is enabled, the WWDT counter compare match interrupt signal is generated and inform to CPU.<br>0 = WWDT counter compare match interrupt Disabled.<br>1 = WWDT counter compare match interrupt Enabled. |
| [0] | **WWDTEN** | **WWDT Enable Control**<br>Set this bit to enable Window Watchdog Timer counter counting.<br>0 = Window Watchdog Timer counter is stopped.<br>1 = Window Watchdog Timer counter is starting counting. |

### Window Watchdog Timer Status Register (WWDT_STATUS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **WWDT_STATUS** | WWDT_BA+0x08 | R/W | Window Watchdog Timer Status Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1] | **WWDTRFC** | **WWDT Timer-out Reset Flag Cleared**<br>When the window watch dog reset happened, the register WWDTRF (WWDT_RSTSTS [1]) will be set to 1. Write 1 to this bit and the WWDTRF (WWDT_RSTSTS [1]) will be cleared. |
| [0] | **WWDTIF** | **WWDT Compare Match Interrupt Flag**<br>This bit indicates the interrupt flag status of WWDT while WWDT counter value matches CMPDAT value.<br>0 = No effect.<br>1 = WWDT counter value matches CMPDAT value.<br>This bit is cleared by writing 1 to WWDT_STATUS[0] |

NUC505 SERIES TECHNICAL REFERENCE MANUAL

### Window Watchdog Timer Counter Value Register (WWDT_CNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **WWDT_CNT** | WWDT_BA+0x0C | R | Window Watchdog Timer Counter Value Register | 0x0000_003F |

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

| Bits | Description | |
|---|---|---|
| [31:6] | **Reserved** | Reserved. |
| [5:0] | **CNTDAT** | **WWDT Counter Value**<br>This register reflects the current WWDT counter value and is read only. |

### Window Watchdog Timer Reset Status Register (WWDT_RSTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **WWDT_RSTSTS** | WWDT_BA+0x18 | R | Window Watchdog Timer Reset Status Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1] | **WWDTRF** | **WWDT Timer-out Reset Flag**<br>This bit indicates the system has been reset by WWDT time-out reset or not.<br>0 = WWDT time-out reset did not occur.<br>1 = WWDT time-out reset occurred.<br>This bit is cleared by writing 1 to WWDTRFC (WWDT_STATUS [1]) |
| [0] | **Reserved** | Reserved. |

## 6.9 Real Time Clock (RTC)

### 6.9.1 Overview

The Real Time Clock (RTC) block can be operated by independent power supply while the system power is off. The RTC uses a 32.768 kHz external crystal (LXT) or internal oscillator (LIRC), and offers programmable time tick and alarm match interrupts. The data format of time and calendar messages are expressed in BCD format. A digital frequency compensation feature is available to compensate the frequency accuracy of external crystal oscillator (LXT) or internal oscillator (LIRC).

The RTC controller also offers 32 bytes spare registers to store user’s important information.

The wake-up signal is used to wake the system from Idle mode, Power-down mode and Deep Power-down mode.

### 6.9.2 Features

- Supports real time counter in RTC_TIME (hour, minute, second) and calendar counter in RTC_CAL (year, month, day) for RTC time and calendar check
- Supports alarm time (hour, minute, second) and calendar (year, month, day) settings in RTC_TALM and RTC_CALM
- Selectable 12-hour or 24-hour time scale in RTC_CLKFMT register
- Supports Leap Year indication in RTC_LEAPYEAR register
- Supports Day of the Week counter in RTC_WEEKDAY register
- Frequency of RTC clock source compensate by RTC_FREQADJ register
- All time and calendar message expressed in BCD format
- Supports periodic RTC Time Tick interrupt with 8 period interval options 1/128, 1/64, 1/32, 1/16, 1/8, 1/4, 1/2 and 1 second
- Supports RTC Time Tick and Alarm Match interrupt
- Supports chip wake-up from Idle mode, Power-down mode and Deep Power-down mode while a RTC interrupt signal is generated
- Supports 32 bytes spare registers and these registers values are preserved when RTC power domain is existed

### 6.9.3 Block Diagram

The block diagram of Real Time Clock (RTC) is depicted as follows.

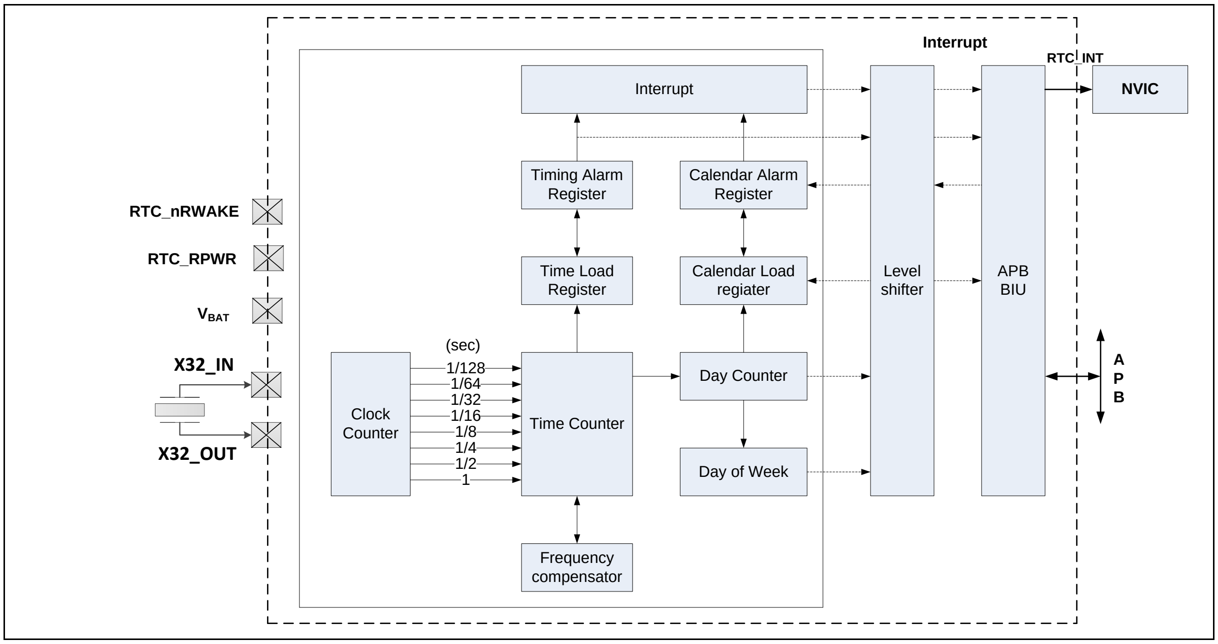


Figure 6.9-1 RTC Block Diagram

### 6.9.4 Basic Configuration

The default RTC controller clock source is internal oscillator (LIRC) when $V_{BAT}$ is connected to 3.3V supply. If user want to use external Crystal (LXT) to connect X32_IN and X32_OUT, user can disable internal oscillator (LIRC) by setting CKSRC (RTC_CLKSRC[0] to 0 and check whether CBEN (RTC_SET[1]) is 1.

The whole chip power system can use RTC_nRWAKE and RTC_RPWR to control external power source. When RTC detects an enable key by RTC_nRWAKE and the external power source will be enabled by RTC_RPWR.

### 6.9.5 Functional Description

When NUC505 is powered on, RTC_nRWAKE needs to be grounded so that the RTC function can be enabled. After power on, if the RTC_nRWAKE pin continues low for a period of time, the system hardware will automatically shut down. Time period can set by register POWOFFT (RTC_POWCTL[19:16]). After shut down, the RTC_nRWAKE pin must be powered on or powered down again before the system is restarted. If you do not want the system to shut down, before shut down set POWOFFEN (RTC_POWCTL[2]) to 0 to disable the RTC hardware power-down function.

#### *6.9.5.1 RTC Initiation*

When a RTC block is powered on, RTC is at reset state. User needs to write 0xa5eb1357 to the RTC initial register RTC_INIT (INIT[31:0]) to make RTC exit from reset state. Once the RTC_INIT register is written as 0xa5eb1357, the RTC will be in normal active state permanently. User can read the Active bit (INIT[0]) to check if the RTC is in normal active state or reset state.

#### *6.9.5.2 Access to RTC register*

Due to clock frequency difference between RTC clock and system clock, when user writes new data to any of the RTC registers, the data will not be updated immediately. Therefore, user can wait until a flag RTC_SYNC (SYNC[0]) is set to 1 to write another register continuously.

#### *6.9.5.3 RTC Read/Write Enable*

The RWEN (RTC_RWEN[15:0]) is served as read/write access of RTC registers to unlock register read/write protection function. If RTC_RWEN[15:0] is written to 0xa965, user can read register access enable flag RWENF (RTC_RWEN[16]) to check the RTC registers are read/write accessible or locked.

#### *6.9.5.4 Frequency Compensation*

The RTC source clock may not be as precise as exactly 32768 Hz and the RTC_FREQADJ register allows user to make digital compensation to the RTC source clock.

Following is a compensation example for the real RTC source clock.

**Example:**

Set ADJTRG (RTC_FREQADJ[31]) =1.

Wait ADJTRG (RTC_FREQADJ[31]) until it is cleared

Read content of CALCNT (RTC_CALCNT[31:0])

Calculate the RTC clock from the value of RTC_CALCNT register,

RTC_Clock_Rate = (PCLK/RTC_CALCNT)*32768

RTC_FREQADJ[23:8] = INTEGER = Integer part of RTC_Clock_Rate-1

RTC_FREQADJ[7:0] = FRACTION = (Fractional part of RTC_Clock_Rate)*60-1

If RTC_CALCNT=50103132, then

RTC_Clock_Rate = (50000000/50103132)*32768=32700.5505

RTC_FREQADJ[23:8] = INTEGER =32699

RTC_FREQADJ[7:0] = FRACTION = 0.5505*60-1=32.03

The FRACTION (RTC_FREQADJ[7:0] should be set to 32 due to round the 32.03 value.

RTC_FREQADJ[23:0] register should be as 0x7FBB20

#### *6.9.5.5 Time and Calendar Counter*

RTC_TIME and RTC_CAL are used to load the time and calendar. RTC_TALM and RTC_CALM are used for alarm. They are all represented by BCD.

#### *6.9.5.6 12/24 hour Time Scale Selection*

The 12/24 hour time scale selection depends on 24HEN (RTC_CLKFMT[0]).

When RTC runs as 12-hour time scale mode, RTC_TIME[21] (the high bit of TENHR[1:0]) means AM/PM indication. (If RTC_TIME[21] is 1, it indicates PM time message.)

| 24-Hour Time Scale<br>(24HEN = 1) | 24-Hour Time Scale<br>(24HEN = 1) | 24-Hour Time Scale<br>(24HEN = 0) | 12-Hour Time Scale<br>(24HEN = 0) (PM Time + 20) |
|---|---|---|---|
| 0x00 | 0x12 | 0x12 (AM12) | 0x32 (PM12) |
| 0x01 | 0x13 | 0x01 (AM01) | 0x21 (PM01) |
| 0x02 | 0x14 | 0x02 (AM02) | 0x22 (PM02) |
| 0x03 | 0x15 | 0x03 (AM03) | 0x23 (PM03) |
| 0x04 | 0x16 | 0x04 (AM04) | 0x24 (PM04) |
| 0x05 | 0x17 | 0x05 (AM05) | 0x25 (PM05) |
| 0x06 | 0x18 | 0x06 (AM06) | 0x26 (PM06) |
| 0x07 | 0x19 | 0x07 (AM07) | 0x27 (PM07) |
| 0x08 | 0x20 | 0x08 (AM08) | 0x28 (PM08) |
| 0x09 | 0x21 | 0x09 (AM09) | 0x29 (PM09) |
| 0x10 | 0x22 | 0x10 (AM10) | 0x30 (PM10) |
| 0x11 | 0x23 | 0x11 (AM11) | 0x31 (PM11) |

#### *6.9.5.7 Day of the Week Counter*

The RTC controller provides day of week in WEEKDA (RTC_WEEKDAY[2:0]). The value is defined from 0 to 6 to represent Sunday to Saturday respectively.

#### *6.9.5.8 Periodic Time tick interrupt*

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

#### *6.9.5.9 Alarm interrupt*

When the real time and calendar message in RTC_TIME and RTC_CAL registers are equal to alarm time and calendar values in RTC_TALM and RTC_CALM registers, the RTC alarm interrupt flag ALMIF (RTC_INTSTS[0]) is set to 1 and the RTC alarm interrupt signal assert if the alarm interrupt enable ALMIEN (RTC_INTEN[0]) is enabled.

The RTC controller provides Time Alarm Mask function (RTC_TALM register) and Calendar Alarm Mask function (RTC_CALM register) to mask the specified digit and generate periodic interrupt without changing the alarm match condition in RTC_TALM and RTC_CALM registers in each alarm interrupt service routine.

#### *6.9.5.10 Application Note*

When system power is off but RTC power is on, data stored in RTC registers will not lost except RTC_INTEN and RTC_INTSTS. Because of clock difference between RTC clock and system clock, when user write new data to any one of the registers, the register will not be updated until 2 RTC clocks later (60us). Hence programmer should consider about access sequence between

RTC_CLKFMT, RTC_TALM and RTC_TIME.

In addition, user must be aware that RTC block does not check whether the loaded data is out of bounds or not. RTC does not check rationality between RTC_WEEKDAY and RTC_CAL either.

1. RTC_TALM, RTC_CALM, RTC_TIME and RTC_CAL registers are all BCD counter.
2. User has to make sure that the loaded values are reasonable, For example, Load RTC_CAL as 201a (year), 13 (month), 00 (day), or RTC_CAL does not match with RTC_WEEKDAY, etc.
3. The register values after powered on are described in the following table:

| **Register** | **Reset State** |
|---|---|
| **RTC_RWEN** | 0 |
| **RTC_CAL** | 05/1/1 (year/month/day) |
| **RTC_TIME** | 00:00:00 (hour : minute : second) |
| **RTC_CALM** | 00/00/00 (year/month/day) |
| **RTC_TALM** | 00:00:00 (hour : minute : second) |
| **RTC_CLKFMT** | 1 (24 hr mode) |
| **RTC_WEEKDAY** | 6 (Saturday) |
| **RTC_INTEN** | 0 |
| **RTC_INTSTS** | 0 |
| **RTC_LEAPYEAR** | 0 |
| **RTC_TICK** | 0 |
| **RTC_POWCTL** | 50000 |

4. In RTC_CAL and RTC_TALM, only 2 BCD digits are used to express “year”. The 2 BCD digits of xy means 20xy, rather than 19xy or 21xy.
5. Example of 12-Hour Time Setting

   If current RTC time is PM12:59:30 in 12-Hour Time Scale mode, the RTC_TIME setting as:

   RTC_TIME[21:16]: 0x32 (0x12+0x20)

   TENHR (RTC_TIME[21:20]) is 0x3, HR (RTC_TIME[19:16]) is 0x2.

   RTC_TIME[14:8]: 0x59

   TENMIN (RTC_TIME[14:12]) is 0x5, MIN (RTC_TIME[11:8]) is 0x9.

   RTC_TIME[6:0]: 0x30

   TENSEC (RTC_TIME[6:4]) is 0x3, SEC (RTC_TIME[3:0]) is 0x0.

### 6.9.6 System Power Control Flow

#### *6.9.6.1 Normal system Power Control Flow*

The state machine of power On/Off Control is as follows.

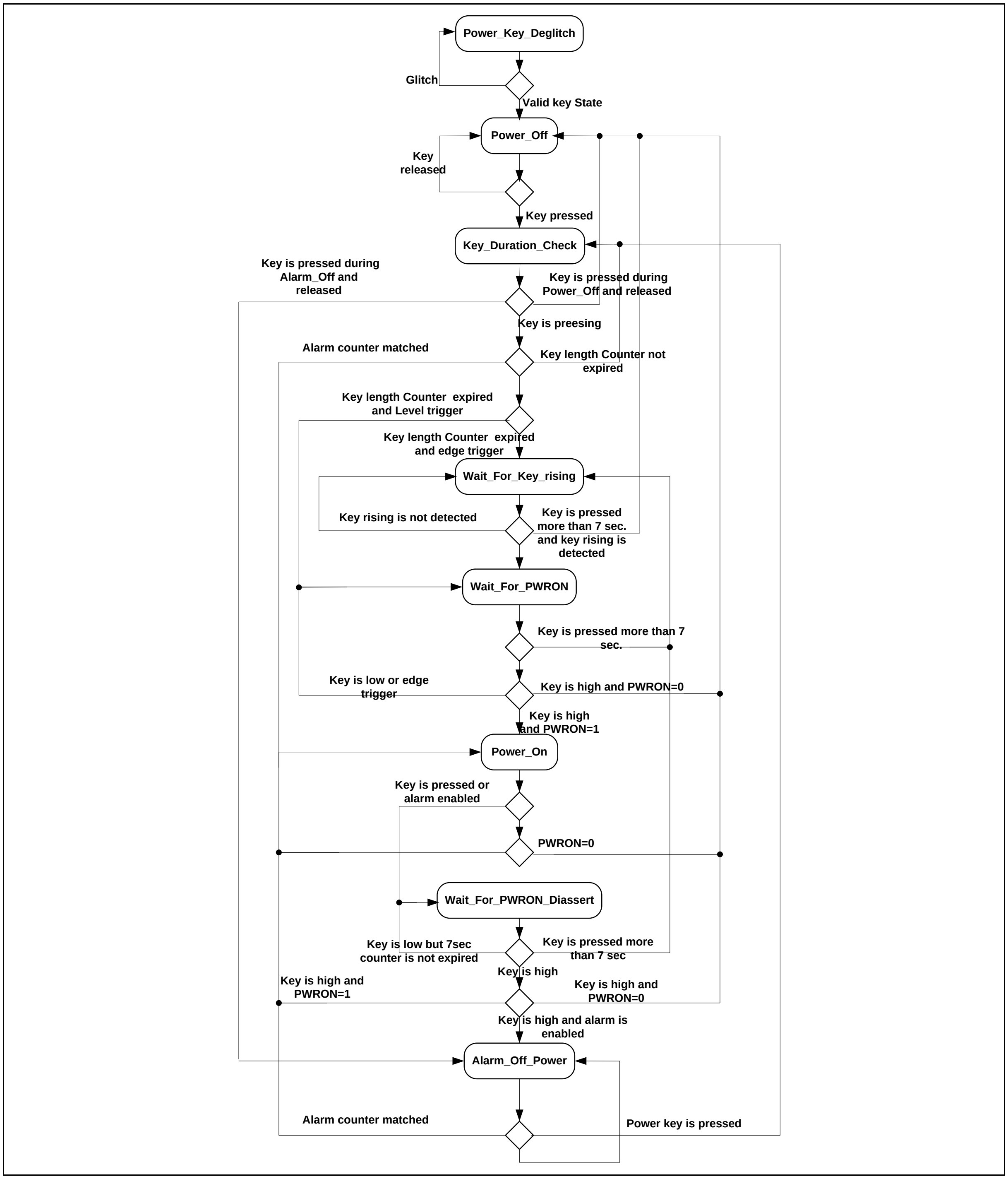

Power_Key_Deglitch
Glitch
Valid key State
Power_Off
Key released
Key pressed
Key_Duration_Check
Key is pressed during Alarm_Off and released
Key is pressed during Power_Off and released
Key is preesing
Alarm counter matched
Key length Counter not expired
Key length Counter expired and Level trigger
Key length Counter expired and edge trigger
Wait_For_Key_rising
Key rising is not detected
Key is pressed more than 7 sec. and key rising is detected
Wait_For_PWRON
Key is pressed more than 7 sec.
Key is low or edge trigger
Key is high and PWRON=0
Key is high and PWRON=1
Power_On
Key is pressed or alarm enabled
PWRON=0
Wait_For_PWRON_Diassert
Key is low but 7sec counter is not expired
Key is pressed more than 7 sec
Key is high
Key is high and PWRON=1
Key is high and PWRON=0
Key is high and alarm is enabled
Alarm_Off_Power
Alarm counter matched
Power key is pressed

Figure 6.9-2 System Power Control Flow Chart

### *6.9.6.2 Force system Power Off Control Flow*

The RTC supports a hardware automatic power off function and a software power off function. For hardware power off function, it can be enable and disable in POWOFFEN (RTC_POWCTL[2]) and the user presses the power button for a few seconds to power off system. The time to press power button to power off is configured in POWOFFT (RTC_POWCTL[19:16]).

The relationship between the setting of POWOFFT (RTC_pOWCTL[19:16]) and the key-pressed period to power off is as follows.

Key_Pressed_Period_To_Power_Off=(POWOFFT+3) sec.

The timing of the hardware power on function is as follows.

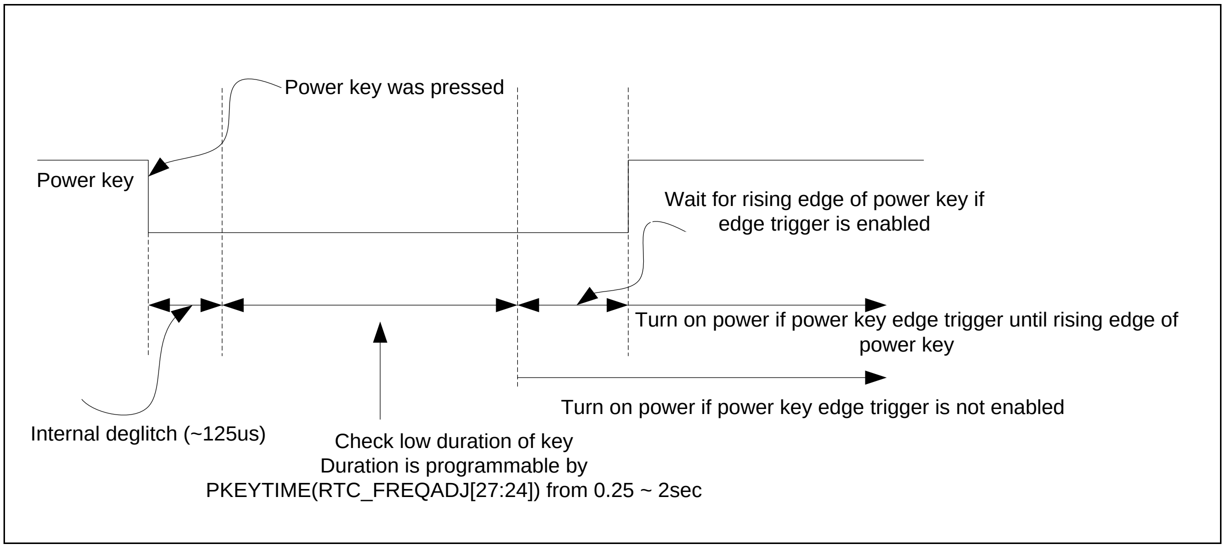

Power key was pressed
Power key
Wait for rising edge of power key if edge trigger is enabled
Turn on power if power key edge trigger until rising edge of power key
Turn on power if power key edge trigger is not enabled
Internal deglitch (~125us)
Check low duration of key
Duration is programmable by
PKEYTIME(RTC_FREQADJ[27:24]) from 0.25 ~ 2sec

Figure 6.9-3 Power Key Detection Mechanism

### 6.9.7 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC Base Address:**<br>**RTC_BA = 0x400E_7000** | | | | |
| **RTC_INIT** | RTC_BA+0x000 | R/W | RTC Initiation Register | 0x0000_0000 |
| **RTC_RWEN** | RTC_BA+0x004 | R/W | RTC Access Enable Register | 0x0000_0000 |
| **RTC_FREQADJ** | RTC_BA+0x008 | R/W | RTC Frequency Compensation Register | 0x007F_FF00 |
| **RTC_TIME** | RTC_BA+0x00C | R/W | RTC Time Loading Register | 0x0000_0000 |
| **RTC_CAL** | RTC_BA+0x010 | R/W | RTC Calendar Loading Register | 0x0005_0101 |
| **RTC_CLKFMT** | RTC_BA+0x014 | R/W | RTC Time Scale Selection Register | 0x0000_0001 |
| **RTC_WEEKDAY** | RTC_BA+0x018 | R/W | RTC Day of the Week Register | 0x0000_0006 |
| **RTC_TALM** | RTC_BA+0x01C | R/W | RTC Time Alarm Register | 0x0000_0000 |
| **RTC_CALM** | RTC_BA+0x020 | R/W | RTC Calendar Alarm Register | 0x0000_0000 |
| **RTC_LEAPYEAR** | RTC_BA+0x024 | R | RTC Leap Year Indication Register | 0x0000_0000 |
| **RTC_INTEN** | RTC_BA+0x028 | R/W | RTC Interrupt Enable Register | 0x0000_0000 |
| **RTC_INTSTS** | RTC_BA+0x02C | R/W | RTC Interrupt Indication Register | 0x0000_0000 |
| **RTC_TICK** | RTC_BA+0x030 | R/W | RTC Time Tick Register | 0x0000_0000 |
| **RTC_POWCTL** | RTC_BA+0x034 | R/W | RTC Power Time-out Register | 0x0005_0000 |
| **RTC_SET** | RTC_BA+0x038 | R/W | RTC Setting Register | 0x0000_0000 |
| **RTC_CLKSRC** | RTC_BA+0x03C | R/W | RC Oscillator Setting Register | 0x0000_0001 |
| **RTC_CALCNT** | RTC_BA+0x040 | R | RC Oscillator Calibration Register | 0x0000_0000 |
| **RTC_SYNC** | RTC_BA+0x044 | R | RTC Register Write Complete | 0x0000_0000 |
| **RTC_SPR0** | RTC_BA+0x054 | R/W | RTC Spare Register 0 | 0x0000_0000 |
| **RTC_SPR1** | RTC_BA+0x058 | R/W | RTC Spare Register 1 | 0x0000_0000 |
| **RTC_SPR2** | RTC_BA+0x05C | R/W | RTC Spare Register 2 | 0x0000_0000 |
| **RTC_SPR3** | RTC_BA+0x060 | R/W | RTC Spare Register 3 | 0x0000_0000 |
| **RTC_SPR4** | RTC_BA+0x064 | R/W | RTC Spare Register 4 | 0x0000_0000 |
| **RTC_SPR5** | RTC_BA+0x068 | R/W | RTC Spare Register 5 | 0x0000_0000 |
| **RTC_SPR6** | RTC_BA+0x06C | R/W | RTC Spare Register 6 | 0x0000_0000 |
| **RTC_SPR7** | RTC_BA+0x070 | R/W | RTC Spare Register 7 | 0x0000_0000 |

### 6.9.8 Register Description

**RTC Initiation Register (RTC_INIT)**

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_INIT** | RTC_BA+0x000 | R/W | RTC Initiation Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:1] | **INIT/STS** | **RTC Initiation (While Writing)**<br>When RTC block is powered on, RTC is at reset state. User has to write a number (0x a5eb1357) to INIT to make RTC exit from reset state. Once the INIT is written as 0xa5eb1357, the RTC will be in un-reset state permanently.<br>The INIT is a write-only field and read value will be always “0”.<br>**RTC Internal Status (While Reading)**<br>[31:8]: INIT[31:8]<br>[7:5]: RTC internal state machine of key detection<br>[4]: Status of power key, 0:pressed and 1:released<br>[3]: Status of power off request pwr_key_off<br>[2]: Level shifter reset<br>[1]: Level shifter enable |
| [0] | **ACTIVE** | **RTC Active Status (Read Only)**<br>0 = RTC is at reset state.<br>1 = RTC is at normal active state. |

### RTC Access Enable Register (RTC_RWEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_RWEN** | RTC_BA+0x004 | R/W | RTC Access Enable Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [16] | **RWENF** | **RTC Register Access Enable Flag (Read Only)**<br>0 = RTC register access Disabled.<br>1 = RTC register access Enabled.<br>**Note:** This bit will be set after RTC_RWEN[15:0] register is load a 0xA965, and it will be cleared when RTC_RWEN[15:0] is not 0xA965. |
| [15:0] | **RWEN** | **RTC Register Access Enable Password (R/W)**<br>0xA965 = Access Password.<br>Others = Access Disabled. |

### RTC Frequency Compensation Register (RTC_FREQADJ)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_FREQADJ** | RTC_BA+0x008 | R/W | RTC Frequency Compensation Register | 0x007F_FF00 |

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

| Bits | Description | |
|---|---|---|
| [31] | **ADJTRG** | **RTC Clock Calibration Control**<br>This bit will be kept at “High” while the calibration is ongoing and cleared to “Low” automatically while the calibration is done and the content of RTC_CALCNT register is valid calibration flow as follows.<br>0 = RTC Clock calibration mechanism Disabled.<br>1 = RTC Clock calibration mechanism Enabled. |
| [30] | **Reserved** | Reserved. |
| [29:28] | **Reserved** | Reserved. |
| [27:24] | **PKEYTIME** | **Minimum Duration That Power Key Must Be Pressed to Turn on Core Power**<br>Minimum power key duration = 0.25*(PKEYTIME+1) sec. |
| [23:8] | **INTEGER** | **Integer Part**<br>Real Oscillator Frequency = (Integer part of detected RTC clock rate -1). |
| [7:6] | **Reserved** | Reserved. |
| [5:0] | **FRACTION** | **Fraction Part**<br>Formula = (fraction part of detected RTC clock rate) x 60-1.<br>Digit in RTC_FREQADJ must be expressed as hexadecimal number. |

### RTC Time Loading Register (RTC_TIME)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_TIME** | RTC_BA+0x00C | R/W | RTC Time Loading Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:22] | **Reserved** | Reserved. |
| [21:20] | **TENHR** | 10-Hour Time Digit |
| [19:16] | **HR** | 1-Hour Time Digit |
| [15] | **Reserved** | Reserved. |
| [14:12] | **TENMIN** | 10-Min Time Digit |
| [11:8] | **MIN** | 1-Min Time Digit |
| [7] | **Reserved** | Reserved. |
| [6:4] | **TENSEC** | 10-Sec Time Digit |
| [3:0] | **SEC** | 1-Sec Time Digit |

**Note:** RTC_TIME is a BCD digit counter and RTC will not check loaded data.

### RTC Calendar Loading Register (RTC_CAL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_CAL** | RTC_BA+0x010 | R/W | RTC Calendar Loading Register | 0x0005_0101 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:20] | **TENYEAR** | 10-Year Calendar Digit |
| [19:16] | **YEAR** | 1-Year Calendar Digit |
| [15:13] | **Reserved** | Reserved. |
| [12] | **TENMON** | 10-Month Calendar Digit |
| [11:8] | **MON** | 1-Month Calendar Digit |
| [7:6] | **Reserved** | Reserved. |
| [5:4] | **TENDAY** | 10-Day Calendar Digit |
| [3:0] | **DAY** | 1-Day Calendar Digit |

**Note:** RTC_CAL is a BCD digit counter and RTC will not check loaded data.

### RTC Time Scale Selection Register (RTC_CLKFMT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_CLKFMT** | RTC_BA+0x014 | R/W | RTC Time Scale Selection Register | 0x0000_0001 |

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

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **24HEN** | **24-hour / 12-hour Mode Selection**<br>Indicate that RTC_TIME and RTC_TALM are in 24-hour mode or 12-hour mode.<br>0 = 12-hour time scale with AM and PM indication selected.<br>1 = 24-hour time scale selected. |

### RTC Day of the Week Register (RTC_WEEKDAY)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_WEEKDAY** | RTC_BA+0x018 | R/W | RTC Day of the Week Register | 0x0000_0006 |

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

| Bits | Description | |
|---|---|---|
| [31:3] | **Reserved** | Reserved. |
| [2:0] | **WEEKDAY** | **Day of the Week Register**<br>0 = Sunday.<br>1 = Monday.<br>2 = Tuesday.<br>3 = Wednesday.<br>4 = Thursday.<br>5 = Friday.<br>6 = Saturday. |

## RTC Time Alarm Register (RTC_TALM)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_TALM** | RTC_BA+0x01C | R/W | RTC Time Alarm Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23] | **MSKHR** | **Mask Alarm by Hour**<br>0 = Activate.<br>1 = Mask. |
| [22] | **Reserved** | Reserved. |
| [21:20] | **TENHR** | **10-hour Time Digit of Alarm Setting (0-2)**<br>When RTC runs as 12-hour time scale mode, RTC_TIME[21] (the high bit of TENHR[1:0]) means AM/PM indication. (If RTC_TIME[21] is 1, it indicates PM time message.) |
| [19:16] | **HR** | 1-Hour Time Digit of Alarm Setting (0-9) |
| [15] | **MSKMIN** | **Mask Alarm by Minute**<br>0 = Activate.<br>1 = Mask. |
| [14:12] | **TENMIN** | 10-Min Time Digit of Alarm Setting (0-5) |
| [11:8] | **MIN** | 1-Min Time Digit of Alarm Setting (0-9) |
| [7] | **MSKSEC** | **Mask Alarm by Second**<br>0 =Activate.<br>1 = Mask. |
| [6:4] | **TENSEC** | 10-Sec Time Digit of Alarm Setting (0-5) |
| [3:0] | **SEC** | 1-Sec Time Digit of Alarm Setting (0-9) |

**Note:** RTC_TALM is a BCD digit counter and RTC will not check loaded data.

### RTC Calendar Alarm Register (RTC_CALM)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_CALM** | RTC_BA+0x020 | R/W | RTC Calendar Alarm Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MSKWEEKDAY | WEEKDAY | | | Reserved | | | MSKYEAR |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| TENYEAR | | | | YEAR | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| MSKMON | Reserved | | TENMON | MON | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| MSKDAY | Reserved | TENDAY | | DAY | | | |

| Bits | Description | |
|---|---|---|
| [31] | **MSKWEEKDAY** | **Mask Alarm by Week Day**<br>0 =Activate.<br>1 =Mask. |
| [30:28] | **WEEKDAY** | Week Day Alarm Digit |
| [27:25] | **Reserved** | Reserved. |
| [24] | **MSKYEAR** | **Mask Alarm by Year**<br>0 = Activate.<br>1 = Mask. |
| [23:20] | **TENYEAR** | 10-Year Calendar Digit of Alarm Setting (0-9) |
| [19:16] | **YEAR** | 1-Year Calendar Digit of Alarm Setting (0-9) |
| [15] | **MSKMON** | **Mask Alarm by Month**<br>0 = Activate.<br>1 = Mask. |
| [14:13] | **Reserved** | Reserved. |
| [12] | **TENMON** | 10-Month Calendar Digit of Alarm Setting (0-1) |
| [11:8] | **MON** | 1-Month Calendar Digit of Alarm Setting (0-9) |
| [7] | **MSKDAY** | **Mask Alarm by Day**<br>0 = Activate.<br>1 = Mask. |
| [6] | **Reserved** | Reserved. |
| [5:4] | **TENDAY** | 10-Day Calendar Digit of Alarm Setting (0-3) |
| [3:0] | **DAY** | 1-Day Calendar Digit of Alarm Setting (0-9) |

**Note:**

1. RTC_CALM is a BCD digit counter and RTC will not check loaded data.
2. Alarm will be disabled automatically while all alarm bits of RTC_CALM and RTC_TALM are masked

### RTC Leap Year Indication Register (RTC_LEAPYEAR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_LEAPYEAR** | RTC_BA+0x024 | R | RTC Leap Year Indication Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **LEAPYEAR** | **Leap Year Indication Register (Read Only)**<br>0 = This year is not a leap year.<br>1 = This year is leap year. |

### RTC Interrupt Enable Register (RTC_INTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_INTEN** | RTC_BA+0x028 | R/W | RTC Interrupt Enable Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [3] | **RALMIEN** | **Relative Alarm Interrupt Enable Control**<br>0 = RTC Relative Alarm Interrupt Disabled.<br>1 = RTC Relative Alarm Interrupt Enabled. |
| [2] | **PKEYIEN** | **Power Switch Interrupt Enable Control**<br>0 = Power Switch Be Pressed Interrupt Disabled.<br>1 = Power Switch Be Pressed Interrupt Enabled. |
| [1] | **TICKIEN** | **Time Tick Interrupt Enable Control**<br>0 = RTC Time Tick Interrupt and counter Disabled.<br>1 = RTC Time Tick Interrupt and counter Enabled. |
| [0] | **ALMIEN** | **Alarm Interrupt Enable Control**<br>0 = RTC Alarm Interrupt Disabled.<br>1 = RTC Alarm Interrupt Enabled. |

### RTC Interrupt Indication Register (RTC_INTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_INTSTS** | RTC_BA+0x02C | R/W | RTC Interrupt Indication Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [3] | **RELALMIF** | **RTC Relative Alarm Interrupt Indication**<br>0 = Relative alarm interrupt never occurred.<br>1 = Relative time counter and calendar counter have counted to a specified time recorded in RTC_TALM and RTC_CALM. RTC alarm interrupt has been activated.<br>**Note:** Software can also clear this bit after RTC interrupt has occurred |
| [2] | **POWKEYIF** | **Power Switch Interrupt Flag**<br>When RTC detect power key (RTC_nRWAKE) is pressed , the POWKEYIF (RTC_INTSYS[2]) is set to 1<br>0 = The power switch interrupt never occurred.<br>1 = The power switch has been activated.<br>**Note:** Software can also clear this bit after RTC interrupt has occurred |
| [1] | **TICKIF** | **RTC Time Tick Interrupt Flag**<br>When RTC time tick happened, this bit will be set to 1 and an interrupt will be generated if RTC Tick Interrupt enabled TICKIEN (RTC_INTEN[1]) is set to 1. Chip will also be woken up if RTC Tick Interrupt is enabled and this bit is set to 1 when chip is running at Power-down mode.<br>0 = Tick condition does not occur.<br>1 = Tick condition occur.<br>**Note:** Write 1 to clear to clear this bit. |
| [0] | **ALMIF** | **RTC Alarm Interrupt Flag**<br>When RTC time counters RTC_TIME and RTC_CAL match the alarm setting time registers RTC_TALM and RTC_CALM, this bit will be set to 1 and an interrupt will be generated if RTC Alarm Interrupt enabled ALMIEN (RTC_INTEN[0]) is set to 1. Chip will be woken up if RTC Alarm Interrupt is enabled when chip is at Power-down mode.<br>0 = Alarm condition is not matched.<br>1 = Alarm condition is matched.<br>**Note:** Write 1 to clear this bit. |

### RTC Time Tick Register (RTC_TICK)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_TICK** | RTC_BA+0x030 | R/W | RTC Time Tick Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:3] | **Reserved** | Reserved. |
| [2:0] | **TICKSEL** | **Time Tick Register**<br>These bits are used to select RTC time tick period for Periodic Time Tick Interrupt request.<br>000 = Time tick is 1 second.<br>001 = Time tick is 1/2 second.<br>010 = Time tick is 1/4 second.<br>011 = Time tick is 1/8 second.<br>100 = Time tick is 1/16 second.<br>101 = Time tick is 1/32 second.<br>110 = Time tick is 1/64 second.<br>111 = Time tick is 1/28 second. |

### RTC Power Time-out Register (RTC_POWCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_POWCTL** | RTC_BA+0x034 | R/W | RTC Power Time-out Register | 0x0005_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| RALMTIME | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| RALMTIME | | | | POWOFFT | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| POWKEY | Reserved | EDGE_TRIG | RALMIEN | ALMIEN | POWOFFEN | SWPOWOFF | POWEN |

| Bits | Description | |
|---|---|---|
| [31:20] | **RALMTIME** | **Relative Time Alarm Period (Second Unit)**<br>Indicates the period of the relative time alarm. Its maximum value is 1800.<br>When RALMIEN (RTC_INTEN[3]) = 0 , it will be cleared to 0. |
| [19:16] | **POWOFFT** | **Power Clear Period**<br>Indicates that the period of the power core will be cleared after the power key is pressed. Its time scalar is one second so that the default is 5 second. |
| [15:8] | **Reserved** | Reserved |
| [7] | **POWKEY** | **Power Key Status**<br>0 = The power key is pressed to low.<br>1 = The power key status is high. |
| [6] | **Reserved** | Reserved. |
| [5] | **EDGE_TRIG** | **Power Key Trigger Mode**<br>0 = LEVEL TRIGGER, RTC is powered on while power key is pressed longer programmed duration.<br>1 = EDGE TRIGE, RTC is powered on while power key is pressed longer than programmed duration and then released. |
| [4] | **RALMIEN** | **Relative Time Alarm**<br>0 = The relative time alarm control Disabled.<br>1 = The relative time alarm control Enabled. |
| [3] | **ALMIEN** | **Normal Time Alarm**<br>0 = Normal time alarm control Disabled.<br>1 = Normal time alarm control Enabled. |

| | | |
|---|---|---|
| [2] | **POWOFFEN** | **Hardware Power Clear Enable Control**<br>0 = The RTC_RPWR pin will not be influenced by the pressed time of power key.<br>1 = The RTC_RPWR pin will be cleared to low when the power key is pressed over the POWOFFT second. |
| [1] | **SWPOWOFF** | **Software Core Power Disable Control**<br>If the power key is pressed, the RTC_RPWR pin can be cleared by setting this bit and this can be cleared to 0 when the pressed power key, RTC_RPWR is released. If the power is not pressed, it is not used to set this bit.<br>1 = Force the RTC_RPWR to low. |
| [0] | **POWEN** | **Power ON**<br>RTC_RPWR will change to high state when POWEN value change from 0 to 1.<br>**Note:** The following conditions will make RTC_RPWR low:<br>• Set POWEN bit to 0<br>• POWOFFEN is set to 1 and the power key is pressed over the period of POWOFFT.<br>This bit can be read back after the RTC enable is active. |

### RTC Setting Register (RTC_SET)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_SET** | RTC_BA+0x038 | R/W | RTC Setting Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:5] | **Reserved** | Reserved. |
| [4] | **XOUTDAT** | **X32_OUT PAD Status**<br>Input signal when IOMSEL = 0 (Read Only). |
| [3] | **XININDAT** | **X32_IN PAD Status**<br>Input signal  when IOMSEL = 0 (Read Only). |
| [2] | **IOMSEL** | **X32_IN and X32_OUT PAD Digital Input Mode Control**<br>0 = Digital input mode.<br>1 = Crystal mode (default value). |
| [1] | **CBEN** | **32768 Hz (LXT) Crystal Control**<br>0 = Crystal Disabled.<br>1 = Crystal Enabled. |
| [0] | **Reserved** | Reserved |

### RC Oscillator Setting Register RTC_CLKSRC)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_CLKSRC** | RTC_BA+0x03C | R/W | RC Oscillator Setting Register | 0x0000_0001 |

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

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **CKSRC** | **Internal RC Oscillator Control**<br>0 = Internal RC oscillator Disabled.<br>1 = Internal RC oscillator Enabled. |

### RC Oscillator Calibration Register (RTC_CALCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_CALCNT** | RTC_BA+0x040 | R | RC Oscillator Calibration Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **CALCNT** | **Cycle Number of PCLK During 1Hz**<br>That is generated by dividing RTC Clock. This number can be used to deduct the real clock rate of RTC clock. |

### RTC Register Complete Register (RTC_SYNC)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_SYNC** | RTC_BA+0x044 | R | RTC Register Write Complete | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **SYNC** | **Polling the Flag to Detect RTC Register Write Complete**<br>0 = Register cannot be written.<br>1 = Register can be written because write complete. |

### RTC Spare Register 0~7

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **RTC_SPR0** | RTC_BA+0x054 | R/W | RTC Spare Register 0 | 0x0000_0000 |
| **RTC_SPR1** | RTC_BA+0x058 | R/W | RTC Spare Register 1 | 0x0000_0000 |
| **RTC_SPR2** | RTC_BA+0x05C | R/W | RTC Spare Register 2 | 0x0000_0000 |
| **RTC_SPR3** | RTC_BA+0x060 | R/W | RTC Spare Register 3 | 0x0000_0000 |
| **RTC_SPR4** | RTC_BA+0x064 | R/W | RTC Spare Register 4 | 0x0000_0000 |
| **RTC_SPR5** | RTC_BA+0x068 | R/W | RTC Spare Register 5 | 0x0000_0000 |
| **RTC_SPR6** | RTC_BA+0x06C | R/W | RTC Spare Register 6 | 0x0000_0000 |
| **RTC_SPR7** | RTC_BA+0x070 | R/W | RTC Spare Register 7 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **RTC_SPRn** | **RTC Spare Register**<br>n = 0~7. |

## 6.10 UART Interface Controller (UART)

### 6.10.1 Overview

The NUC505 series provides three channels of Universal Asynchronous Receiver/Transmitters (UART). The UART Controller performs a serial-to-parallel conversion on data received from the peripheral and a parallel-to-serial conversion on data transmitted from the CPU. Each UART Controller channel supports ten types of interrupts. The UART controller also supports IrDA SIR, RS-485, auto-flow control function and auto-baud rate measuring function.

### 6.10.2 Features

- Full-duplex asynchronous communications
- Separates receive and transmit 16/16(UART0) / 64/64(UART1 and UART2) bytes entry FIFO for data payloads
- Supports hardware auto-flow control ( nCTS and nRTS) with UART1 and UART2
- Programmable receiver buffer trigger level
- Supports programmable baud rate generator for each channel individually
- Supports nCTS and data wake-up function
- Supports 8-bit receiver buffer time-out detection function
- Programmable transmitting data delay time between the last stop and the next start bit by setting DLY (UART_TOUT [15:8])
- Supports Auto-Baud Rate measurement
- Supports break error, frame error, parity error and receive/transmit buffer overflow detection function
- Fully programmable serial-interface features
  - Programmable number of data bit, 5-, 6-, 7-, 8- bit character
  - Programmable parity bit, even, odd, no parity or stick parity bit generation and detection
  - Programmable stop bit, 1, 1.5, or 2 stop bit generation
- Supports IrDA SIR function mode
  - Supports for 3/16 bit duration for normal mode
- Supports LIN function mode (Only UART1 /UART2 with LIN function)
  - Supports LIN Master/Slave mode
  - Supports programmable break generation function for transmitter
  - Supports break detection function for receiver
- Supports RS-485 function mode
  - Supports RS-485 9-bit mode
  - Supports hardware or software enables to program nRTS pin to control RS-485 transmission direction

### 6.10.3 Block Diagram

The UART clock control and block diagram are shown in Figure 6.10-1 and Figure 6.10-2 respectively.

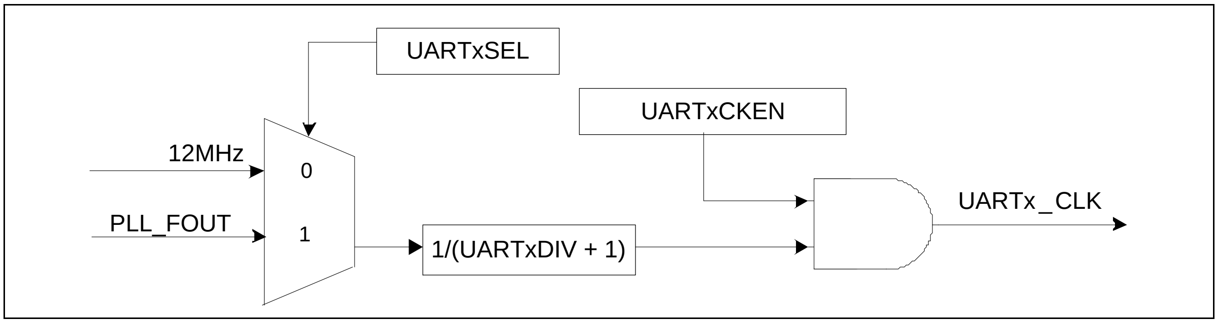


Figure 6.10-1 UART Clock Control Diagram

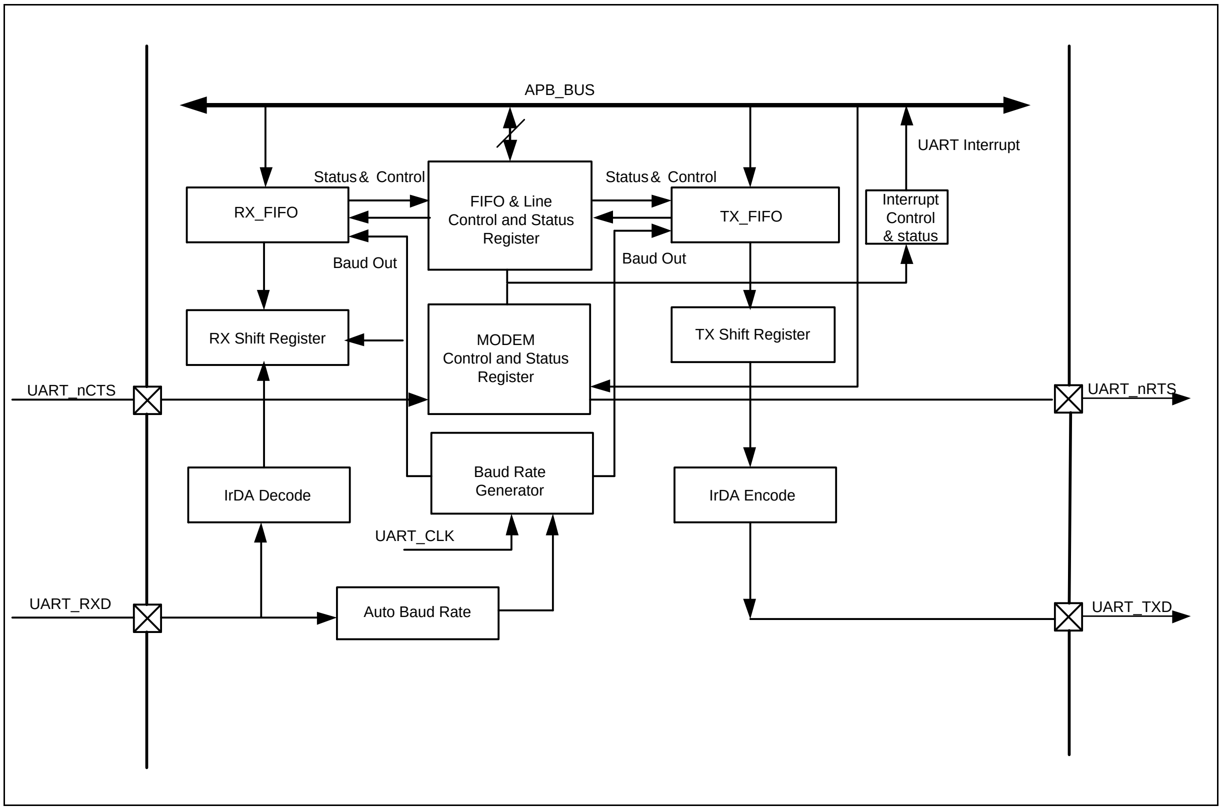


Figure 6.10-2 UART Block Diagram

Each block is described in detail as follows:

**TX_FIFO**
The transmitter is buffered with a 16 (UART0) / 64 (UART1 and UART2) bytes FIFO to reduce the number of interrupts presented to the CPU.

**RX_FIFO**
The receiver is buffered with a 16 (UART0) / 64 (UART1 and UART2) bytes FIFO (plus three error bits, BIF (UART_FIFOSTS[6]), FEF (UART_FIFOSTS[5]), PEF (UART_FIFOSTS[4])) to reduce the number of interrupts presented to the CPU.

**TX Shift Register**
This block is responsible for shifting out the transmitting data serially.

**RX Shift Register**
This block is responsible for shifting in the receiving data serially.

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

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

**IrDA Encode**
This block is IrDA encoding control block.

**IrDA Decode**
This block is IrDA decoding control block.

**FIFO & Line Control and Status Register**

This field is register set that including the FIFO control registers (UART_FIFO), FIFO status registers (UART_FIFOSTS), and line control register (UART_LINE) for transmitter and receiver. The time-out control register (UART_TOUT) identifies the condition of time-out interrupt.

**Auto-Baud Rate Measurement**

This block is responsible for auto-baud rate measurement.

**Interrupt Control and Status Register**

There are ten types of interrupts, transmitter FIFO empty interrupt (THERIF), receiver threshold level reaching interrupt (RDAIF), receive line status interrupt (parity error or framing error or break interrupt) (RLSIF), time-out interrupt (RXTOINT), Buffer error interrupt (BUFERRINT), LIN bus interrupt (LININT), data wake-up interrupt, nCTS wake-up interrupt and auto-baud rate detection finish or auto-baud rate detection counter overflow interrupt. Interrupt enable register (UART_INTEN) enable or disable the responding interrupt and interrupt status register (UART_INTSTS) identifying the occurrence of the responding interrupt.

### 6.10.4 Basic Configuration

The UART Controller function pins are configured in SYS_GPA_MFPH, SYS_GPB_MFPL and SYS_GPB_MFPH Multi-function Registers.

The UART Controller clock are enabled in UART0CKEN (CLK_APBCLK[11]) for UART0, UART1CKEN(CLK_APBCLK[12]) for UART1 and UART2CKEN(CLK_APBCLK[13] for UART2.

The UART Controller clock source is selected by UART0SEL (CLK_CLKDIV3[4]) for UART0, UART1SEL (CLK_CLKDIV3[12]) for UART1 and UART2SEL (CLK_CLKDIV3[20]) for UART2.

The UART Controller clock pre-scale is determined by UART0DIV (CLK_CLKDIV3[3:0]) for UART0, UART1DIV (CLK_CLKDIV3[11:8]) for UART1 and UART2DIV (CLK_CLKDIV3[19:16]) for UART2.

The UART Interface Controller Pin description is shown as follows:

| Pin | Type | Description |
|---|---|---|
| UART_TXD | Output | UART transmit |
| UART_RXD | Input | UART receive |
| UART_nCTS | Input | UART modem clear to send |
| UART_nRTS | Output | UART modem request to send |

Table 6.10-1 UART Interface Controller Pin

### 6.10.5 Functional Description

The UART Controller supports four function modes including UART, IrDA, LIN and RS-485 mode. User can select a function by setting the UART_FUNCSEL register. The four function modes will be described in following section.

#### *6.10.5.1 UART Controller Baud Rate Generator*

The UART Controller includes a programmable baud rate generator capable of dividing clock input by divisors to produce the serial clock that transmitter and receiver need. The following tables list the UART baud rate equations in the various conditions and UART baud rate parameter settings. There is no error for the baud rate results calculated through the baud rate parameter and register setting below. In IrDA function mode, the baud rate generator must be set in mode 0. More detail register description is shown in UART_BAUD register. There are three setting mode. Mode 0 is set by UART_BAUD[29:28] with 00. Mode 1 is set by UART_BAUD[29:28] with 10. Mode 2 is set by UART_BAUD[29:28] with 11.

| Mode | BAUDM1 | BAUDM0 | Baud Rate Equation |
|---|---|---|---|
| Mode 0 | 0 | 0 | UART_CLK / [16 * (BRD+2)] |
| Mode 1 | 1 | 0 | UART_CLK / [(EDIVM1+1) * (BRD+2)], EDIVM1 must >= 8 |
| Mode 2 | 1 | 1 | UART_CLK / (BRD+2).<br>If UART_CLK <= 3*PCLK, BRD must >= 9.<br>If UART_CLK > 3*PCLK, BRD must >= 3*N – 1.<br>N is the smallest integer larger than or equal to the ratio of UART_CLK /PCLK.<br>For example,<br>if 3*PCLK < UART_CLK =< 4*PCLK, BRD must >=11. |

| | | | |
|---|---|---|---|
| | | | if 4*PCLK < UART_CLK =< 5*PCLK, BRD must >=14. |

Table 6.10-2 UART Controller Baud Rate Equation Table

### *6.10.5.2 UART Controller Auto-Baud Rate Function Mode*

Auto-Baud Rate function can measure baud rate of receiving data from UART RX pin automatically. When the Auto-Baud Rate measurement is finished, the measuring baud rate is loaded to BRD (UART_BAUD[15:0]). Both of the BAUDM1 (UART_BAUD[29]) and BAUDM0 (UART_BAUD[28]) are set to 1 automatically. UART RX data from Start bit to 1st rising edge time is set ABRDBITS (UART_ALTCTL[20:19]) in Auto-Baud Rate function detection frame.

Setting ABRDEN (UART_ALTCTL[18]) is to enable auto-baud rate function. In beginning stage, the UART RX is kept at 1. Once falling edge is detected, START bit is received. The auto-baud rate counter is reset and starts counting. The auto-baud rate counter will be stop when the 1 st rising edge is detected. Then, auto-baud rate counter value divided by ABRDBITS (UART_ALTCTL[20:19]) is loaded to BRD(UART_BAUD[15:0]) automatically. ABRDEN (UART_ALTCTL[18]) is cleared. Once the auto-baud rate measurement is finished, the ABRDIF (UART_FIFOSTS[1]) is set. When auto-baud rate counter is overflow, ABRTOIF (UART_FIFOSTS[2]) is set. If the ABRIEN (UART_INTEN[18]) is enabled, ABRDIF(UART_FIFOSTS[1]) or (UART_FIFOSTS[2])cause the auto-baud rate interrupt ABRIF(UART_ALTCTL[17]) is generated.

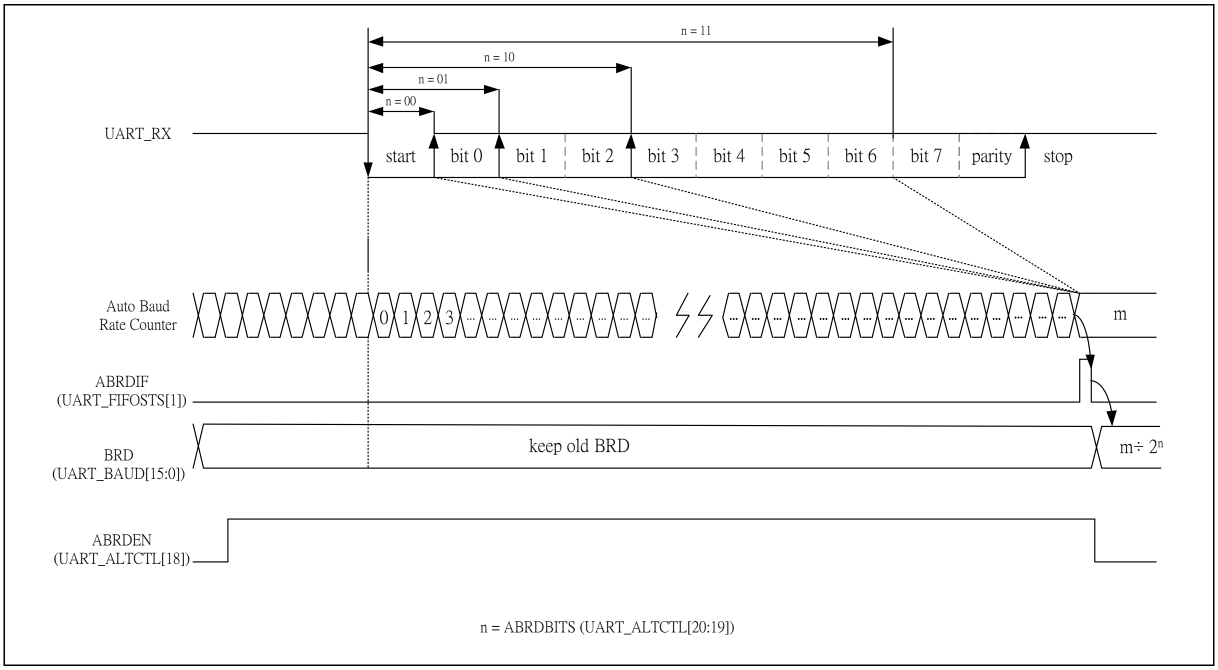


Figure 6.10-3 Auto-Baud Rate Measurement

**Programming Sequence Example:**

1. Program ABRDBITS (UART_ALTCTL[20:19]) to determines UART RX data from Start bit to 1st rising edge BIT time.
2. Set ABRIEN (UART_INTEN[18]) to enable auto-baud rate function interrupt.

3. Set ABRDEN (UART_ALTCTL[18]) to enable auto-baud rate function.
4. ABRDIF (UART_FIFOSTS[1]) is set, the auto-baud rate measurement is finished.
5. Operate UART transmit and receive action.
6. ABRDTOIF (UART_FIFOSTS[2]) is set, if auto-baud rate counter is overflow.
7. Go to Step 2.

#### *6.10.5.3 UART Controller Transmit Delay Time Value*

The UART Controller programs DLY (UART_TOUT [15:8]) to control the transfer delay time between the last stop bit and next start bit in transmission. The unit is baud. The operation is shown in Figure 6.10-4

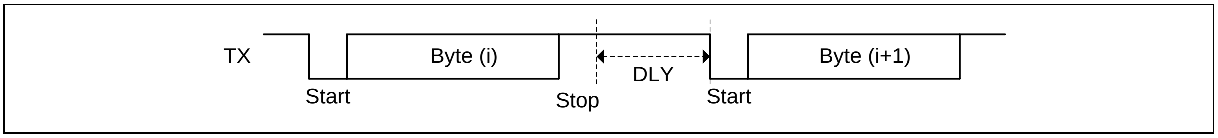


Figure 6.10-4 Transmit Delay Time Operation

#### *6.10.5.4 UART Controller FIFO Control and Status*

The UART Controller is built-in with a 16 (UART0) / 64 (UART1 and UART2) bytes transmitter FIFO (TX_FIFO) and a 16 (UART0) / 64 (UART1 and UART2) bytes receiver FIFO (RX_FIFO) that reduces the number of interrupts presented to the CPU. The CPU can read the status of the UART at any time during operation. The reported status information includes condition of the transfer operations being performed by the UART, as well as 3 error conditions (parity error, framing error, break interrupt) occur if receiving data has parity, frame or break error. UART, IrDA, LIN and RS-485 mode support FIFO control and status function.

#### *6.10.5.5 UART Controller Wake-up Function*

The UART controller supports wake-up system function. The wake-up function includes nCTS and data wake-up function. When the system is in Power-down, the UART can wake-up system by nCTS pin or incoming data. When incoming data wakes system up, the incoming data will be received and stored in FIFO and controller will clear the WKDATIEN (UART_INTEN [10]) automatically. However, the first byte of receiving data is lost. The data is received after second bytes. The following diagram demonstrates the wake-up function.

**nCTS Wake-Up Case 1 (nCTS transition from low to high)**

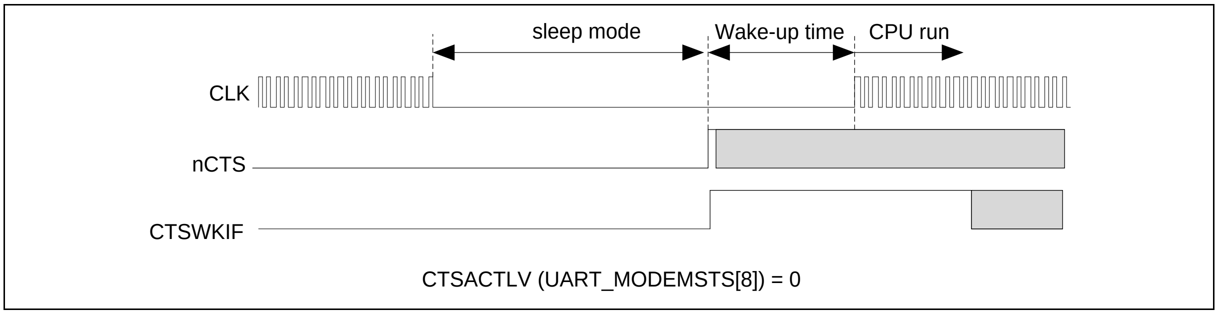


Figure 6.10-5 UART nCTS Wake-UP Case1

**nCTS Wake-Up Case 2 (nCTS transition from high to low)**

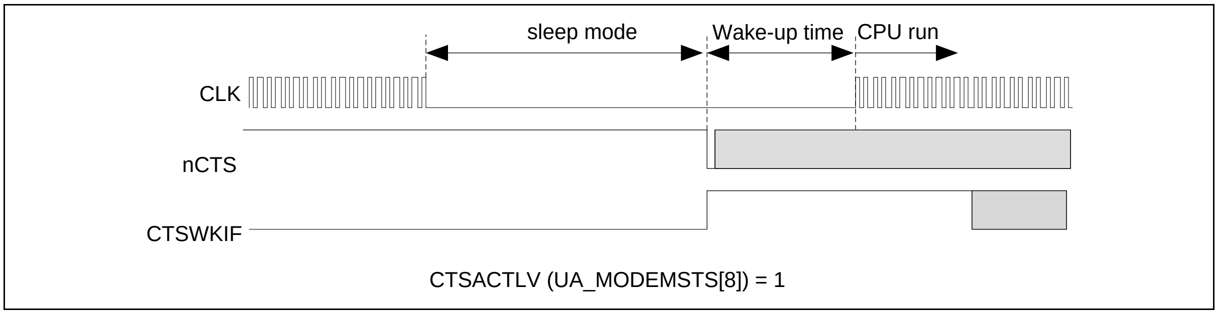


Figure 6.10-6 UART nCTS Wake-UP Case2

**Data Wake-Up**

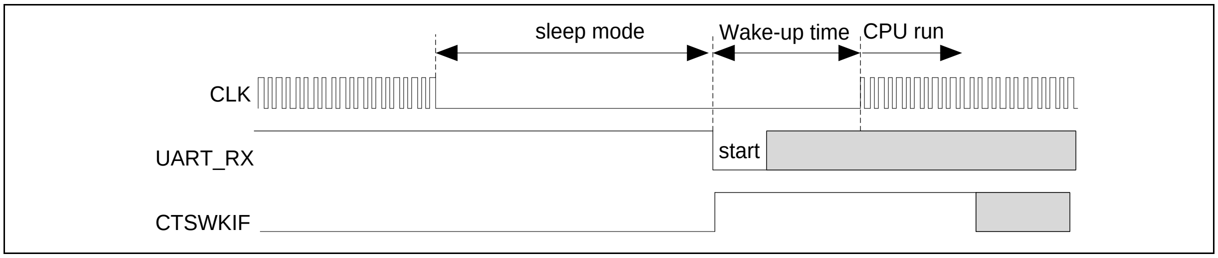


Figure 6.10-7 UART Data Wake-Up

### *6.10.5.6 UART Controller Interrupt and Status*

Each UART Controller supports ten types of interrupts including:

- Receiver threshold level reached interrupt (RDAINT)
- Transmitter FIFO empty interrupt (THERINT)
- Line status interrupt (parity error, frame error or break error) (RLSINT)
- MODEM status interrupt (MODEMINT)
- Receiver buffer time-out interrupt (RXTOINT)
- Buffer error interrupt (BUFERRINT)
- LIN bus interrupt (LININT)
- nCTS wake-up interrupt (CTSWKIF)
- Data wake-up interrupt (DATWKIF)
- Auto-baud rate interrupt (ABRIF)

The following tables describe the interrupt sources and flags. The interrupt is generated when the interrupt flag is generated and the interrupt enable bit is set. User must clear the interrupt flag after the interrupt is generated.

| Interrupt Source | Interrupt Indicator | Interrupt Enable Bit | Interrupt Flag | Flag Cleared By |
|---|---|---|---|---|
| Receive Data Available Interrupt | RDAINT | RDAIEN | **RDAIF** | Read UART_DAT |
| Transmit Holding Register Empty Interrupt | THERINT | TJREIEN | **THREIF** | Write UART_DAT |

| | | | | |
|---|---|---|---|---|
| Receive Line Status Interrupt | RLSINT | RLSIEN | **RLSIF** = BIF | **Write ‘1’ to** BIF |
| | | | **RLSIF** = FEF | **Write ‘1’ to** FEF |
| | | | **RLSIF =** PEF | **Write ‘1’ to** PEF |
| | | | **RLSIF** = ADDRDETF | **Write ‘1’ to** ADDRDETF |
| Modem Status Interrupt | MODEMINT | MODEMIEN | **MODEMIF** = CTSDETF | **Write ‘1’ to** CTSDETF |
| RX Time-out Interrupt | RXTOINT | RXTOIEN | **RXTOIF** | Read UART_DAT |
| Buffer Error Interrupt | BUFERRINT | BUFERRIEN | **BUFERRIF** = TXOVIF | **Write ‘1’ to** TXOVIF |
| | | | **BUFERRIF** = RXOVIF | **Write ‘1’ to** RXOVIF |
| LIN Bus interrupt | LININT | LIN _IEN | **LINIF =** BRKDETF | **Write ‘1’ to** LINIF **and**<br>**Write ‘1’ to** BRKDETF |
| | | | **LINIF =** BITEF | **Write ‘1’ to** BITEF |
| | | | **LINIF =** SLVIDPEF | **Writing ‘1’ to** SLVIDPEF |
| | | | **LINIF =** SLVHEF | **Wrie ‘1’ to** SLVHEF |
| | | | **LINIF =** SLVHDETF | **Write ‘1’ to** SLVHDETF |
| nCTS wake-up interrupt | N/A | WKCTSIEN | **CTSWKIF** | **Write ‘1’ to** CTSWKIF |
| Data wake-up interrupt | N/A | WKDATIEN | **DATWKIF** | **Write ‘1’ to** DATWKIF |
| Auto-baud rate interrupt | N/A | ABRIEN | **ABRIF =** ABRDIF | **Write ‘1’ to** ABRDIF |
| | | | **ABRIF =** ABRDTOIF | **Write ‘1’ to** ABRDTOIF. |

Table 6.10-3 UART Controller Interrupt Source and Flag List

### *6.10.5.7 UART Function Mode*

The UART Controller provides UART function (Setting FUNCSEL (UART_FUNCSEL [1:0]) to ’00’ to enable UART function mode). The UART baud rate is up to 1 Mbps.

The UART provides full-duplex and asynchronous communications. The transmitter and receiver contain 16 bytes FIFO for payloads. User can program receiver buffer trigger level and receiver buffer time-out detection for receiver. The transmitting data delay time between the last stop and the next start bit can be programed by setting DLY (UART_TOUT [15:8]) register. The UART supports hardware auto-flow control that provides programmable nRTS flow control trigger level.

The number of data bytes in RX FIFO is equal to or greater than RTSTRGLV (UART_FIFO[19:16]), the nRTS is de-asserted.

### UART Line Control Function

The UART Controller supports the fully programmable serial-interface feature by setting the UART_LINE register. User can program UART_LINE register for the word length, stop bit and parity bit setting. The following tables list the UART word, stop bit length and the parity bit settings.

| NSB (UART_LINE[2]) | WLS (UART_LINE[1:0]) | Word Length (Bit) | Stop Length (Bit) |
|---|---|---|---|
| 0 | 00 | 5 | 1 |
| 0 | 01 | 6 | 1 |
| 0 | 10 | 7 | 1 |
| 0 | 11 | 8 | 1 |
| 1 | 00 | 5 | 1.5 |
| 1 | 01 | 6 | 2 |
| 1 | 10 | 7 | 2 |
| 1 | 11 | 8 | 2 |

Table 6.10-4 UART Line Control of Word and Stop Length Setting

| Parity Type | SPE (UART_LINE[5]) | EPE (UART_LINE[4]) | PBE (UART_LINE[3]) | Description |
|---|---|---|---|---|
| No Parity | x | x | 0 | No parity bit output. |
| Odd Parity | 0 | 0 | 1 | Odd Parity is calculated by adding all the “1’s” in a data stream and adding a parity bit to the total bits, to make the total count an odd number. |
| Even Parity | 0 | 1 | 1 | Even Parity is calculated by adding all the “1’s” in a data stream and adding a parity bit to the total bits, to make the count an even number. |
| Forced Mask Parity | 1 | 0 | 1 | Parity bit always logic 1.<br>Parity bit on the serial byte is set to “1” regardless of total number of “1’s” (even or odd counts). |
| Forced Space Parity | 1 | 1 | 1 | Parity bit always logic 0.<br>Parity bit on the serial byte is set to “0” regardless of total number of “1’s” (even or odd counts). |

Table 6.10-5 UART Line Control of Parity Bit Setting

### UART Auto-Flow Control Function

The UART supports auto-flow control function that uses two signals, nCTS (clear-to-send) and nRTS (request-to-send), to control the flow of data transfer between the UART and external devices (e.g. Modem). When auto-flow is enabled, the UART is not allowed to receive data until the UART asserts nRTS to external device. When the number of bytes stored in the RX FIFO equals the value of RTSTRGLV (UART_FIFO [19:16]), the nRTS is de-asserted. The UART sends data out when UART detects nCTS is asserted from external device. If the valid asserted nCTS is not detected, the UART will not send data out.

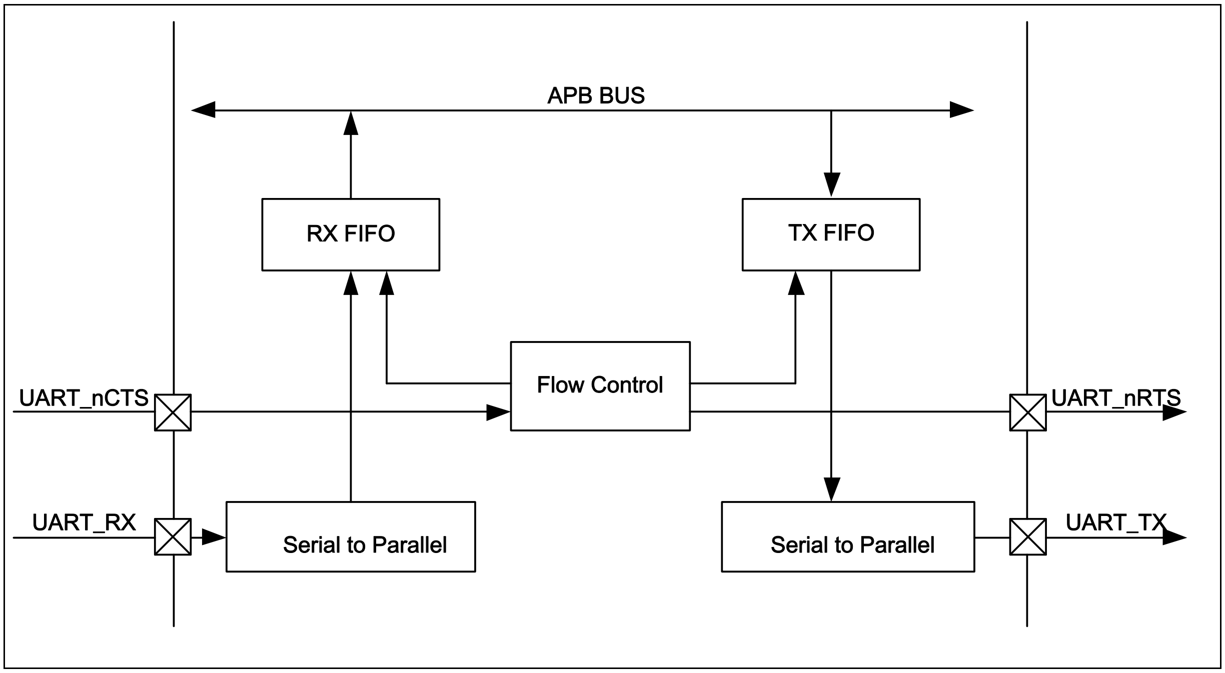


Figure 6.10-8 Auto-Flow Control Block Diagram

The following diagram demonstrates the nCTS auto-flow control of UART function mode. User must set ATOCTSEN (UART_INTEN [13]) to enable nCTS auto-flow control function. The CTSACTLV (UART_MODEMSTS [8]) can set nCTS pin input active state. The CTSDETF (UART_MODEMSTS[0]) is set when any state change of nCTS pin input has occurred, and then TX data will be automatically transmitted from TX FIFO.

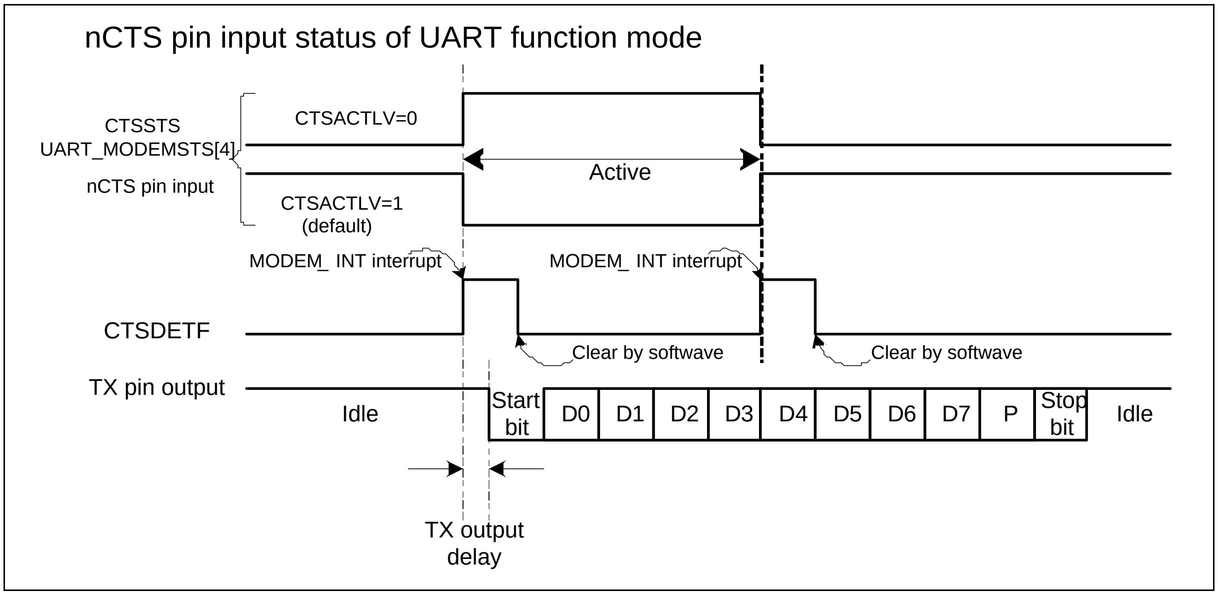


Figure 6.10-9 UART nCTS Auto-Flow Control Enabled

As shown in the following figure, in UART nRTS auto-flow control mode (ATORTSEN(UART_INTEN[12])=1), the nRTS internal signal is controlled by UART FIFO controller with RTSTRGLV(UART_FIFO[19:16]) trigger level.

Setting RTSACTLV(UART_MODEM[9]) can control the nRTS pin output is inverse or non-inverse from nRTS signal. User can read the RTSSTS (UART_MODEM[13]) bit to get real nRTS pin output voltage logic status.

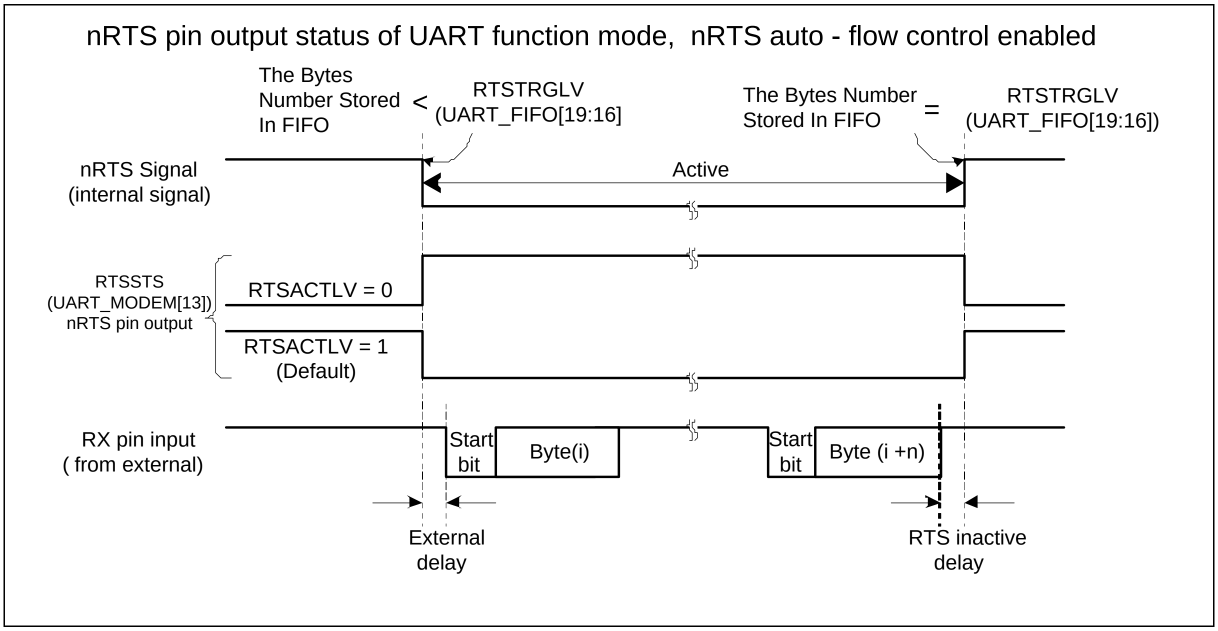

nRTS pin output status of UART function mode, nRTS auto - flow control enabled
The Bytes Number Stored In FIFO
<
RTSTRGLV (UART_FIFO[19:16]
The Bytes Number Stored In FIFO
=
RTSTRGLV (UART_FIFO[19:16])
nRTS Signal (internal signal)
Active
RTSSTS (UART_MODEM[13]) nRTS pin output
RTSACTLV = 0
RTSACTLV = 1 (Default)
RX pin input ( from external)
Start bit
Byte(i)
Start bit
Byte (i +n)
External delay
RTS inactive delay

Figure 6.10-10 UART nRTS Auto-Flow Control Enabled

As shown in the following figure, in software mode (ATORTSEN(UART_INTEN[12])=0), the nRTS flow is directly controlled by software programming of RTS(UART_MODEM[1]) control bit.

Setting RTSACTLV(UART_MODEM[9]) can control the nRTS pin output is inverse or non-inverse from RTS(UART_MODEM[1]) control bit. User can read the RTSSTS(UART_MODEM[13]) bit to get real nRTS pin output voltage logic status.

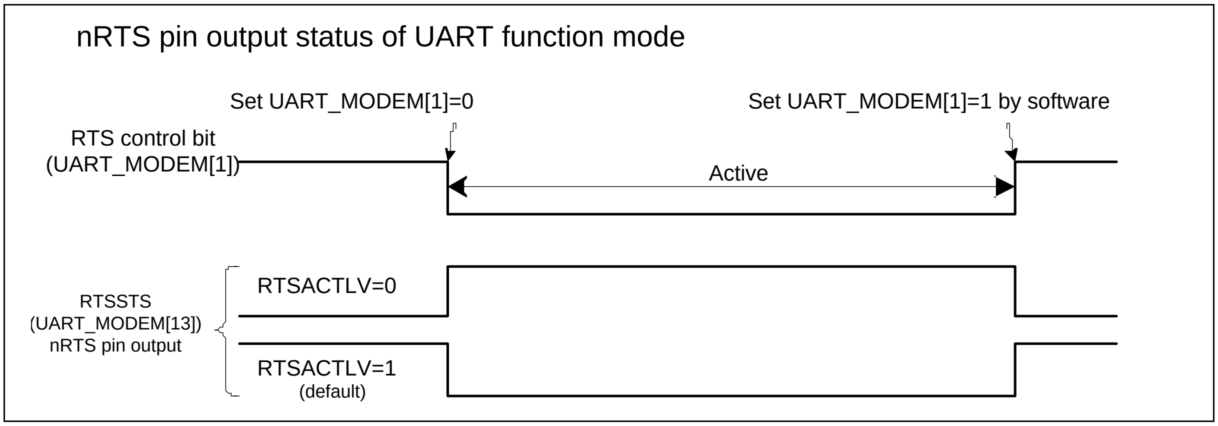

nRTS pin output status of UART function mode
Set UART_MODEM[1]=0
Set UART_MODEM[1]=1 by software
RTS control bit (UART_MODEM[1])
Active
RTSSTS (UART_MODEM[13]) nRTS pin output
RTSACTLV=0
RTSACTLV=1 (default)

Figure 6.10-11 UART nRTS Auto-Flow with Software Control

### *6.10.5.8 IrDA Function Mode*

The UART Controller also provides Serial IrDA (SIR, Serial Infrared) function (Setting UART_FUNCSEL [1:0] to '10' to enable the IrDA function). The SIR specification defines a short-range infrared asynchronous serial transmission mode with one start bit, 8 data bits, and 1 stop bit. The maximum data rate is 115.2 kbps. The IrDA SIR block contains an IrDA SIR protocol encoder/decoder. The IrDA SIR protocol is half-duplex only. Thus, it cannot transmit and receive data at the same time. The IrDA SIR physical layer specifies a minimum 10 ms transfer delay between transmission and reception, and this delay feature must be implemented by software.

In IrDA mode, the BAUDM1 (UART_BAUD [29]) must be cleared.

**Baud Rate = Clock / (16 * BRD +2)**, where BRD (UART_BAUD[15:0]) is Baud Rate Divider in UART_BAUD register.

The IrDA control block diagram is shown as follows.

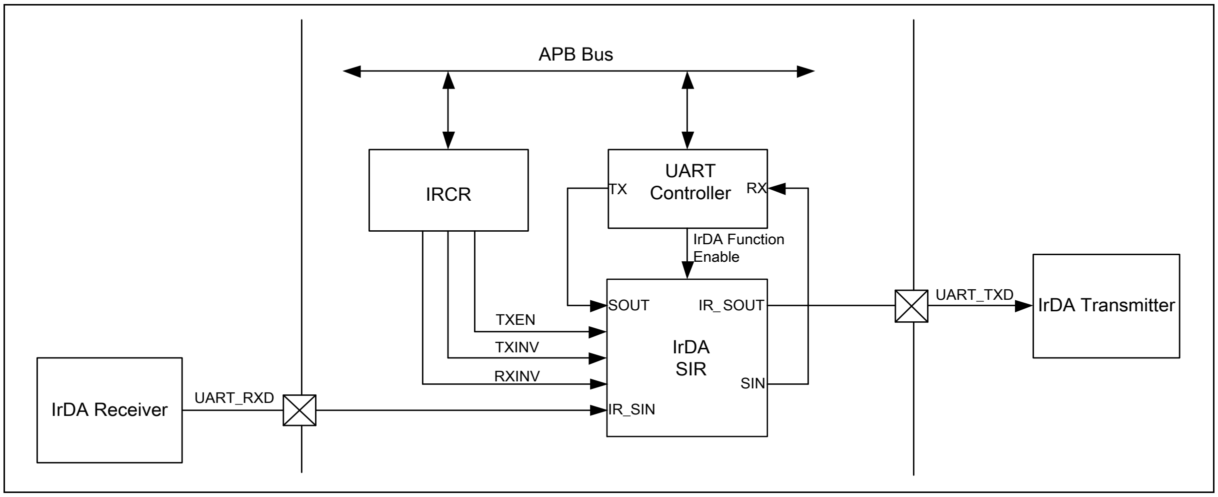


Figure 6.10-12 IrDA Control Block Diagram

**IrDA SIR Transmit Encoder**

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

The transmitted pulse width is specified as 3/16 period of baud rate.

**IrDA SIR Receive Decoder**

The IrDA SIR Receive Decoder demodulates the Return-to-Zero bit stream from the input detector and outputs the NRZ serial bits stream to the UART received data input.

A start bit is detected when the decoder input is LOW.

### IrDA SIR Operation

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

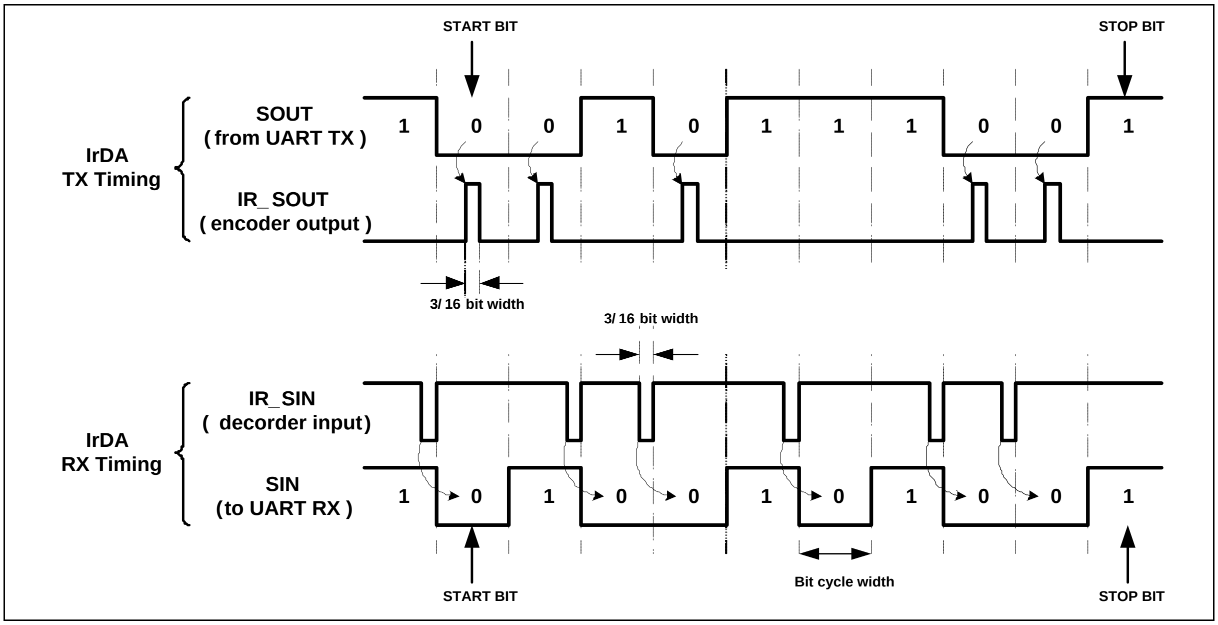


Figure 6.10-13 IrDA TX/RX Timing Diagram

### 6.10.5.9 *LIN Function Mode (Local Interconnection Network)*

The UART1/UART2 supports LIN function. Setting FUNCSEL (UART_FUNCSEL[1:0]) to ’01’ to select LIN mode operation. The UART1/UART2 supports LIN break/delimiter generation and break/delimiter detection in LIN master mode, and supports header detection and automatic resynchronization in LIN Slave mode.

### Structure of LIN Frame

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

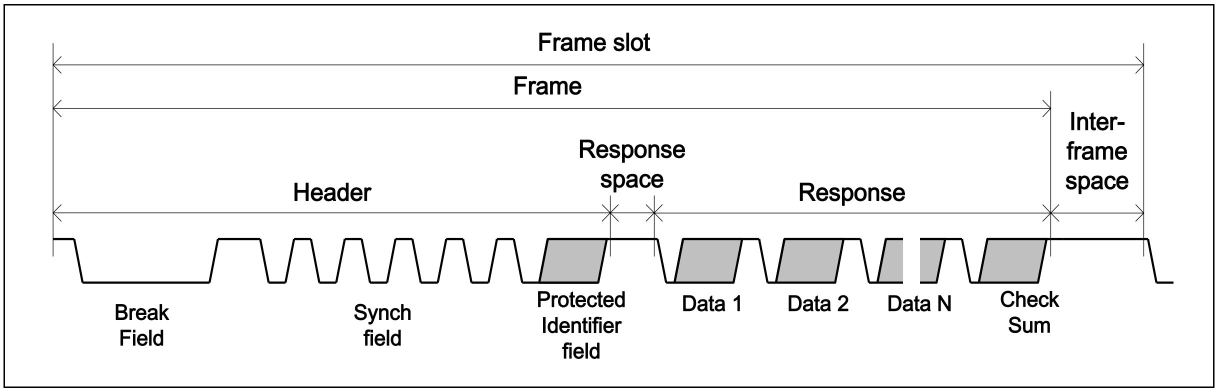


Figure 6.10-14 Structure of LIN Frame

### Structure of LIN Byte

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

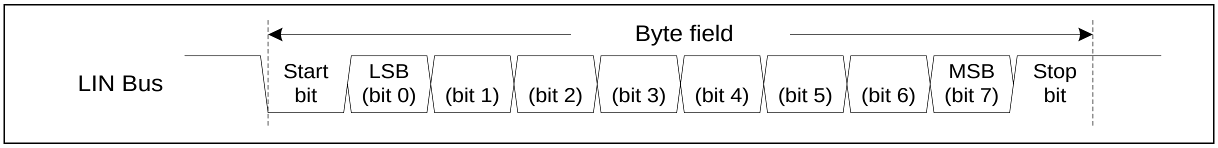
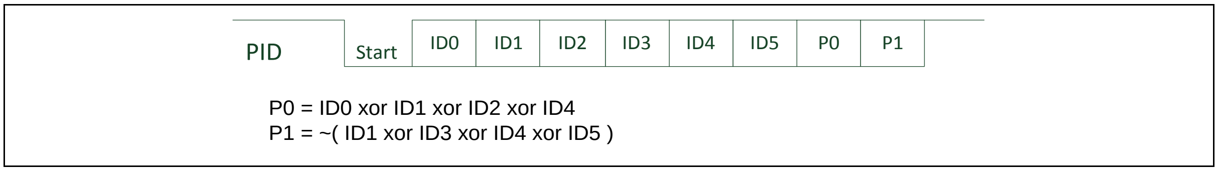


Figure 6.10-15 Structure of LIN Byte

### LIN Frame ID and Parity Format

The LIN frame ID value in LIN function mode is shown, and the frame ID parity can be generated by software or hardware depending on whether IDPEN (UART_LINCTL[9]) = 1.

If the parity generated by hardware, and user fills in ID0~ID5, (UART_LINCTL [29:24]), hardware will calculate P0 (UART_LINCTL[30]) and P1 (UART_LINCTL[31]), otherwise user must fill a frame ID and parity in this field.



P0 = ID0 xor ID1 xor ID2 xor ID4
P1 = ~( ID1 xor ID3 xor ID4 xor ID5 )

Figure 6.10-16 LIN Frame ID and Parity Format

### LIN Master Mode

The UART1/UART2 controller supports LIN Master mode. To enable and initialize the LIN Master mode, the following steps are necessary:

1. Setting the UART_BAUD register to select the desired baud rate.
2. Setting WLS (UART_LINE[1:0]) to '11' to configure the word length with 8 bits, clearing PBE (UART_LINE[3]) bit to disable parity check and clearing NSB (UART_LINE[2]) bit to configure with one stop bit.
3. Setting FUNCSEL (UART_FUNCSEL[1:0]) to '01' to select LIN function mode operation.

A complete header consists of a break field and sync field followed by a frame identifier (frame ID). The UART1/UART2 controller can be selected header sending by three header selected modes. The header selected mode can be "break field" or "break field and sync field" or "break field, sync field and frame ID field" by setting HSEL (UART_LINCTL[23:22]). If the selected header is "break field", software must handle the following sequence to send a complete header to bus by filling sync data (0x55) and frame ID data to the UART_DAT register. If the selected header is "break field and sync field", software must handle the sequence to send a complete header to bus by filling the frame ID data to UART_DAT register, and if the selected header is "break field, sync field and frame ID field", hardware will control the header sending sequence automatically but software must fill a frame ID data to PID (UART_LINCTL [31:24]). When operating in header selected mode in which the selected header is "break field, sync field and frame ID field", the frame ID parity bit can be calculated by software or hardware depending whether the IDPEN (UART_LINCTL[9]) bit is set or not.

| HSEL | Break Field | Sync Field | ID Field |
|---|---|---|---|

| | | | |
|---|---|---|---|
| 0 | Generated by Hardware | Handled by Software | Handled by Software |
| 1 | Generated by Hardware | Generated by Hardware | Handled by Software |
| 2 | Generated by Hardware | Generated by Hardware | Generated by Hardware<br>(But Software needs to fill ID to PID<br>(UART_LINCTL[31:24]) first |

Table 6.10-6 LIN Header Selection in Master Mode

When UART is operated in LIN data transmission, LIN bus transfer state can be monitored by hardware or software. User can enable hardware monitoring by setting BITERREN (UART_LINCTL [12]) to “1”, if the input pin (UART_RX) state is not equal to the output pin (UART_TX) state in LIN transmitter state that hardware will generate an interrupt to CPU. Software can also monitor the LIN bus transfer state by checking the read back data in UART_DAT register. The following sequence is a program sequence example.

The procedure without software error monitoring in Master mode:

1. Fill Protected Identifier to PID (UART_LINCTL[31:24]).
2. Select the hardware transmission header field including “break field + sync field + protected identifier field” by setting HSEL (UART_LINCTL [23:22]) to “10”.
3. Set SENDH (UART_LINCTL[8]) bit to 1 for requesting header transmission.
4. Wait until SENDH (UART_LINCTL[8]) bit cleared by hardware.
5. Wait until TXEMPTYF (UART_FIFOSTS[28]) set to 1 by hardware.

**Note1:** The default setting of break field is 12 dominant bits (break field) and 1 recessive bit break/sync delimiter. Setting BRKFL (UART_LINCTL [19:16]) and BSL (UART_LINCTL[21:20]) to change the LIN break field length and break/sync delimiter length.

**Note2:** The default setting of break/sync delimiter length is 1-bit time and the inter-byte spaces default setting is also 1-bit time. Setting BSL (UART_LINCTL[21:20]) and DLY(UART_TOUT[15:8]) can change break/sync delimiter length and inter-byte spaces.

**Note3:** If the header includes the “break field, sync field and frame ID field”, software must fill a frame ID to PID (UART_LINCTL[31:24]) before trigger header transmission (setting the SENDH (UART_LINCTL[8]). The frame ID parity can be generated by software or hardware depending on IDPEN (UART_LINCTL[9]) setting. If the parity generated by software with IDPEN (UART_LINCTL[9]) is set to ‘0’, software must fill 8 bit data (include 2 bit parity) in this field. If the parity generated by hardware with IDPEN (UART_LINCTL[9]) is set to ‘1’ , software fill ID0~ID5 and hardware calculates P0 and P1.

The procedure with software error monitoring in Master mode:

1. Choose the hardware transmission header field to only include “break field” by setting HSEL (UART_LINCTL [23:22])] to ‘00’.
2. Enable break detection function by setting BRKDETEN (UART_LINCTL[10]).
3. Request break + break/sync delimiter transmission by setting the SENDH (UART_LINCTL[8]).
4. Wait until the BRKDETF (UART_LINSTS[8]) flag is set to “1” by hardware.
5. Request sync field transmission by writing 0x55 into UART_DAT register.
6. Wait until the RDAIF (UART_INTSTS[0]) is set to “1” by hardware and then read back the

UART_DAT register.

7. Request header frame ID transmission by writing the protected identifier value to UART_DAT register.

8. Wait until the RDAIF (UART_INTSTS[0]) is set to "1" by hardware and then read back the UART_DAT register.

### LIN break and delimiter detection

When software enables the break detection function by setting BRKDETEN (UART_LINCTL[10]), the break detection circuit is activated. The break detection circuit is totally independent from the UART1/UART2 receiver.

When the break detection function is enabled, the circuit looks at the input UART_RX pin for a start signal. If UART LIN controller detects consecutive dominant is greater than 11 bits dominant followed by a recessive bit (delimiter), the BRKDETF (UART_LINSTS[8]) flag is set at the end of break field. If the LINIEN (UART_INTEN[8]) bit is set to 1, an interrupt LININT (UART_INTSTS[15]) will be generated. The behavior of the break detection and break flag are shown in the following figure.

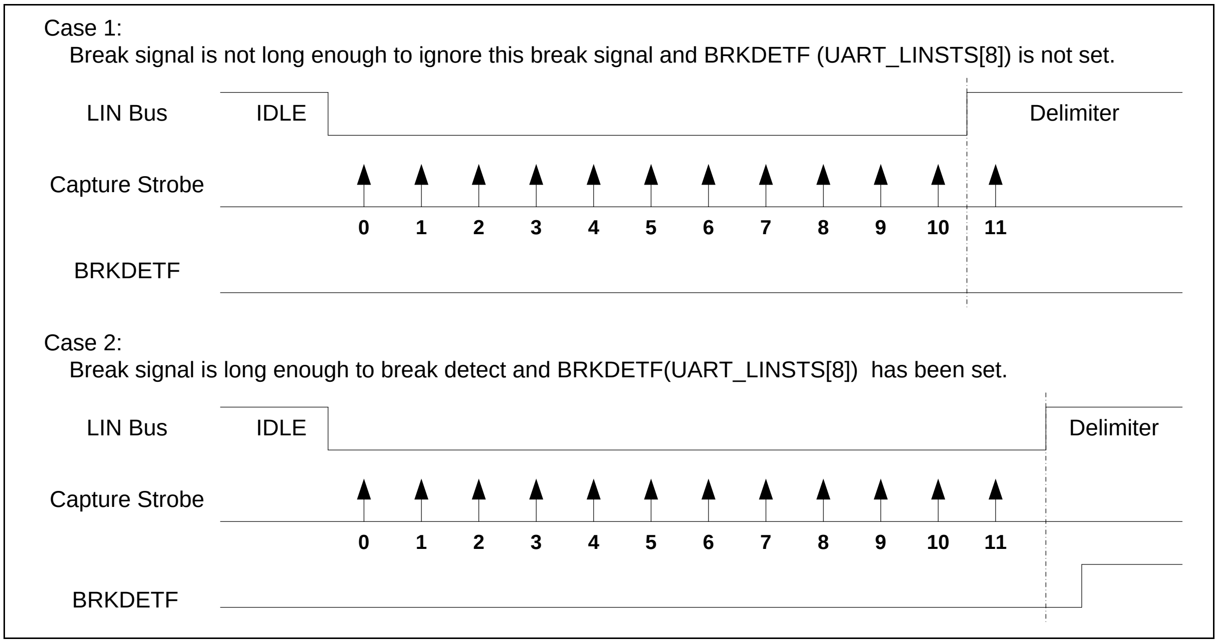

Case 1:
Break signal is not long enough to ignore this break signal and BRKDETF (UART_LINSTS[8]) is not set.
LIN Bus
IDLE
Delimiter
Capture Strobe
0
1
2
3
4
5
6
7
8
9
10
11
BRKDETF
Case 2:
Break signal is long enough to break detect and BRKDETF(UART_LINSTS[8]) has been set.
LIN Bus
IDLE
Delimiter
Capture Strobe
0
1
2
3
4
5
6
7
8
9
10
11
BRKDETF

Figure 6.10-17 Break Detection in LIN Mode

### LIN Slave Mode

The UART1/UART2 controller supports LIN Slave mode. To enable and initialize the LIN Slave mode, the following steps are necessary:

1. Set the UART_BAUD register to select the desired baud rate

2. Configure the data length to 8 bits by setting WLS (UART_LINE[1:0]) to '11' and disable parity check by clearing PBE (UART_LINE[3]) bit and configure with one stop bit by clearing NSB (UART_LINE[2]) bit.

3. Select LIN function mode by setting FUNCSEL (UART_FUNCSEL[1:0]) to '01'

4. Enable LIN Slave mode by setting the SLVEN (UART_LINCTL[0]) to 1.

### LIN header reception

According to the LIN protocol, a slave node must wait for a valid header which comes from the master node. Next the slave task will take one of following actions (depend on the master header frame ID value)

- Receive the response.
- Transmit the response.
- Ignore the response and wait for the next header.

In LIN Slave mode, user can enable the slave header detection function by setting the SLVHDEN (UART_LINCTL[10]) to detect complete frame header (receive “break field”, “sync field” and “frame ID field”). When a LIN header is received, the SLVHDETF (UART_LINSTS[0]) flag will be set. If the LINIEN (UART_INTEN[8]) bit is set to 1, an interrupt will be generated. User can enable the frame ID parity check function by setting IDPEN (UART_LINCTL[9]). If only received frame ID parity is not correct (break and sync filed are correct), the SLVIDPEF (UART_LINSTS[2]) flag is set to ‘1’. If the LINIEN(UART_INTEN[8]) is set to 1, an interrupt will be generated and SLVHDETF ( UART_LINSTS[0]) is set to ‘1’. User can also put LIN in mute mode by setting MUTE (UART_LINCTL[4]) to ‘1’. This mode allows detection of headers only (break + sync + frame ID) and prevents the reception of any other characters. In order to avoid bit rate tolerance, the controller supports automatic resynchronization function to avoid clock deviation error, user can enable this feature by setting SLVAREN (UART_LINCTL[2]).

### LIN response transmission

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

### LIN header time-out error

The LIN slave controller contains a header time-out counter. If the entire header is not received within the maximum time limit of 57 bit times, the header error flag SLVHEF (UART_LINSTS [1]) will be set. The time-out counter is enabled at each break detect edge and stopped in the following conditions.

- A LIN frame ID field has been received.
- The header error flag asserts.
- Writing 1 to the SLVSYNCF (UART_LINSTS[3]) to re-search a new frame header.

### Mute mode and LIN exit from mute mode condition

In Mute mode, a LIN slave node will not receive any data until specified condition occurred. It allows header detection only and prevents the reception of any other characters. User can enable Mute mode by setting the MUTE (UART_LINCTL[4]) and exiting from Mute mode condition can be selected by HSEL (UART_LINCTL[23:22]).

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

The LIN slave controller exiting from Mute mode is described as follows: If HSEL (UART_LINCTL[23:22]) is set to “break field”, when LIN slave controller detects a valid LIN break + delimiter, the controller will enable the receiver (exit from Mute mode) and subsequent data (sync data, frame ID data, response data) are received in RX-FIFO.

If HSEL (UART_LINCTL[23:22]) is set to “break field and sync field”, when the LIN slave controller detects a valid LIN break + delimiter followed by a valid sync field without frame error, the controller will enable the receiver (exit from mute mode) and subsequent data(ID data, response data) are received in RX-FIFO. If HSEL (UART_LINCTL[23:22]) is set to “break field, sync field and ID field”, when the LIN slave controller detects a valid LIN break + delimiter and valid sync field without frame error followed by ID data without frame error and received ID data matched PID (UART_LINCTL[31:24]) value. The controller will enable the receiver (exit from mute mode) and subsequent data (response data) are received in RX-FIFO.

**Slave mode non-automatic resynchronization (NAR)**

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

1. Select the desired baud rate by setting the UART_BAUD register.
2. Select LIN function mode by setting FUNCSEL (UART_FUNCSEL[1:0]) to ‘01’.
3. Disable automatic resynchronization function by setting SLVAREN (UART_LINCTL[2]) is set to 0.
4. Enable LIN Slave mode by setting the SLVEN (UART_LINCTL[0]) is set to 1.

**Slave mode with automatic resynchronization (AR)**

In Automatic Resynchronization (AR) mode, the controller will adjust the baud rate generator after each sync field reception. The initialization process flow of Automatic Resynchronization mode is shown as follows:

1. Select the desired baud rate by setting the UART_BAUD register.
2. Select LIN function mode by setting UART_FUNCSEL (UART_FUNCSEL[1:0]) to ‘01’
3. Enable automatic resynchronization function by setting SLVAREN (UART_LINCTL[2]) to ‘1’.
4. Enable LIN Slave mode by setting the SLVEN (UART_LINCTL[0]) is set to ‘1’.

When the automatic resynchronization function is enabled, after each LIN break field, the time duration between five falling edges is sampled on peripheral clock and the result of this measurement is stored in an internal 13-bit register and the UART_BAUD register value will be automatically updated at the end of the fifth falling edge. If the measure timer (13-bit) overflows before five falling edges, then the header error flag SLVHEF (UART_LINSTS [1]) will be set.

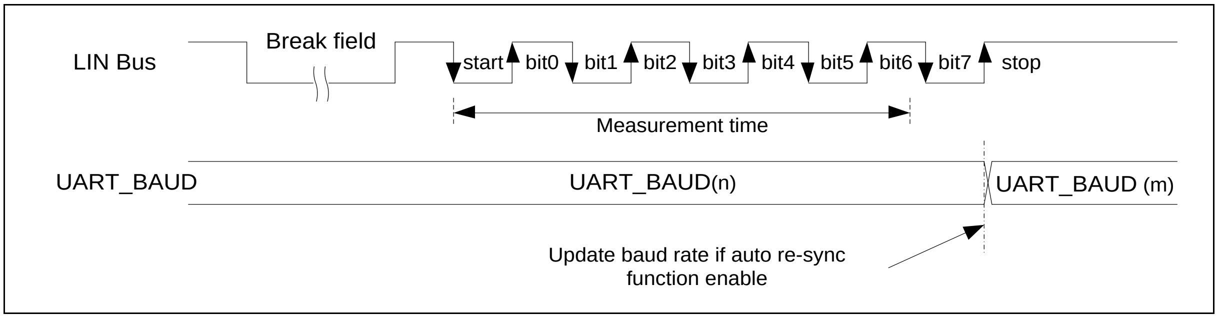


Figure 6.10-18 LIN Sync Field Measurement

When operating in Automatic Resynchronization (AR) mode, software must select the desired baud rate by setting the UART_BAUD register and hardware will store it at internal TEMP_REG register, after each LIN break field, the time duration between five falling edges is sampled on peripheral clock and the result of this measurement is stored in an internal 13-bit register BAUD_LIN and the result will be updated to UART_BAUD register automatically.

To guarantee the transmission baud rate, the baud rate generator must reload the initial value before each new break reception. The initial value is programmed by the application during initialization (TEMP_REG). User can set SLVDUEN (UART_LINCTL [3]) to enable auto reload initial baud rate value function. If the SLVDUEN (UART_LINCTL [3]) is set, when received the next character, hardware will auto reload the initial value to UART_BAUD, and when the UART_BAUD be updated, the SLVDUEN (UART_LINCTL [3]) will be cleared automatically. The behavior of LIN updated method as shown in the following figure.

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

**Note2:** When a header error is detected, user must write 1 to SLVSYNCF (UART_LINSTS[3]) to re-search new frame header. When writing 1 to it, hardware will reload the initial baud rate TEMP_REG and re-search new frame header.

**Note3:** When operating in Automatic Resynchronization mode, the baud rate setting must be operated at mode2 (BAUDM1 (UART_BAUD [29]) and BAUDM0 (UART_BAUD[28]) must be 1).

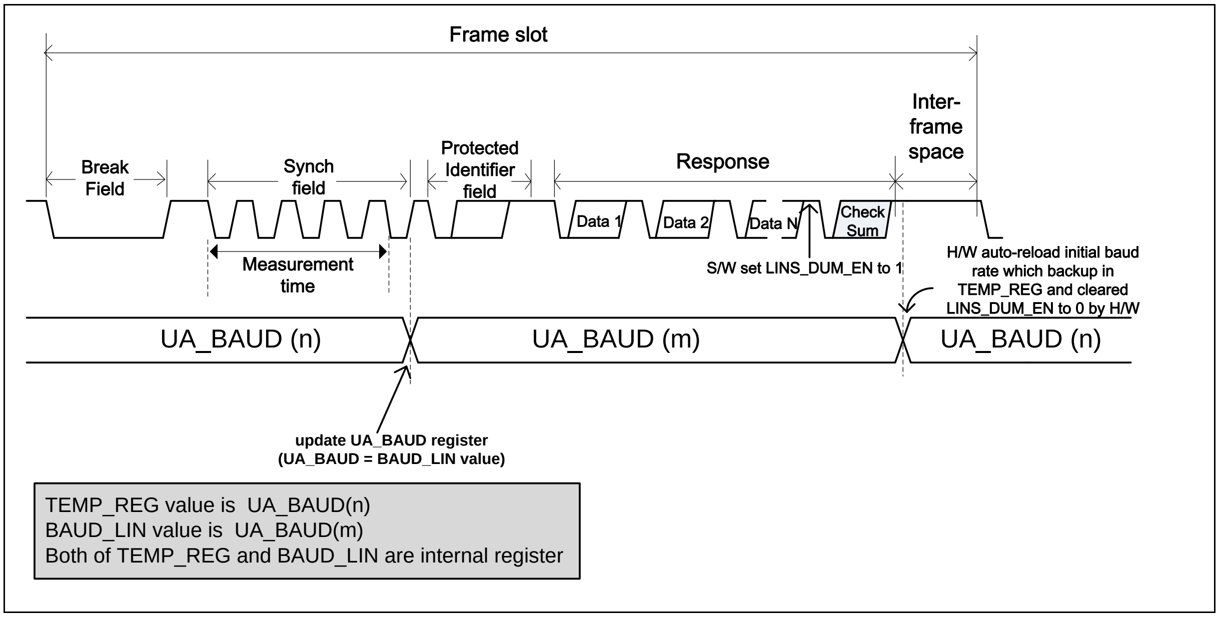

Frame slot
Break Field
Synch field
Protected Identifier field
Response
Inter-frame space
Data 1
Data 2
Data N
Check Sum
Measurement time
S/W set LINS_DUM_EN to 1
H/W auto-reload initial baud rate which backup in TEMP_REG and cleared LINS_DUM_EN to 0 by H/W
UA_BAUD (n)
UA_BAUD (m)
UA_BAUD (n)
update UA_BAUD register (UA_BAUD = BAUD_LIN value)
TEMP_REG value is UA_BAUD(n)
BAUD_LIN value is UA_BAUD(m)
Both of TEMP_REG and BAUD_LIN are internal register

Figure 6.10-19 UART_BAUD Update Sequence in AR mode if SLVDUEN is 1

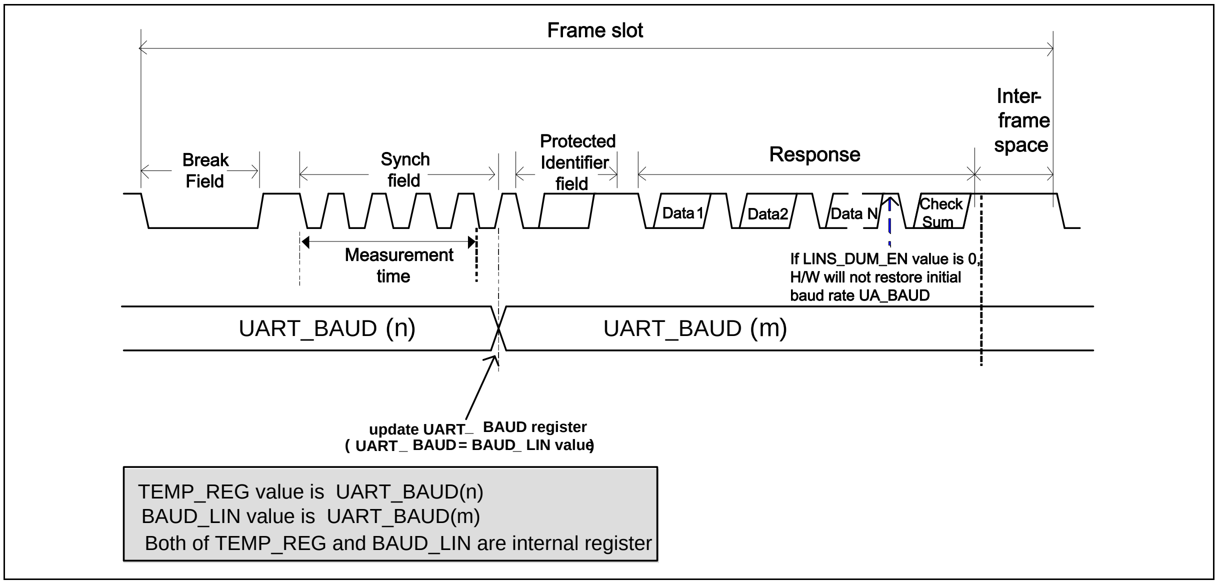

Frame slot
Inter-frame space
Break Field
Synch field
Protected Identifier field
Response
Data 1
Data2
Data N
Check Sum
Measurement time
If LINS_DUM_EN value is 0, H/W will not restore initial baud rate UA_BAUD
UART_BAUD (n)
UART_BAUD (m)
update UART_ BAUD register
( UART_ BAUD= BAUD_ LIN value)
TEMP_REG value is UART_BAUD(n)
BAUD_LIN value is UART_BAUD(m)
Both of TEMP_REG and BAUD_LIN are internal register

Figure 6.10-20 UART_BAUD Update Sequence in AR mode if SLVDUEN is 0

**Deviation error on the sync field**

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

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

- If the difference is more than 14.84%, the header error flag SLVHEF (UART_LINSTS[1]) will be set.
- If the difference is less than 14.06%, the header error flag SLVHEF (UART_LINSTS[1]) will not be set.
- If the difference is between 14.84% and 14.06%, the header error flag SLVHEF (UART_LINSTS[1]) may either set or not.

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

- If the difference is more than 18.75%, the header error flag SLVHEF (UART_LINSTS[1]) will be set.
- If the difference is less than 15.62%, the header error flag SLVHEF (UART_LINSTS[1]) will not be set.
- If the difference is between 18.75% and 15.62%, the header error flag SLVHEF (UART_LINSTS[1]) may either set or not.

The deviation check is based on the current baud rate clock. Therefore, in order to guarantee correct deviation checking, the baud rate must reload the nominal value before each new break reception by setting SLVDUEN (UART_LINCTL[3]) register (It is recommend setting the SLVDUEN (UART_LINCTL[3]) bit before every checksum reception)

**LIN header error detection**

In LIN Slave function mode, when user enables the header detection function by setting the SLVHDEN (UART_LINCTL[1]), hardware will handle the header detect flow. If the header has an error, the LIN header error flag SLVHEF (UART_LINSTS[1]) will be set and an interrupt is generated if the LINIEN (UART_INTEN[8]) bit is set. When header error is detected, user must reset the detect circuit to re-search a new frame header by writing 1 to SLVSYNCF (UART_LINSTS[3]) to re-search a new frame header.

The LIN header error flag SLVHEF (UART_LINSTS[1]) is set if one of the following conditions occurs:

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

### *6.10.5.10 RS-485 Function Mode*

Another alternate function of UART Controller is RS-485 function (user must set UART_FUNCSEL [1:0] to ’11’ to enable RS-485 function), and direction control provided by nRTS pin from an asynchronous serial port. The RS-485 transceiver control is implemented by using the nRTS control signal to enable the RS-485 driver. Many characteristics of the RX and TX are same as UART in RS-485 mode.

The UART controller can be configured as an RS-485 addressable slave and the RS-485 master transmitter will identify an address character by setting the parity (9-th bit) to 1. For data characters, the parity is set to 0. Software can use UART_LINE register to control the 9-th bit (When the PBE, EPE and SPE are set, the 9-th bit is transmitted 0 and when PBE and SPE are set and EPE is cleared, the 9-th bit is transmitted 1).

The controller supports three operation modes: RS-485 Normal Multidrop Operation Mode (NMM), RS-485 Auto Address Detection Operation Mode (AAD) and RS-485 Auto Direction Control Operation Mode (AUD). Software can choose any operation mode by programming the UART_ALTCTL register, and drive the transfer delay time between the last stop bit leaving the TX FIFO and the de-assertion of by setting DLY (UART_TOUT [15:8]) register.

### RS-485 Normal Multidrop Operation Mode (NMM)

In RS-485 Normal Multidrop Operation Mode (RS485NMM (UART_ALTCTL[8]) = 1), in first, software must determine the data which before the address byte be detected will be stored in RX FIFO or not. If software wants to ignore any data before address byte detected, the flow is set RXOFF (UART_FIFO [8]) then enable RS485NMM (UART_ALTCTL [8]) and the receiver will ignore any data until an address byte is detected (bit 9 = 1) and the address byte data will be stored in the RX FIFO. If software wants to receive any data before address byte detected, the flow is disables RXOFF (UART_FIFO [8]) then enable RS485NMM (UART_ALTCTL [8]) and the receiver will received any data.

If an address byte is detected (bit 9 = 1), it will generate an interrupt to CPU and RXOFF (UART_FIFO [8]) can determine whether accepting the following data bytes stored in the RX FIFO. If software disables receiver by setting the RXOFF (UART_FIFO [8]) register, when a next address byte is detected, the controller will clear the RXOFF (UART_FIFO [8]) bit and the address byte data will be stored in the RX FIFO.

### RS-485 Auto Address Detection Operation Mode (AAD)

In RS-485 Auto Address Detection Operation Mode (RS485AAD (UART_ALTCTL[9]) = 1), the receiver will ignore any data until an address byte is detected (bit 9 = 1) and the address byte data matches the ADDRMV (UART_ALTCTL[31:24]) value. The address byte data will be stored in the RX FIFO. The all received byte data will be accepted and stored in the RX FIFO until an address byte data not match the ADDRMV (UART_ALTCTL[31:24]) value.

### RS-485 Auto Direction Function (AUD)

Another option function of RS-485 controllers is RS-485 auto direction control function (RS485AUD (UART_ALTCTL[10) = 1). The RS-485 transceiver control is implemented by using the nRTS control signal from an asynchronous serial port. The nRTS line is connected to the RS-485 transceiver enable pin such that setting the nRTS line to high (logic 1) enables the RS-485 transceiver. Setting the nRTS line to low (logic 0) puts the transceiver into the tri-state condition to disabled. User can set RTSACTLV in UART_MODEM register to change the nRTS driving level.

The following diagram demonstrates the RS-485 nRTS driving level in AUD mode. The nRTS pin will be automatically driven during TX data transmission.

Setting RTSACTLV(UART_MODEM[9]) can control nRTS pin output driving level. User can read the RTSSTS(UART_MODEM[13]) bit to get real nRTS pin output voltage logic status.

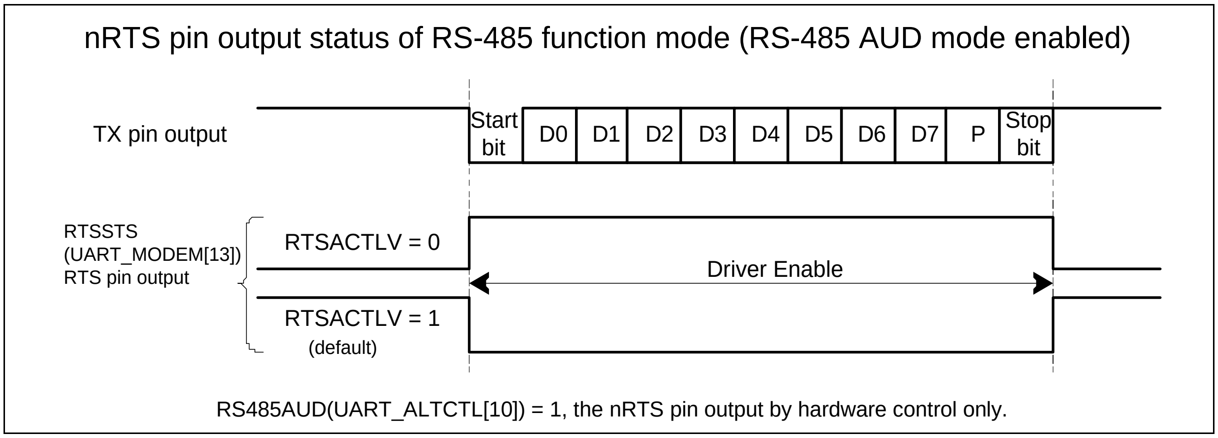

nRTS pin output status of RS-485 function mode (RS-485 AUD mode enabled)
TX pin output
Start bit
D0
D1
D2
D3
D4
D5
D6
D7
P
Stop bit
RTSSTS
(UART_MODEM[13])
RTS pin output
RTSACTLV = 0
Driver Enable
RTSACTLV = 1
(default)
RS485AUD(UART_ALTCTL[10]) = 1, the nRTS pin output by hardware control only.

Figure 6.10-21 RS-485 nRTS Driving Level in Auto Direction Mode

The following demonstrates the RS-485 nRTS driving level in software control (RS485AUD

(UART_ALTCTL[10])=0). The nRTS driving level is controlled by programing the RTS(UART_MODEM[1]) control bit.

Setting RTSACTLV (UART_MODEM[9]) can control the nRTS pin output is inverse or non-inverse from RTS(UART_MODEM[1]) control bit. User can read the RTSSTS (UART_MODEM[13]) bit to get real nRTS pin output voltage logic status.

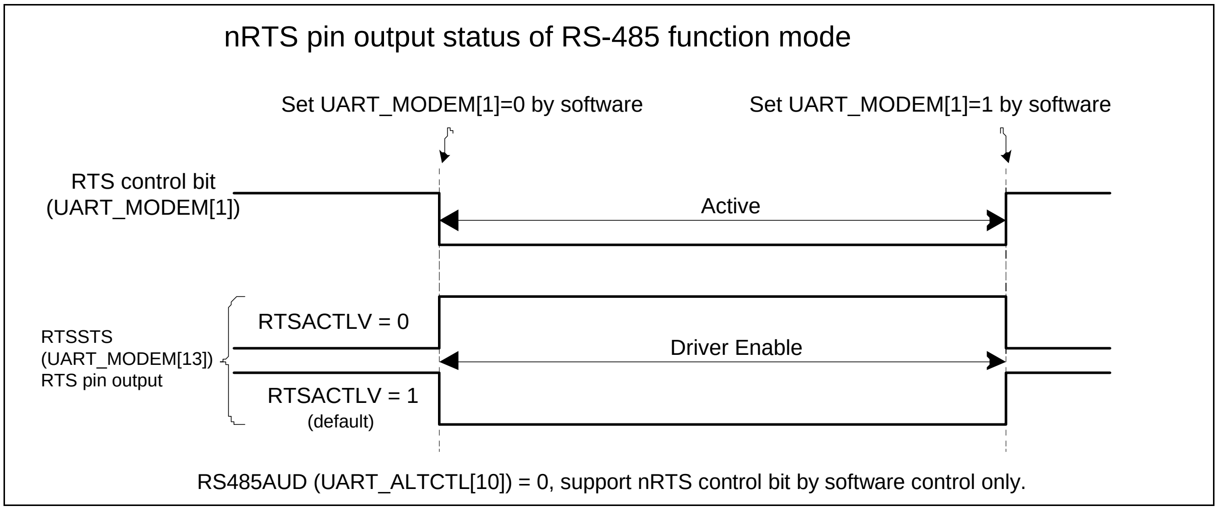
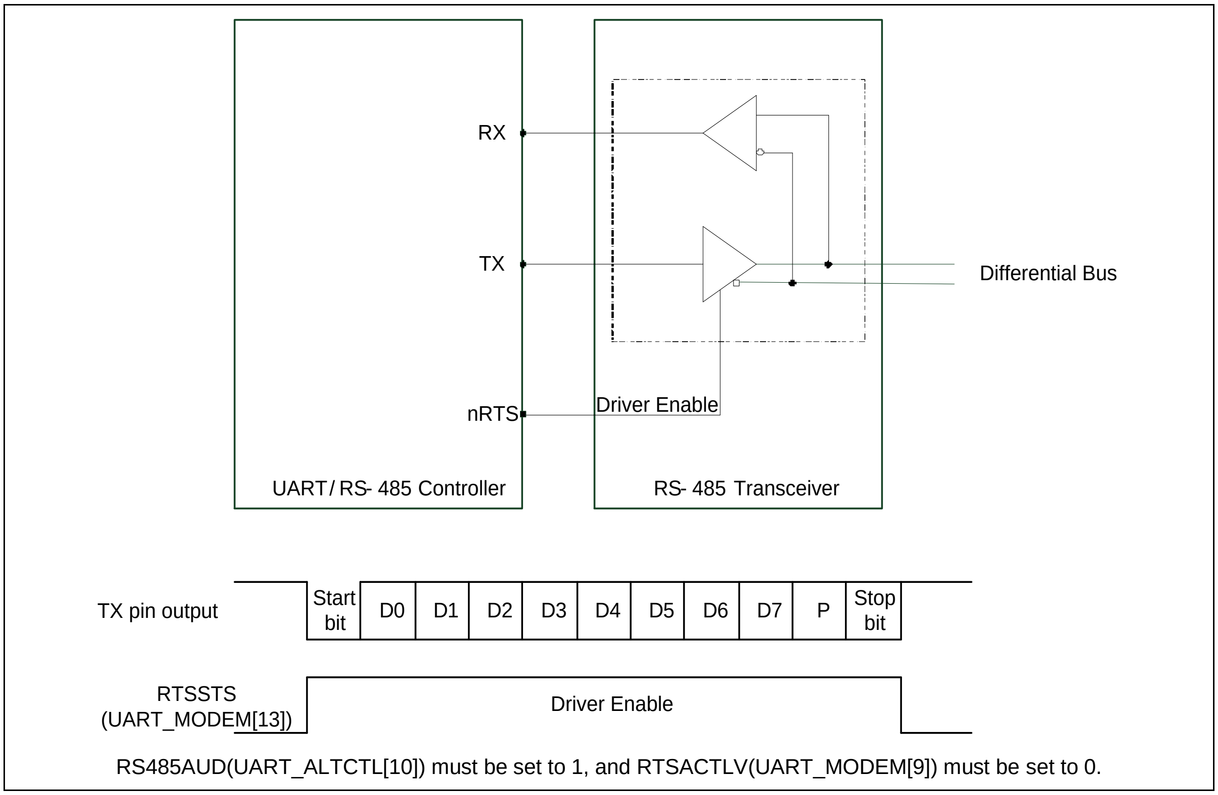

nRTS pin output status of RS-485 function mode
Set UART_MODEM[1]=0 by software
Set UART_MODEM[1]=1 by software
RTS control bit
(UART_MODEM[1])
Active
RTSSTS
(UART_MODEM[13])
RTS pin output
RTSACTLV = 0
Driver Enable
RTSACTLV = 1
(default)
RS485AUD (UART_ALTCTL[10]) = 0, support nRTS control bit by software control only.

Figure 6.10-22 RS-485 nRTS Driving Level with Software Control

**Programming Sequence Example:**

1. Program FUNCSEL in UART_FUNCSEL to select RS-485 function.
2. Program the RXOFF (UART_FIFO[8]) to determine enable or disable the receiver RS-485 receiver
3. Program the RS485NMM (UART_ALTCTL[8]) or RS485AAD (UART_ALTCTL[9]) mode.
4. If the RS485AAD (UART_ALTCTL[9]) mode is selected, the ADDRMV (UART_ALTCTL[31:24]) is programmed for auto address match value.
5. Determine auto direction control by programming RS485AUD (UART_ALTCTL[10]).



RS485AUD(UART_ALTCTL[10]) must be set to 1, and RTSACTLV(UART_MODEM[9]) must be set to 0.

Figure 6.10-23 Structure of RS-485 Frame

### 6.10.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UART Base Address:**<br>**UARTx_BA = 0x400E_C000+(x * 0x1000)**<br>**x=0,1,2** | | | | |
| **UART_DAT** | UARTx_BA+0x00 | R/W | UART Receive/Transmit Buffer Register | Undefined |
| **UART_INTEN** | UARTx_BA+0x04 | R/W | UART Interrupt Enable Register | 0x0000_0000 |
| **UART_FIFO** | UARTx_BA+0x08 | R/W | UART FIFO Control Register | 0x0000_0101 |
| **UART_LINE** | UARTx_BA+0x0C | R/W | UART Line Control Register | 0x0000_0000 |
| **UART_MODEM** | UARTx_BA+0x10 | R/W | UART Modem Control Register | 0x0000_0200 |
| **UART_MODEMSTS** | UARTx_BA+0x14 | R/W | UART Modem Status Register | 0x0000_0110 |
| **UART_FIFOSTS** | UARTx_BA+0x18 | R/W | UART FIFO Status Register | 0x1040_4000 |
| **UART_INTSTS** | UARTx_BA+0x1C | R/W | UART Interrupt Status Register | 0x0000_0002 |
| **UART_TOUT** | UARTx_BA+0x20 | R/W | UART Time-out Register | 0x0000_0000 |
| **UART_BAUD** | UARTx_BA+0x24 | R/W | UART Baud Rate Divisor Register | 0x0F00_0000 |
| **UART_IRDA** | UARTx_BA+0x28 | R/W | UART IrDA Control Register | 0x0000_0040 |
| **UART_ALTCTL** | UARTx_BA+0x2C | R/W | UART Alternate Control/Status Register | 0x0000_000C |
| **UART_FUNCSEL** | UARTx_BA+0x30 | R/W | UART Function Select Register | 0x0000_0000 |
| **UART_LINCTL** | UARTx_BA+0x34 | R/W | UART LIN Control Register * | 0x000C_0000 |
| **UART_LINSTS** | UARTx_BA+0x38 | R/W | UART LIN Status Register * | 0x0000_0000 |

**Note[*]:** Not available in UART0 Channel.

### 6.10.7 Register Description

#### UART Receive/Transmit Buffer Register (UART_DAT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UART_DAT** | UARTx_BA+0x00 | R/W | UART Receive/Transmit Buffer Register | Undefined |

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

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **DAT** | **Receiving/Transmit Buffer**<br>Write Operation:<br>By writing one byte to this register, the data byte will be stored in transmitter FIFO. The UART Controller will send out the data stored in transmitter FIFO top location through the UART_TXD.<br>Read Operation:<br>By reading this register, the UART will return an 8-bit data received from receiving FIFO. |

### UART Interrupt Enable Register (UART_INTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| UART_INTEN | UARTx_BA+0x04 | R/W | UART Interrupt Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| Reserved | | | | | ABRIEN | Reserved | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | | ATOCTSEN | ATORTSEN | TOCNTEN | WKDATIEN | WKCTSIEN | LINIEN |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Reserved | | BUFERRIEN | RXTOIEN | MODEMIEN | RLSIEN | THREIEN | RDAIEN |

| Bits | Description | |
|---|---|---|
| [31:19] | **Reserved** | Reserved. |
| [18] | **ABRIEN** | **Auto-baud Rate Interrupt Enable Control**<br>0 = Auto-baud rate interrupt Disabled.<br>1 = Auto-baud rate interrupt Enabled. |
| [17:14] | **Reserved** | Reserved. |
| [13] | **ATOCTSEN** | **nCTS Auto-flow Control Enable Control**<br>0 = nCTS auto-flow control Disabled.<br>1 = nCTS auto-flow control Enabled.<br>When nCTS auto-flow is enabled, the UART will send data to external device if nCTS input assert (UART will not send data to device until nCTS is asserted). |
| [12] | **ATORTSEN** | **nRTS Auto-flow Control Enable Control**<br>0 = nRTS auto-flow control Disabled.<br>1 = nRTS auto-flow control Enabled.<br>When nRTS auto-flow is enabled, if the number of bytes in the RX FIFO equals the RTSTRGLV (UART_FIFO[19:16]), the UART will de-assert nRTS signal. |
| [11] | **TOCNTEN** | **Time-out Counter Enable Control**<br>0 = Time-out counter Disabled.<br>1 = Time-out counter Enabled. |
| [10] | **WKDATIEN** | **Incoming Data Wake-up Interrupt Enable Control**<br>0 = Incoming data wake-up system function Disabled.<br>1 = Incoming data wake-up system function Enabled. When the system is in Power-down mode, incoming data will wake up system from Power-down mode..<br>Hardware will clear this bit when the incoming data wake-up operation finishes and “system clock” work stable. |
| [9] | **WKCTSIEN** | **nCTS Wake-up Interrupt Enable Control**<br>0 = nCTS wake-up system function Disabled. |

| | | |
|---|---|---|
| | | 1 = Wake-up system function Enabled. When the system is in Power-down mode, an external nCTS change will wake up system from Power-down mode. |
| [8] | **LINIEN** | **LIN Bus Interrupt Enable Control (Not Available in UART1/UART2)**<br>0 = LIN bus interrupt Disabled.<br>1 = LIN bus interrupt Enabled.<br>This bit is used for LIN function mode. |
| [7:6] | **Reserved** | Reserved. |
| [5] | **BUFERRIEN** | **Buffer Error Interrupt Enable Control**<br>0 = Buffer error interrupt Disabled.<br>1 = Buffer error interrupt Enabled. |
| [4] | **RXTOIEN** | **RX Time-out Interrupt Enable Control**<br>0 = RX time-out interrupt Disabled.<br>1 = RX time-out interrupt Enabled. |
| [3] | **MODEMIEN** | **Modem Status Interrupt Enable Control**<br>0 = Modem status interrupt Disabled.<br>1 = Modem status interrupt Enabled. |
| [2] | **RLSIEN** | **Receive Line Status Interrupt Enable Control**<br>0 = Receive Line Status interrupt Disabled.<br>1 = Receive Line Status interrupt Enabled. |
| [1] | **THREIEN** | **Transmit Holding Register Empty Interrupt Enable Control**<br>0 = Transmit holding register empty interrupt Disabled.<br>1 = Transmit holding register empty interrupt Enabled. |
| [0] | **RDAIEN** | **Receive Data Available Interrupt Enable Control**<br>0 = Receive data available interrupt Disabled.<br>1 = Receive data available interrupt Enabled. |

### UART FIFO Control Register (UART_FIFO)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| UART_FIFO | UARTx_BA+0x08 | R/W | UART FIFO Control Register | 0x0000_0101 |

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

| Bits | Description | |
|---|---|---|
| [31:20] | **Reserved** | Reserved. |
| [19:16] | **RTSTRGLV** | **nRTS Trigger Level for Auto-flow Control Use**<br>0000 = nRTS Trigger Level is 1 bytes.<br>0001 = nRTS Trigger Level is 4bytes.<br>0010 = nRTS Trigger Level is 8 bytes.<br>0011 = nRTS Trigger Level is 14 bytes.<br>0100 = nRTS Trigger Level is 30/14 (64 FIFO/16 FIFO).<br>0101 = nRTS Trigger Level is 46/14 (64 FIFO/16 FIFO).<br>0110 = nRTS Trigger Level is 62/14 (64 FIFO/16 FIFO).<br>Others = Reserved.<br>**Note:** This field is used for automatic nRTS flow control. |
| [15:9] | **Reserved** | Reserved. |
| [8] | **RXOFF** | **Receiver Disable Control**<br>The receiver is disabled or not (set 1 to disable receiver)<br>0 = Receiver Enabled.<br>1 = Receiver Disabled.<br>**Note:** This bit is used for RS-485 Normal Multi-drop mode. It should be programmed before RS485NMM (UART_ALTCTL [8]) is programmed. |
| [7:4] | **RFITL** | **RX FIFO Interrupt Trigger Level**<br>When the number of bytes in the receive FIFO equals the RFITL, the RDAIF will be set (if RDAIEN (UART_INTEN [0]) enabled, and an interrupt will be generated).<br>0000 = RX FIFO Interrupt Trigger Level is 1 byte.<br>0001 = RX FIFO Interrupt Trigger Level is 4 bytes.<br>0010 = RX FIFO Interrupt Trigger Level is 8 bytes.<br>0011 = RX FIFO Interrupt Trigger Level is 14 bytes.<br>0100 = RX FIFO Interrupt Trigger Level is 30/14 (64 FIFO/16 FIFO). |

| | | |
|---|---|---|
| | | 0101 = RX FIFO Interrupt Trigger Level is 46/14 (64 FIFO/16 FIFO).<br>0110 = RX FIFO Interrupt Trigger Level is 62/14 (64 FIFO/16 FIFO).<br>Others = Reserved. |
| [3] | **Reserved** | Reserved. |
| [2] | **TXRST** | **TX Field Software Reset Control**<br>When TXRST (UART_FIFO[2]) is set, all the byte in the transmit FIFO and TX internal state machine are cleared.<br>0 = No effect.<br>1 = Reset the TX internal state machine and pointers.<br>**Note:** This bit will automatically clear at least 3 UART peripheral clock cycles. |
| [1] | **RXRST** | **RX Field Software Reset Control**<br>When RXRST (UART_FIFO[1]) is set, all the byte in the receiver FIFO and RX internal state machine are cleared.<br>0 = No effect.<br>1 = Reset the RX internal state machine and pointers.<br>**Note:** This bit will automatically clear at least 3 UART peripheral clock cycles. |
| [0] | **Reserved** | Reserved. |

## UART Line Control Register (UART_LINE)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UART_LINE** | UARTx_BA+0x0C | R/W | UART Line Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:7] | **Reserved** | Reserved. |
| [6] | **BCB** | **Break Control**<br>0 = Break Control Disabled.<br>1 = Break Control Enabled.<br>**Note:** When this bit is set to logic 1, the serial data output (TX) is forced to the Spacing State (logic 0). This bit acts only on TX line and has no effect on the transmitter logic. |
| [5] | **SPE** | **Stick Parity Enable Control**<br>0 = Stick parity Disabled.<br>1 = Stick parity Enabled.<br>**Note:** If PBE (UART_LINE[3]) and EPE (UART_LINE[4]) are logic 1, the parity bit is transmitted and checked as logic 0. If PBE (UART_LINE[3]) is 1 and EPE (UART_LINE[4]) is 0 then the parity bit is transmitted and checked as 1. |
| [4] | **EPE** | **Even Parity Enable Control**<br>0 = Odd number of logic 1's is transmitted and checked in each word.<br>1 = Even number of logic 1's is transmitted and checked in each word.<br>**Note:** This bit is effective only when PBE (UART_LINE[3]) is set. |
| [3] | **PBE** | **Parity Bit Enable Control**<br>0 = No parity bit generated Disabled.<br>1 = Parity bit generated Enabled.<br>**Note:** Parity bit is generated on each outgoing character and is checked on each incoming data. |
| [2] | **NSB** | **Number of "STOP Bit"**<br>0 = One "STOP bit" is generated in the transmitted data.<br>1 = When select 5-bit word length, 1.5 "STOP bit" is generated in the transmitted data. When select 6-, 7- and 8-bit word length, 2 "STOP bit" is generated in the transmitted data. |
| [1:0] | **WLS** | **Word Length Selection** |

| | | |
|---|---|---|
| | | This field sets UART word length.<br>00 = 5 bits.<br>01 = 6 bits.<br>10 = 7 bits.<br>11 = 8 bits. |

### UART Modem Control Register (UART_MODEM)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UART_MODEM** | UARTx_BA+0x10 | R/W | UART Modem Control Register | 0x0000_0200 |

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

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13] | **RTSSTS** | **nRTS Pin Status (Read Only)**<br>This bit mirror from nRTS pin output of voltage logic status.<br>0 = nRTS pin output is low level voltage logic state.<br>1 = nRTS pin output is high level voltage logic state. |
| [12:10] | **Reserved** | Reserved. |
| [9] | **RTSACTLV** | **nRTS Pin Active Level**<br>This bit defines the active level state of nRTS pin output.<br>0 = nRTS pin output is high level active.<br>1 = nRTS pin output is low level active. (Default)<br>**Note1:** Refer to Figure 6.10-10 and Figure 6.10-11 for UART function mode.<br>**Note2:** Refer to Figure 6.10-21 and Figure 6.10-22 for RS-485 function mode. |
| [8:2] | **Reserved** | Reserved. |
| [1] | **RTS** | **nRTS (Request-to-send) Signal Control**<br>This bit is direct control internal nRTS signal active or not, and then drive the nRTS pin output with RTSACTLV bit configuration.<br>0 = nRTS signal is active.<br>1 = nRTS signal is inactive.<br>**Note1:** This nRTS signal control bit is not effective when nRTS auto-flow control is enabled in UART function mode.<br>**Note2:** This nRTS signal control bit is not effective when RS-485 auto direction mode (AUD) is enabled in RS-485 function mode. |
| [0] | **Reserved** | Reserved. |

### UART Modem Status Register (UART_MODEMSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UART_MODEMSTS** | UARTx_BA+0x14 | R/W | UART Modem Status Register | 0x0000_0110 |

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

| Bits | Description | |
|---|---|---|
| [31:9] | **Reserved** | Reserved. |
| [8] | **CTSACTLV** | **nCTS Pin Active Level**<br>This bit defines the active level state of nCTS pin input.<br>0 = nCTS pin input is high level active.<br>1 = nCTS pin input is low level active. (Default) |
| [7:5] | **Reserved** | Reserved. |
| [4] | **CTSSTS** | **nCTS Pin Status (Read Only)**<br>This bit mirror from nCTS pin input of voltage logic status.<br>0 = nCTS pin input is low level voltage logic state.<br>1 = nCTS pin input is high level voltage logic state.<br>**Note:** This bit echoes when UART Controller peripheral clock is enabled, and nCTS multi-function port is selected. |
| [3:1] | **Reserved** | Reserved. |
| [0] | **CTSDETF** | **Detect nCTS State Change Flag (Read Only)**<br>This bit is set whenever nCTS input has change state, and it will generate Modem interrupt to CPU when MODEMIEN (UART_INTEN [3]) is set to 1.<br>0 = nCTS input has not change state.<br>1 = nCTS input has change state.<br>**Note:** This bit is read only, but can be cleared by writing “1” to it. |

### UART FIFO Status Register (UART_FIFOSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| UART_FIFOSTS | UARTx_BA+0x18 | R/W | UART FIFO Status Register | 0x1040_4000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | TXEMPTYF | Reserved | | | TXOVIF |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| TXFULL | TXEMPTY | TXPTR | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RXFULL | RXEMPTY | RXPTR | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | BIF | FEF | PEF | ADDRDETF | ABRDTOIF | ABRDIF | RXOVIF |

| Bits | Description | |
|---|---|---|
| [31:29] | Reserved | Reserved. |
| [28] | TXEMPTYF | **Transmitter Empty Flag (Read Only)**<br>This bit is set by hardware when TX FIFO (UART_DAT) is empty and the STOP bit of the last byte has been transmitted.<br>0 = TX FIFO is not empty.<br>1 = TX FIFO is empty.<br>**Note:** This bit is cleared automatically when TX FIFO is not empty or the last byte transmission has not completed. |
| [27:25] | Reserved | Reserved. |
| [24] | TXOVIF | **TX Overflow Error Interrupt Flag (Read Only)**<br>If TX FIFO (UART_DAT) is full, an additional write to UART_DAT will cause this bit to logic 1.<br>0 = TX FIFO is not overflow.<br>1 = TX FIFO is overflow.<br>**Note:** This bit is read only, but can be cleared by writing “1” to it. |
| [23] | TXFULL | **Transmitter FIFO Full (Read Only)**<br>This bit indicates TX FIFO full or not.<br>0 = TX FIFO is not full.<br>1 = TX FIFO is full.<br>**Note:** This bit is set when the number of usage in TX FIFO Buffer is equal to 16/64, otherwise is cleared by hardware. |
| [22] | TXEMPTY | **Transmitter FIFO Empty (Read Only)**<br>This bit indicates TX FIFO empty or not.<br>0 = TX FIFO is not empty.<br>1 = TX FIFO is empty.<br>**Note:** When the last byte of TX FIFO has been transferred to Transmitter Shift Register, hardware sets this bit high. It will be cleared when writing data into DAT (TX FIFO not |

| | | |
|---|---|---|
| | | empty). |
| [21:16] | **TXPTR** | **TX FIFO Pointer (Read Only)**<br>This field indicates the TX FIFO Buffer Pointer. When CPU writes one byte into UART_DAT, TXPTR increases one. When one byte of TX FIFO is transferred to Transmitter Shift Register, TXPTR decreases one.<br>The Maximum value shown in TXPTR is 15/63. When the using level of TX FIFO Buffer equal to 16/64, the TXFULL bit is set to 1 and TXPTR will show 0. As one byte of TX FIFO is transferred to Transmitter Shift Register, the TXFULL bit is cleared to 0 and TXPTR will show 15/63. |
| [15] | **RXFULL** | **Receiver FIFO Full (Read Only)**<br>This bit initiates RX FIFO full or not.<br>0 = RX FIFO is not full.<br>1 = RX FIFO is full.<br>**Note:** This bit is set when the number of usage in RX FIFO Buffer is equal to 16/64, otherwise is cleared by hardware. |
| [14] | **RXEMPTY** | **Receiver FIFO Empty (Read Only)**<br>This bit initiate RX FIFO empty or not.<br>0 = RX FIFO is not empty.<br>1 = RX FIFO is empty.<br>**Note:** When the last byte of RX FIFO has been read by CPU, hardware sets this bit high. It will be cleared when UART receives any new data. |
| [13:8] | **RXPTR** | **RX FIFO Pointer (Read Only)**<br>This field indicates the RX FIFO Buffer Pointer. When UART receives one byte from external device, RXPTR increases one. When one byte of RX FIFO is read by CPU, RXPTR decreases one.<br>The Maximum value shown in RXPTR is 15/63. When the using level of RX FIFO Buffer equal to 16/64, the RXFULL bit is set to 1 and RXPTR will show 0. As one byte of RX FIFO is read by CPU, the RXFULL bit is cleared to 0 and RXPTR will show 15/63. |
| [7] | **Reserved** | Reserved. |
| [6] | **BIF** | **Break Interrupt Flag (Read Only)**<br>This bit is set to logic 1 whenever the received data input (RX) is held in the “spacing state” (logic 0) for longer than a full word transmission time (that is, the total time of “start bit” + data bits + parity + stop bits).<br>0 = No Break interrupt is generated.<br>1 = Break interrupt is generated.<br>**Note:** This bit is read only, but can be cleared by writing ‘1’ to it. |
| [5] | **FEF** | **Framing Error Flag (Read Only)**<br>This bit is set to logic 1 whenever the received character does not have a valid “stop bit” (that is, the stop bit following the last data bit or parity bit is detected as logic 0).<br>0 = No framing error is generated.<br>1 = Framing error is generated.<br>**Note:** This bit is read only, but can be cleared by writing ‘1’ to it. |
| [4] | **PEF** | **Parity Error Flag (Read Only)**<br>This bit is set to logic 1 whenever the received character does not have a valid “parity bit”.<br>0 = No parity error is generated.<br>1 = Parity error is generated.<br>**Note:** This bit is read only, but can be cleared by writing ‘1’ to it. |
| [3] | **ADDRDETF** | **RS-485 Address Byte Detect Flag (Read Only)** |

| | | |
|---|---|---|
| | | 0 = Receiver detects a data that is not an address bit (bit 9 =’0’).<br>1 = Receiver detects a data that is an address bit (bit 9 =’1’).<br>**Note1:** This field is used for RS-485 function mode and ADDRDEN (UART_ALTCTL[15]) is set to 1 to enable Address detection mode.<br>**Note2:** This bit is read only, but can be cleared by writing ‘1’ to it. |
| [2] | **ABRDTOIF** | **Auto-baud Rate Time-out Interrupt (Read Only)**<br>0 = Auto-baud rate counter is underflow.<br>1 = Auto-baud rate counter is overflow.<br>**Note1:** This bit is set to logic “1” in Auto-baud Rate Detect mode and the baud rate counter is overflow.<br>**Note2:** This bit is read only, but can be cleared by writing “1” to it. |
| [1] | **ABRDIF** | **Auto-baud Rate Detect Interrupt (Read Only)**<br>0 = Auto-baud rate detect function is not finished.<br>1 = Auto-baud rate detect function is finished.<br>**Note1:** This bit is set to logic “1” when auto-baud rate detect function is finished.<br>**Note2:** This bit is read only, but can be cleared by writing “1” to it. |
| [0] | **RXOVIF** | **RX Overflow Error Interrupt Flag (Read Only)**<br>This bit is set when RX FIFO overflow.<br>If the number of bytes of received data is greater than RX_FIFO (UART_DAT) size, 16/64 bytes this bit will be set.<br>0 = RX FIFO is not overflow.<br>1 = RX FIFO is overflow.<br>**Note:** This bit is read only, but can be cleared by writing “1” to it. |

### UART Interrupt Status Control Register (UART_INTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UART_INTSTS** | UARTx_BA+0x1C | R/W | UART Interrupt Status Register | 0x0000_0002 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | **DATWKIF** | **CTSWKIF** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **LININT** | **WKINT** | **BUFERRINT** | **RXTOINT** | **MODEMINT** | **RLSINT** | **THREINT** | **RDAINT** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **LINIF** | **WKIF** | **BUFERRIF** | **RXTOIF** | **MODEMIF** | **RLSIF** | **THREIF** | **RDAIF** |

| Bits | Description | |
|---|---|---|
| [31:18] | **Reserved** | Reserved. |
| [17] | **DATWKIF** | **Data Wake-up Interrupt Flag (Read Only)**<br>This bit is set if chip wake-up from Power-down state by data wake-up.<br>0 = Chip stays in Power-down state.<br>1 = Chip wake-up from Power-down state by data wake-up.<br>**Note1:** If WKDATIEN (UART_INTEN[10]) is enabled, the wake-up interrupt is generated.<br>**Note2:** This bit is read only, but can be cleared by writing ‘1’ to it. |
| [16] | **CTSWKIF** | **nCTS Wake-up Interrupt Flag (Read Only)**<br>0 = Chip stays in Power-down state.<br>1 = Chip wake-up from Power-down state by nCTS wake-up.<br>**Note1:** If WKCTSIEN (UART_INTEN[9])is enabled, the wake-up interrupt is generated.<br>**Note2:** This bit is read only, but can be cleared by writing ‘1’ to it. |
| [15] | **LININT** | **LIN Bus Interrupt Indicator (Read Only)(Not Available in UART0 Channel)**<br>This bit is set if LINIEN (UART_INTEN[8]) and LIN IF(UART_INTSTS[7]) are both set to 1.<br>0 = No LIN Bus interrupt is generated.<br>1 = The LIN Bus interrupt is generated. |
| [14] | **WKINT** | **UART Wake-up Interrupt Indicator (Read Only)**<br>This bit is set when DATWKIF or CTSWKIF is set to 1.<br>0 = NO data or nCTS wake-up interrupt are generated.<br>1 = Data or nCTS wake-up interrupt are generated. |
| [13] | **BUFERRINT** | **Buffer Error Interrupt Indicator (Read Only)**<br>This bit is set if BFERRIEN(UART_INTEN[5] and BUFERRIF(UART_INTSTS[5]) are both set to 1.<br>0 = No buffer error interrupt is generated.<br>1 = Buffer error interrupt is generated. |

| | | |
|---|---|---|
| [12] | **RXTOINT** | **Time-out Interrupt Indicator (Read Only)**<br>This bit is set if TOUTIEN(UART_INTEN[4]) and RXTOIF(UART_INTSTS[4]) are both set to 1.<br>0 = No Tout interrupt is generated.<br>1 = Tout interrupt is generated. |
| [11] | **MODEMINT** | **MODEM Status Interrupt Indicator (Read Only)**<br>This bit is set if MODEMIEN(UART_INTEN[3] and MODEMIF(UART_INTSTS[4]) are both set to 1<br>0 = No Modem interrupt is generated.<br>1 = Modem interrupt is generated.. |
| [10] | **RLSINT** | **Receive Line Status Interrupt Indicator (Read Only)**<br>This bit is set if RLSIEN (UART_INTEN[2]) and RLSIF(UART_INTSTS[2]) are both set to 1.<br>0 = No RLS interrupt is generated.<br>1 = RLS interrupt is generated. |
| [9] | **THREINT** | **Transmit Holding Register Empty Interrupt Indicator (Read Only)**<br>This bit is set if THREIEN (UART_INTEN[1])and THREIF(UART_INTSTS[1]) are both set to 1.<br>0 = No DATE interrupt is generated.<br>1 = DATE interrupt is generated. |
| [8] | **RDAINT** | **Receive Data Available Interrupt Indicator (Read Only)**<br>This bit is set if RDAIEN (UART_INTEN[0]) and RDAIF (UART_INTSTS[0]) are both set to 1.<br>0 = No RDA interrupt is generated.<br>1 = RDA interrupt is generated. |
| [7] | **LINIF** | **LIN Bus Interrupt Flag (Read Only) (Not Available in UART0 Channel)**<br>This bit is set when LIN slave header detect (SLVHDETF (UART_LINSTS[0] =1)), LIN break detect (BRKDETF(UART_LINSTS[9]=1)), bit error detect (BITEF(UART_LINSTS[9]=1), LIN slave ID parity error (SLVIDPEF(UART_LINSTS[2] = 1) or LIN slave header error detect (SLVHEF (UART_LINSTS[1])). If LIN_ IEN (UART_INTEN [8]) is enabled the LIN interrupt will be generated.<br>0 = None of SLVHDETF, BRKDETF, BITEF, SLVIDPEF and SLVHEF is generated.<br>1 = At least one of SLVHDETF, BRKDETF, BITEF, SLVIDPEF and SLVHEF is generated.<br>**Note:** This bit is read only. This bit is cleared when SLVHDETF(UART_LINSTS[0]), BRKDETF(UART_LINSTS[8]), BITEF(UART_LINSTS[9]), SLVIDPEF (UART_LINSTS[2]), SLVHEF(UART_LINSTS[1]) and SLVSYNCF(UART_LINSTS[3]) all are cleared. |
| [6] | **WKIF** | **UART Wake-up Interrupt Flag (Read Only)**<br>This bit is set when DATWKIF (UART_INTSTS[17]) or CTSWKIF(UART_INTSTS[16]) is set to 1.<br>0 = No DATWKIF and CTSWKIF are generated.<br>1 = DATWKIF or CTSWKIF.<br>**Note:** This bit is read only. This bit is cleared if both of DATWKIF (UART_INTSTS[17]) and CTSWKIF(UART_INTSTS[16]) are cleared to 0 by writing 1 to DATWKIF (UART_INTSTS[17]) and CTSWKIF (UART_INTSTS[17]). |
| [5] | **BUFERRIF** | **Buffer Error Interrupt Flag (Read Only)**<br>This bit is set when the TX FIFO or RX FIFO overflows (TXOVIF (UART_FIFOSTS[24]) or RXOVIF (UART_FIFOSTS[0]) is set). When BUFERRIF (UART_INTSTS[5])is set, the transfer is not correct. If BFERRIEN (UART_INTEN [8]) is enabled, the buffer error |

|  |  |  |
|---|---|---|
|  |  | interrupt will be generated.<br>0 = No buffer error interrupt flag is generated.<br>1 = Buffer error interrupt flag is generated.<br>**Note:** This bit is read only. This bit is cleared if both of RXOVIF(UART_FIFOSTS[0]) and TXOVIF(UART_FIFOSTS[24]) are cleared to 0 by writing 1 to RXOVIF(UART_FIFOSTS[0]) and TXOVIF(UART_FIFOSTS[24]). |
| [4] | **RXTOIF** | **Time-out Interrupt Flag (Read Only)**<br>This bit is set when the RX FIFO is not empty and no activities occurred in the RX FIFO and the time-out counter equal to TOIC. If TOUTIEN (UART_INTEN [4]) is enabled, the Tout interrupt will be generated.<br>0 = No Time-out interrupt flag is generated.<br>1 = Time-out interrupt flag is generated.<br>**Note:** This bit is read only and user can read UART_DAT (RX is in active) to clear it. |
| [3] | **MODEMIF** | **MODEM Interrupt Flag (Read Only) Channel**<br>This bit is set when the nCTS pin has state change (CTSDETF (UART_MODEMSTS[0]) = 1). If MODEMIEN (UART_INTEN [3]) is enabled, the Modem interrupt will be generated.<br>0 = No Modem interrupt flag is generated.<br>1 = Modem interrupt flag is generated.<br>**Note:** This bit is read only and reset to 0 when bit CTSDETF is cleared by a write 1 on CTSDETF(UART_MODEMSTS[0]). |
| [2] | **RLSIF** | **Receive Line Interrupt Flag (Read Only)**<br>This bit is set when the RX receive data have parity error, frame error or break error (at least one of 3 bits, BIF(UART_FIFOSTS[6]), FEF(UART_FIFOSTS[5]) and PEF(UART_FIFOSTS[4]), is set). If RLSIEN (UART_INTEN [2]) is enabled, the RLS interrupt will be generated.<br>0 = No RLS interrupt flag is generated.<br>1 = RLS interrupt flag is generated.<br>**Note1:** In RS-485 function mode, this field is set include “receiver detect and received address byte character (bit9 = ‘1’) bit". At the same time, the bit of ADDRDETF (UART_FIFOSTS[3]) is also set.<br>**Note2:** This bit is read only and reset to 0 when all bits of BIF (UART_FIFOSTS[6]), FEF(UART_FIFOSTS[5]) and PEF(UART_FIFOSTS[4]) are cleared.<br>**Note3:** In RS-485 function mode, this bit is read only and reset to 0 when all bits of BIF (UART_FIFOSTS[6]) , FEF(UART_FIFOSTS[5]) and PEF(UART_FIFOSTS[4]) and ADDRDETF (UART_FIFOSTS[3]) are cleared. |
| [1] | **THREIF** | **Transmit Holding Register Empty Interrupt Flag (Read Only)**<br>This bit is set when the last data of TX FIFO is transferred to Transmitter Shift Register. If THREIEN (UART_INTEN[1]) is enabled, the THRE interrupt will be generated.<br>0 = No THRE interrupt flag is generated.<br>1 = THRE interrupt flag is generated.<br>**Note:** This bit is read only and it will be cleared when writing data into UART_DAT (TX FIFO not empty). |
| [0] | **RDAIF** | **Receive Data Available Interrupt Flag (Read Only)**<br>When the number of bytes in the RX FIFO equals the RFITL then the RDAIF(UART_INTSTS[0]) will be set. If RDAIEN (UART_INTEN [0]) is enabled, the RDA interrupt will be generated.<br>0 = No RDA interrupt flag is generated.<br>1 = RDA interrupt flag is generated.<br>**Note:** This bit is read only and it will be cleared when the number of unread bytes of RX FIFO drops below the threshold level (RFITL(UART_FIFO[7:4]). |

### UART Time-out Register (UART_TOUT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UART_TOUT** | UARTx_BA+0x20 | R/W | UART Time-out Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:8] | **DLY** | **TX Delay Time Value**<br>This field is used to programming the transfer delay time between the last stop bit and next start bit. The unit is bit time. |
| [7:0] | **TOIC** | **Time-out Interrupt Comparator**<br>The time-out counter resets and starts counting (the counting clock = baud rate) whenever the RX FIFO receives a new data word. Once the content of time-out counter is equal to that of time-out interrupt comparator (TOIC (UART_TOUT[7:0])), a receiver time-out interrupt (RXTOINT(UART_INTSTS[12])) is generated if RXTOIEN (UART_INTEN [4]) enabled. A new incoming data word or RX FIFO empty will clear RXTOINT(UART_INTSTS[12]). In order to avoid receiver time-out interrupt generation immediately during one character is being received, TOIC value should be set between 40 and 255. Thus, for example, if TOIC is set with 40, the time-out interrupt is generated after four characters are not received when 1 stop bit and no parity check is set for UART transfer. |

### UART Baud Rate Divider Register (UART_BAUD)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UART_BAUD** | UARTx_BA+0x24 | R/W | UART Baud Rate Divisor Register | 0x0F00_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | BAUDM1 | BAUDM0 | EDIVM1 | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| BRD | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| BRD | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:30] | **Reserved** | Reserved. |
| [29] | **BAUDM1** | **BAUD Rate Mode Selection Bit 1**<br>This bit is baud rate mode selection bit 1. UART provides three baud rate calculation modes. This bit combines with BAUDM0 (UART_BAUD[28]) to select baud rate calculation mode. The detail description is shown in Table 6.10-2.<br>In IrDA mode must be operated in mode 0. |
| [28] | **BAUDM0** | **BAUD Rate Mode Selection Bit 0**<br>This bit is baud rate mode selection bit 0. UART provides three baud rate calculation modes. This bit combines with BAUDM1 (UART_BAUD[29]) to select baud rate calculation mode. The detail description is shown in Table 6.10-2. |
| [27:24] | **EDIVM1** | **Extra Divider for BAUD Rate Mode 1**<br>This field is used for baud rate calculation in mode 1 and has no effect for baud rate calculation in mode 0 and mode 2. The detail description is shown in Table 6.10-2. |
| [23:16] | **Reserved** | Reserved. |
| [15:0] | **BRD** | **Baud Rate Divider**<br>The field indicates the baud rate divider. This filed is used in baud rate calculation. The detail description is shown in Table 6.10-2. |

### UART_IrDA Control Register (UART_IRDA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UART_IRDA** | UARTx_BA+0x28 | R/W | UART IrDA Control Register | 0x0000_0040 |

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

| Bits | Description | |
|---|---|---|
| [31:7] | **Reserved** | Reserved. |
| [6] | **RXINV** | **IrDA Inverse Receive Input Signal**<br>0 = None inverse receiving input signal.<br>1 = Inverse receiving input signal. (Default) |
| [5] | **TXINV** | **IrDA Inverse Transmitting Output Signal**<br>0 = None inverse transmitting signal. (Default)<br>1 = Inverse transmitting output signal. |
| [4:2] | **Reserved** | Reserved. |
| [1] | **TXEN** | **IrDA Receiver/Transmitter Selection Enable Control**<br>0 = IrDA Transmitter Disabled and Receiver Enabled. (Default)<br>1 = IrDA Transmitter Enabled and Receiver Disabled.<br>In IrDA function mode (FUNCSEL(UART_FUNCSEL[1:0])=10), the first received data is unreliable and it should be skipped if IrDA receiver is enabled (TXEN(UART_IRDA[1])=0) at the first time. |
| [0] | **Reserved** | Reserved. |

**Note:** In IrDA mode, the BAUDM1 (UART_BAUD [29]) register must be disabled, the baud equation must be Clock / (16 * (BRD + 2)).

### UART Alternate Control/Status Register (UART_ALTCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UART_ALTCTL** | UARTx_BA+0x2C | R/W | UART Alternate Control/Status Register | 0x0000_000C |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **ADDRMV** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | **ABRDBITS** | | **ABRDEN** | **ABRIF** | **Reserved** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **ADDRDEN** | **Reserved** | | | | **RS485AUD** | **RS485AAD** | **RS485NMM** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **LINTXEN** | **LINRXEN** | **Reserved** | | **BRKFL** | | | |

| Bits | Description | |
|---|---|---|
| [31:24] | **ADDRMV** | **Address Match Value**<br>This field contains the RS-485 address match values.<br>This field is used for RS-485 auto address detection mode. |
| [23:21] | **Reserved** | Reserved. |
| [20:19] | **ABRDBITS** | **Auto-baud Rate Detect Bit Length**<br>00 = 1-bit time from Start bit to the 1st rising edge. The input pattern shall be 0x01.<br>01 = 2-bit time from Start bit to the 1st rising edge. The input pattern shall be 0x02.<br>10 = 4-bit time from Start bit to the 1st rising edge. The input pattern shall be 0x08.<br>11 = 8-bit time from Start bit to the 1st rising edge. The input pattern shall be 0x80.<br>**Note:** The calculation of bit number includes the START bit. |
| [18] | **ABRDEN** | **Auto-baud Rate Detect Enable Control**<br>0 = Auto-baud rate detect function Disabled.<br>1 = Auto-baud rate detect function Enabled.<br>**Note:** This bit is cleared automatically after auto-baud detection is finished. |
| [17] | **ABRIF** | **Auto-baud Rate Interrupt Flag (Read Only)**<br>This bit is set when auto-baud rate detection function finished or the auto-baud rate counter was overflow and if ABRIEN(UART_INTEN [18]) is set then the auto-baud rate interrupt will be generated.<br>**Note:** This bit is read only, but it can be cleared by writing “1” to ABRDTOIF (UART_FIFOSTS[2]) and ABRDIF(UART_FIFOSTS[1]). |
| [16] | **Reserved** | Reserved. |
| [15] | **ADDRDEN** | **RS-485 Address Detection Enable Control**<br>This bit is used to enable RS-485 Address Detection mode.<br>0 = Address detection mode Disabled.<br>1 = Address detection mode Enabled.<br>**Note:** This bit is used for RS-485 any operation mode. |

| | | |
|---|---|---|
| [14:11] | **Reserved** | Reserved. |
| [10] | **RS485AUD** | **RS-485 Auto Direction Function (AUD)**<br>0 = RS-485 Auto Direction Operation function (AUD) Disabled.<br>1 = RS-485 Auto Direction Operation function (AUD) Enabled.<br>**Note:** It can be active with RS-485_AAD or RS-485_NMM operation mode. |
| [9] | **RS485AAD** | **RS-485 Auto Address Detection Operation Mode (AAD)**<br>0 = RS-485 Auto Address Detection Operation mode (AAD) Disabled.<br>1 = RS-485 Auto Address Detection Operation mode (AAD) Enabled.<br>It cannot be active with RS-485_NMM operation mode. |
| [8] | **RS485NMM** | **RS-485 Normal Multi-drop Operation Mode (NMM)**<br>0 = RS-485 Normal Multi-drop Operation mode (NMM) Disabled.<br>1 = RS-485 Normal Multi-drop Operation mode (NMM) Enabled.<br>**Note:** It cannot be active with RS-485_AAD operation mode. |
| [7] | **LINTXEN** | **LIN TX Break Mode Enable Control (Only Available in UART1/UART2 Channel)**<br>0 = LIN TX Break mode Disabled.<br>1 = LIN TX Break mode Enabled.<br>**Note:** When TX break field transfer operation finished, this bit will be cleared automatically. |
| [6] | **LINRXEN** | **LIN RX Enable Control (Only Available in UART1/UART2 Channel)**<br>0 = LIN RX mode Disabled.<br>1 = LIN RX mode Enabled. |
| [5:4] | **Reserved** | Reserved. |
| [3:0] | **BRKFL** | **UART LIN Break Field Length (Only Available in UART1/UART2 Channel)**<br>This field indicates a 4-bit LIN TX break field count.<br>**Note1:** This break field length is BRKFL + 1<br>**Note2:** According to LIN spec, the reset value is 0xC (break field length = 13). |

### UART Function Select Register (UART_FUNCSEL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UART_FUNCSEL** | UARTx_BA+0x30 | R/W | UART Function Select Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1:0] | **FUNCSEL** | **Function Select**<br>00 = UART function.<br>01 = LIN function (Only Available in UART1/UART2 Channel).<br>10 = IrDA function.<br>11 = RS-485 function.<br>**Note:** In IrDA function mode (FUNCSEL(UART_FUNCSEL[1:0])=10), the first received data is unreliable and it should be skipped if IrDA receiver is enabled (TXEN(UART_IRDA[1])=0) at the first time. |

## UART LIN Control Register (UART_LINCTL) (Not Available in UART0 Channel)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| UART_LINCTL | UARTx_BA+0x34 | R/W | UART LIN Control Register * | 0x000C_0000 |

**Note[*]:** Not Available in UART0 Channel.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| PID | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| HSEL | | BSL | | BRKFL | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | | | BITERREN | RXOFF | BRKDETEN | IDPEN | SENDH |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Reserved | | | MUTE | SLVDUEN | SLVAREN | SLVHDEN | SLVEN |

| Bits | Description | |
|---|---|---|
| [31:24] | **PID** | **LIN PID Bits**<br>This field contains the LIN frame ID value in LIN function mode, the frame ID parity can be generated by software or hardware depending on whether IDPEN (UART_LINCTL[9]) = 1.<br>If the parity generated by hardware, and user fills in ID0~ID5, (PID [29:24]), hardware will calculate P0 (PID[30]) and P1 (PID[31]), otherwise user must fill a frame ID and parity in this field.<br>**Note1:** User can fill in any 8-bit value to this field and the bit 24 indicates ID0 (LSB first).<br>**Note2:** This field can be used for LIN Master mode or Slave mode. |
| [23:22] | **HSEL** | **LIN Header Select**<br>00 = The LIN header includes “break field”.<br>01 = The LIN header includes “break field” and “sync field”.<br>10 = The LIN header includes “break field”, “sync field” and “frame ID field”.<br>11 = Reserved.<br>**Note:** This bit is used to master mode for LIN to send header field (SENDH (UART_LINCTL [8]) = 1) or used to slave to indicates exit from mute mode condition (MUTE (UART_LINCTL[4] = 1). |
| [21:20] | **BSL** | **LIN Break/Sync Delimiter Length**<br>00 = The LIN break/sync delimiter length is 1-bit time.<br>01 = The LIN break/sync delimiter length is 2-bit time.<br>10 = The LIN break/sync delimiter length is 3-bit time.<br>11 = The LIN break/sync delimiter length is 4-bit time.<br>**Note:** This bit is used for LIN master to send header field. |
| [19:16] | **BRKFL** | **LIN Break Field Length**<br>This field indicates a 4-bit LIN TX break field count.<br>**Note1:** These registers are shadow registers of BRKFL, User can read/write it by setting BRKFL (UART_ALTCTL[3:0]) or BRKFL (UART_LINCTL[19:16]).<br>**Note2:** This break field length is BRKFL + 1.<br>**Note3:** According to LIN spec, the reset value is 12 (break field length = 13). |

| | | |
|---|---|---|
| [15:13] | **Reserved** | Reserved. |
| [12] | **BITERREN** | **Bit Error Detect Enable Control**<br>0 = Bit error detection function Disabled.<br>1 = Bit error detection Enabled.<br>**Note:** In LIN function mode, when a bit error occurs, the BITEF (UART_LINSTS[9]) flag will be asserted. If the LINIEN (UART_INTEN[8]) = 1, an interrupt will be generated. |
| [11] | **RXOFF** | **LIN Receiver Disable Control**<br>If the receiver is enabled (RXOFF (UART_LINCTL[11]) = 0), all received byte data will be accepted and stored in the RX-FIFO, and if the receiver is disabled (RXOFF (UART_LINCTL[11] = 1), all received byte data will be ignore.<br>0 = LIN receiver Enabled.<br>1 = LIN receiver Disabled.<br>**Note:** This bit is only valid when operating in LIN function mode (FUNCSEL (UART_FUNCSEL[1:0]) = 01). |
| [10] | **BRKDETEN** | **LIN Break Detection Enable Control**<br>When detect consecutive dominant greater than 11 bits, and are followed by a delimiter character, the BRKDETF (UART_LINSTS[8]) flag is set in UART_LINSTS register at the end of break field. If the LINIEN (UART_INTEN [8])=1, an interrupt will be generated.<br>0 = LIN break detection Disabled.<br>1 = LIN break detection Enabled. |
| [9] | **IDPEN** | **LIN ID Parity Enable Control**<br>0 = LIN frame ID parity Disabled.<br>1 = LIN frame ID parity Enabled.<br>**Note1:** This bit can be used for LIN master to sending header field (SENDH (UART_LINCTL[8])) = 1 and HSEL (UART_LINCTL[23:22]) = 10) or be used for enable LIN slave received frame ID parity checked.<br>**Note2:** This bit is only used when the operation header transmitter is in HSEL (UART_LINCTL[23:22]) = 10. |
| [8] | **SENDH** | **LIN TX Send Header Enable Control**<br>The LIN TX header can be “break field” or “break and sync field” or “break, sync and frame ID field”, it is depend on setting HSEL (UART_LINCTL[23:22]).<br>0 = Send LIN TX header Disabled.<br>1 = Send LIN TX header Enabled.<br>**Note1:** These registers are shadow registers of LINTXEN (UART_ALTCTL [7]); user can read/write it by setting LINTXEN (UART_ALTCTL [7]) or SENDH (UART_LINCTL [8]).<br>**Note2:** When transmitter header field (it may be “break” or “break + sync” or “break + sync + frame ID” selected by HSEL (UART_LINCTL[23:22]) field) transfer operation finished, this bit will be cleared automatically. |
| [7:5] | **Reserved** | Reserved. |
| [4] | **MUTE** | **LIN Mute Mode Enable Control**<br>0 = LIN mute mode Disabled.<br>1 = LIN mute mode Enabled.<br>**Note:** The exit from mute mode condition and each control and interactions of this field are explained in 6.10.5.9 (LIN Slave mode). |
| [3] | **SLVDUEN** | **LIN Slave Divider Update Method Enable Control**<br>0 = UART_BAUD updated is written by software (if no automatic resynchronization update occurs at the same time).<br>1 = UART_BAUD is updated at the next received character. User must set the bit before checksum reception. |

| | | |
|---|---|---|
| | | **Note1:** This bit is only valid in LIN Slave mode (SLVEN (UART_LINCTL[0]) = 1).<br>**Note2:** This bit is used for LIN Slave Automatic Resynchronization mode. (for Non-Automatic Resynchronization mode, this bit should be kept cleared)<br>**Note3:** The control and interactions of this field are explained in 6.10.5.9 (Slave mode with automatic resynchronization). |
| [2] | **SLVAREN** | **LIN Slave Automatic Resynchronization Mode Enable Control**<br>0 = LIN automatic resynchronization Disabled.<br>1 = LIN automatic resynchronization Enabled.<br>**Note1:** This bit is only valid in LIN Slave mode (SLVEN (UART_LINCTL[0]) = 1).<br>**Note2:** When operating in Automatic Resynchronization mode, the baud rate setting must be mode2 (BAUDM1 (UART_BAUD [29]) and BAUDM0 (UART_BAUD [28]) must be 1).<br>**Note3:** The control and interactions of this field are explained in 6.10.5.9(Slave mode with automatic resynchronization). |
| [1] | **SLVHDEN** | **LIN Slave Header Detection Enable Control**<br>0 = LIN slave header detection Disabled.<br>1 = LIN slave header detection Enabled.<br>**Note1:** This bit is only valid in LIN Slave mode (SLVEN (UART_LINCTL[0]) = 1).<br>**Note2:** In LIN function mode, when detect header field (break + sync + frame ID), SLVHDETF (UART_LINSTS [0]) flag will be asserted. If the LINIEN (UART_INTEN[8]) = 1, an interrupt will be generated. |
| [0] | **SLVEN** | **LIN Slave Mode Enable Control**<br>0 = LIN Slave mode Disabled.<br>1 = LIN Slave mode Enabled. |

### UART LIN Status Register (UART_LINSTS) (Not Available in UART0 Channel)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **UART_LINSTS** | UARTx_BA+0x38 | R/W | UART LIN Status Register * | 0x0000_0000 |

**Note[*]:** Not Available in UART0 Channel.

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | **BITEF** | **BRKDETF** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | **SLVSYNCF** | **SLVIDPEF** | **SLVHEF** | **SLVHDETF** |

| Bits | Description | |
|---|---|---|
| [31:10] | **Reserved** | Reserved. |
| [9] | **BITEF** | **Bit Error Detect Status Flag (Read Only)**<br>At TX transfer state, hardware will monitoring the bus state, if the input pin (SIN) state not equals to the output pin (SOUT) state, BITEF (UART_LINSTS[9]) will be set.<br>When occur bit error, if the LINIEN (UART_INTEN[8]) = 1, an interrupt will be generated.<br>**Note1:** This bit is read only, but it can be cleared by writing 1 to it.<br>**Note2:** This bit is only valid when enable bit error detection function (BITERREN (UART_LINCTL [12]) = 1). |
| [8] | **BRKDETF** | **LIN Break Detection Flag (Read Only)**<br>This bit is set by hardware when a break is detected and be cleared by writing 1 to it through software.<br>0 = LIN break not detected.<br>1 = LIN break detected.<br>**Note1:** This bit is read only, but it can be cleared by writing 1 to it.<br>**Note2:** This bit is only valid when LIN break detection function is enabled (BRKDETEN (UART_LINCTL[10]) =1). |
| [7:4] | **Reserved** | Reserved. |
| [3] | **SLVSYNCF** | **LIN Slave Sync Field (Read Only)**<br>This bit indicates that the LIN sync field is being analyzed in Automatic Resynchronization mode. When the receiver header have some error been detect, user must reset the internal circuit to re-search new frame header by writing 1 to this bit.<br>0 = The current character is not at LIN sync state.<br>1 = The current character is at LIN sync state.<br>**Note1:** This bit is only valid in LIN Slave mode (SLVEN(UART_LINCTL[0]) = 1).<br>**Note2:** This bit is read only, but it can be cleared by writing 1 to it.<br>**Note3:** When writing 1 to it, hardware will reload the initial baud rate and re-search a new frame header. |

| | | |
|---|---|---|
| [2] | **SLVIDPEF** | **LIN Slave ID Parity Error Flag**<br>This bit is set by hardware when receipted frame ID parity is not correct.<br>0 = No active.<br>1 = Receipted frame ID parity is not correct.<br>**Note1:** This bit is read only, but it can be cleared by writing 1 to it.<br>**Note2:** This bit is only valid in LIN Slave mode (SLVEN (UART_LINCTL [0])= 1) and enable LIN frame ID parity check function IDPEN (UART_LINCTL [9]). |
| [1] | **SLVHEF** | **LIN Slave Header Error Flag (Read Only)**<br>This bit is set by hardware when a LIN header error is detected in LIN Slave mode and be cleared by writing 1 to it. The header errors include “break delimiter is too short (less than 0.5 bit time)”, “frame error in sync field or Identifier field”, “sync field data is not 0x55 in Non-Automatic Resynchronization mode”, “sync field deviation error with Automatic Resynchronization mode”, “sync field measure time-out with Automatic Resynchronization mode” and “LIN header reception time-out”.<br>0 = LIN header error not detected.<br>1 = LIN header error detected.<br>**Note1:** This bit is read only, but it can be cleared by writing 1 to it.<br>**Note2:** This bit is only valid when UART is operated in LIN Slave mode (SLVEN (UART_LINCTL [0]) = 1) and enables LIN slave header detection function (SLVHDEN (UART_LINCTL [1])). |
| [0] | **SLVHDETF** | **LIN Slave Header Detection Flag (Read Only)**<br>This bit is set by hardware when a LIN header is detected in LIN Slave mode and be cleared by writing 1 to it.<br>0 = LIN header not detected.<br>1 = LIN header detected (break + sync + frame ID).<br>**Note1:** This bit is read only, but it can be cleared by writing 1 to it.<br>**Note2:** This bit is only valid in LIN Slave mode (SLVEN (UART_LINCTL [0]) = 1) and enable LIN slave header detection function (SLVHDEN (UART_LINCTL [1])).<br>**Note3:** When enable ID parity check IDPEN (UART_LINCTL [9]), if hardware detect complete header (“break + sync + frame ID”), the SLVHDETF will be set whether the frame ID correct or not. |

## 6.11 I²C Serial Interface Controller (Master/Slave)

### 6.11.1 Overview

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

Data is transferred between a Master and a Slave synchronously to SCL on the SDA line on a byte-by-byte basis. Each data byte is 8-bit long. There is one SCL clock pulse for each data bit with the MSB being transmitted first. An acknowledge bit follows each transferred byte. Each bit is sampled during the high period of SCL; therefore, the SDA line may be changed only during the low period of SCL and must be held stable during the high period of SCL. A transition on the SDA line while SCL is high is interpreted as a command (START or STOP). Please refer to the following figure for more details about I²C Bus Timing.

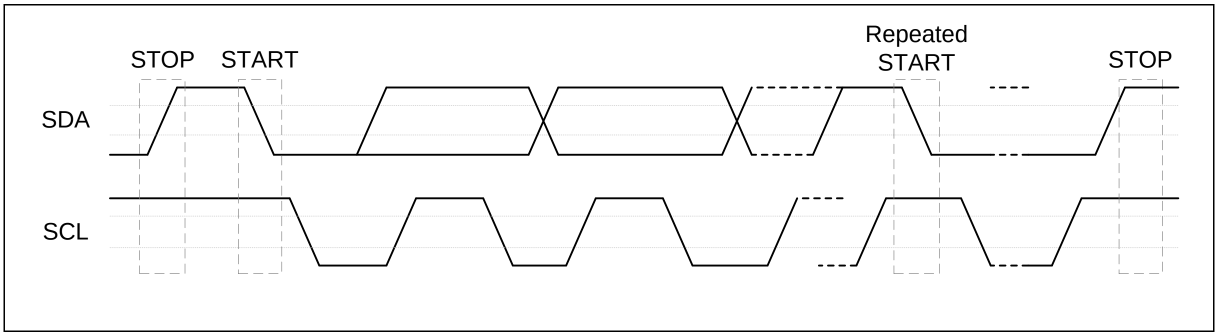


Figure 6.11-1 I²C Bus Timing

The device on-chip I²C logic provides the serial interface that meets the I²C bus standard mode specification. The I²C port handles byte transfers autonomously. To enable this port, the bit I2CEN (I2C_CTL[6]) should be set to '1'. The I²C H/W interfaces to the I²C bus via two pins: SDA and SCL. Pull up resistor is needed for I²C operation as these are open drain pins. When the I/O pins are used as I²C port, user must set the pins function to I²C in advance.

### 6.11.2 Features

The NUC505 series provides two channels of I²C. 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 and General Call Mode
- Bidirectional data transfer between masters and slaves
- Multi-master bus
- Arbitration between simultaneously transmitting masters without corruption of serial data on the bus
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus
- Serial clock synchronization can be used as a handshake mechanism to stretch and un-stretch serial transfer
- Built-in a 14-bit time-out counter will request the I²C interrupt if the I²C bus hangs up and time-out counter overflows.
- Programmable divider allowing for versatile rate control
- Supports 7-bit addressing mode
- Supports multiple address recognition (four slave addresses with mask option)
- Supports address match wake-up function

### 6.11.3 Basic Configuration

The $I^2C$ pins are configured in SYS_GPA_MFPH, SYS_GPB_MFPL, SYS_GPB_MFPH and SYS_GPD_MFPL Multi-function Registers.

The Multi-function Registers  SYS_GPA_MFPH[26:24], SYS_GPB_MFPL[2:0] and SYS_GPD_MFPL[2:0] for I2C0_SCL

The Multi-function Registers  SYS_GPA_MFPH[30:28], SYS_GPB_MFPL[6:4] and SYS_GPD_MFPL[6:4] for I2C0_SDA

The Multi-function Registers  SYS_GPA_MFPH[10:8], SYS_GPB_MFPH[10:8] and SYS_GPB_MFPL[26:24] for I2C1_SCL

The Multi-function Registers  SYS_GPA_MFPH[14:12], SYS_GPB_MFPH[14:12] and SYS_GPB_MFPL[30:28] for I2C1_SDA

### 6.11.4 Functional Description

#### *6.11.4.1 $I^2C$ Protocol*

Normally, a standard communication consists of four parts:

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

4) STOP signal generation

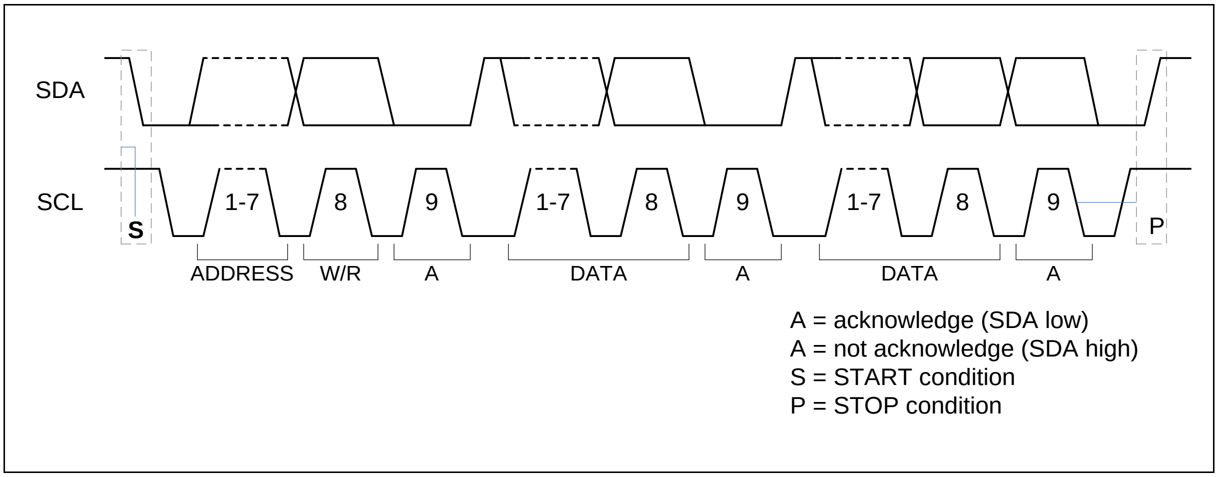


Figure 6.11-2 I²C Protocol

### 6.11.4.2 *Data transfer on the I²C Bus*

The following figure shows a master transmits data to slave. A master addresses a slave with a 7-bit address and 1-bit write index to denote master wants to transmit data to slave. The master keep transmitting data after slave returns acknowledge to master.

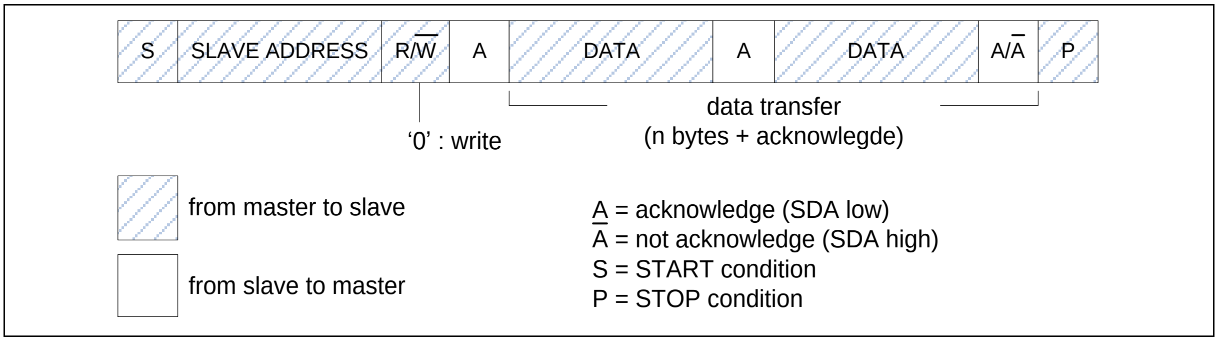


Figure 6.11-3 Master Transmits Data to Slave

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

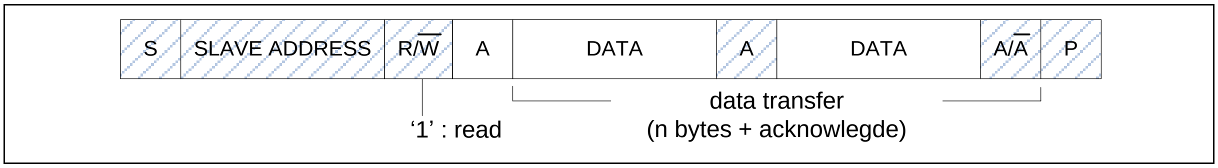


Figure 6.11-4 Master Reads Data from Slave

### 6.11.4.3 *START or Repeated START signal and STOP Signal*

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

A Repeated START (Sr) is no STOP signal between two START signals. The master uses this method to communicate with another slave or the same slave in a different transfer direction (e.g. from writing to a device to reading from a device) without releasing the bus.

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

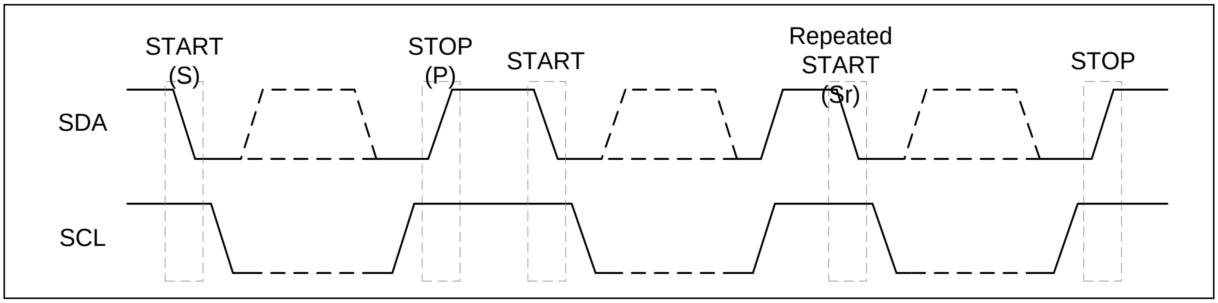


Figure 6.11-5 START and STOP Condition

### 6.11.4.4 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.

### 6.11.4.5 Data Transfer

Once successful slave addressing has been achieved, the data transfer can proceed on a byte-by-byte basis in the direction specified by the RW bit sent by the master. Each transferred byte is followed by an acknowledge bit on the 9th SCL clock cycle. If the slave signals a Not Acknowledge (NACK), the master can generate a STOP signal to abort the data transfer or generate a Repeated START signal and start a new transfer cycle.

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

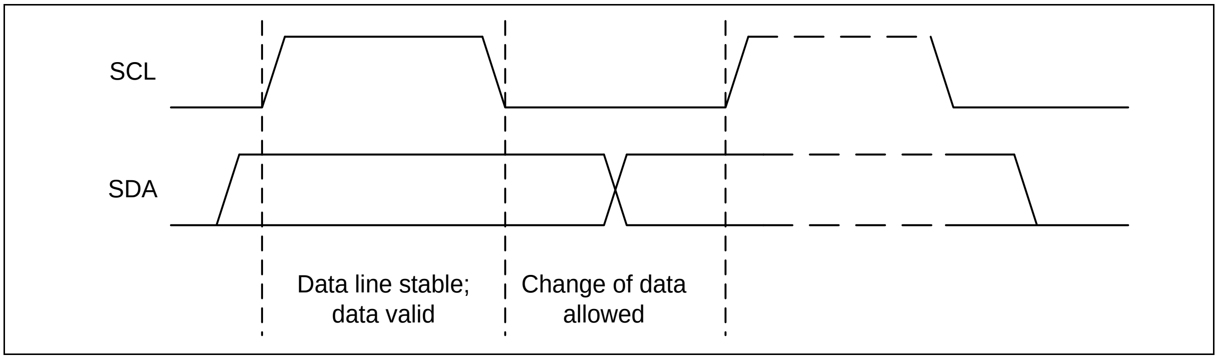


Figure 6.11-6 Bit Transfer on $I^2C$ Bus

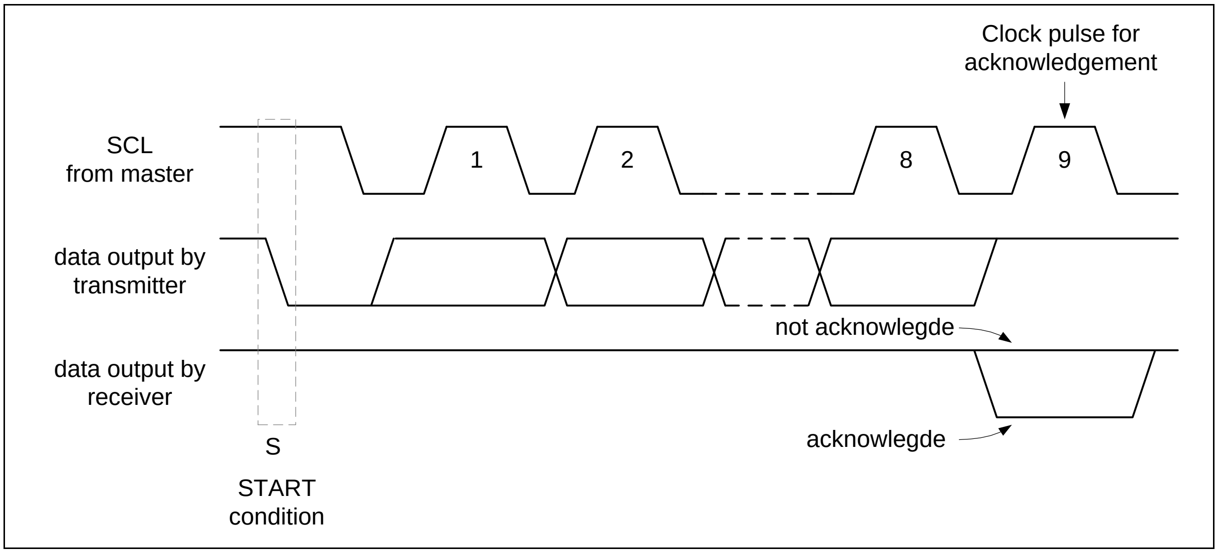


Figure 6.11-7 Acknowledge on $I^2C$ Bus

### 6.11.5 Operation Modes

The on-chip $I^2C$ ports support three operation modes, Master, Slave, and General Call Mode.

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 pulse will be transmitted out on the 9th clock, hence an interrupt is requested on both master and slave devices if the 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.

To control the $I^2C$ bus transfer in each mode, user needs to set I2C_CTL, I2C_DAT registers according to current status code of I2C_STATUS register. In other words, for each $I^2C$ bus action, user needs to check current status by I2C_STATUS register, and then set I2C_CTL, I2C_DAT registers to take bus action. Finally, check the response status by I2C_STATUS.

The bits, STA, STO and AA in I2C_CTL register are used to control the next state of the $I^2C$ hardware after SI flag of I2C_CTL [3] register is cleared. Upon completion of the new action, a new status code will be updated in I2C_STATUS register and the SI flag of I2C_CTL register will be set. If the $I^2C$ interrupt control bit EI (I2C_CTL [7]) is set, appropriate action or software branch of the new status code can be performed in the Interrupt service routine.

The following figure shows the current $I^2C$ status code is 0x08, and then set I2C_DATA=SLA+W and (STA,STO,SI,AA) = (0,0,1,x) to send the address to $I^2C$ bus. If a slave on the bus matches the address and response ACK, the I2C_STATUS will be updated by status code 0x18.

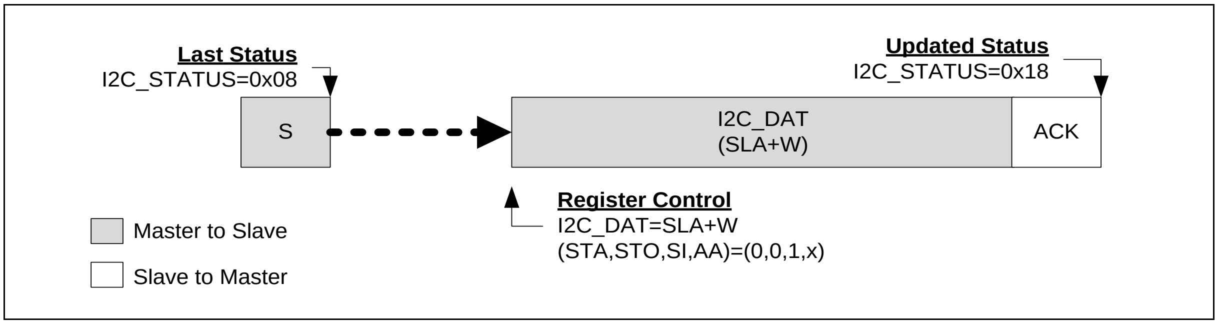


Figure 6.11-8 Control I$^2$C Bus according to Current I$^2$C Status

### *6.11.5.1 Master Mode*

In below figures, all possible protocols for I$^2$C master are shown. User needs to follow proper path of the flow to implement required I$^2$C protocol.

In other words, user can send a START signal to bus and I$^2$C will be in Master Transmitter mode (Figure 6.11-9) or Master receiver mode (Figure 6.11-10) after START signal has been sent successfully and new status code would be 0x08. Followed by START signal, user can send slave address, read/write bit, data and Repeat START, STOP to perform I$^2$C protocol.

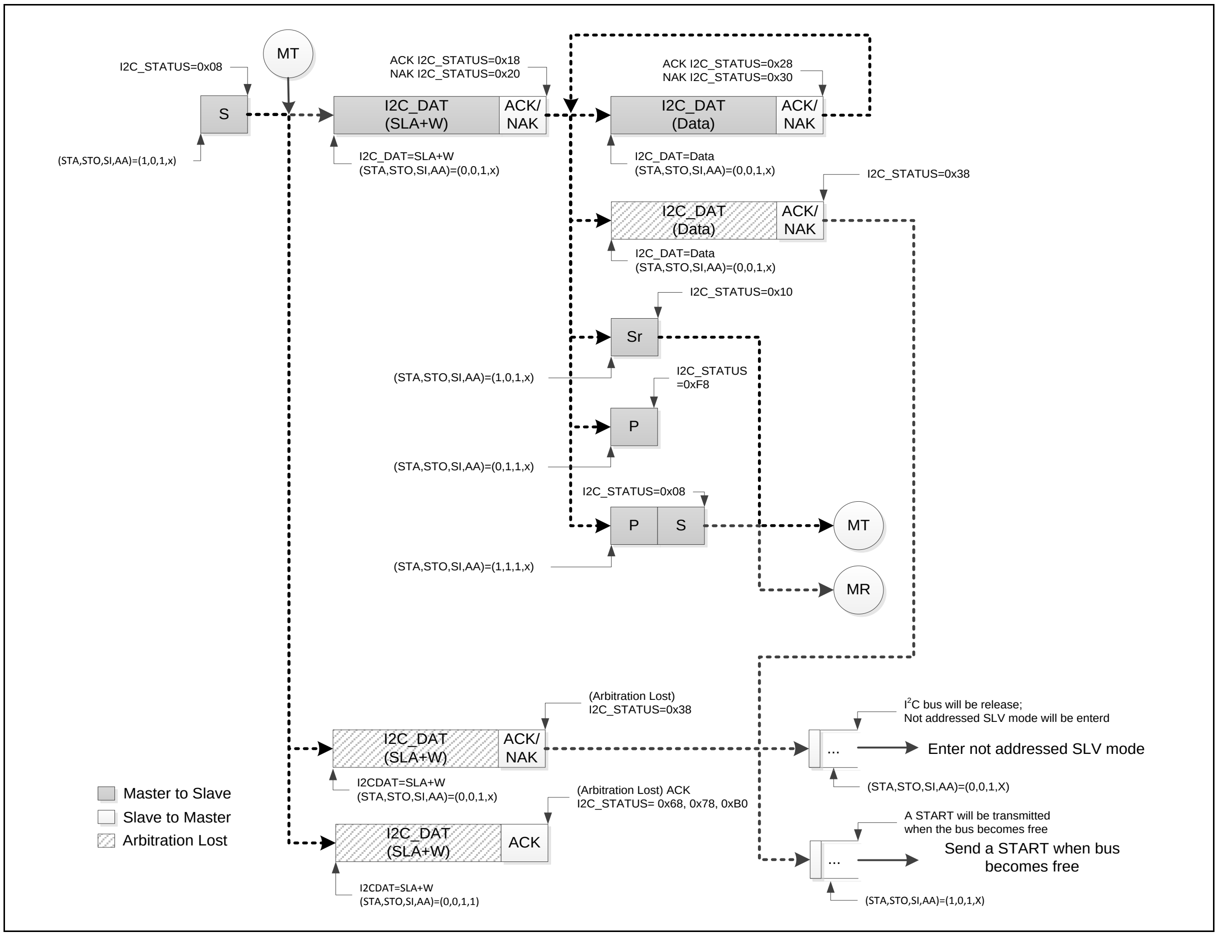


Figure 6.11-9 Master Transmitter Mode Control Flow

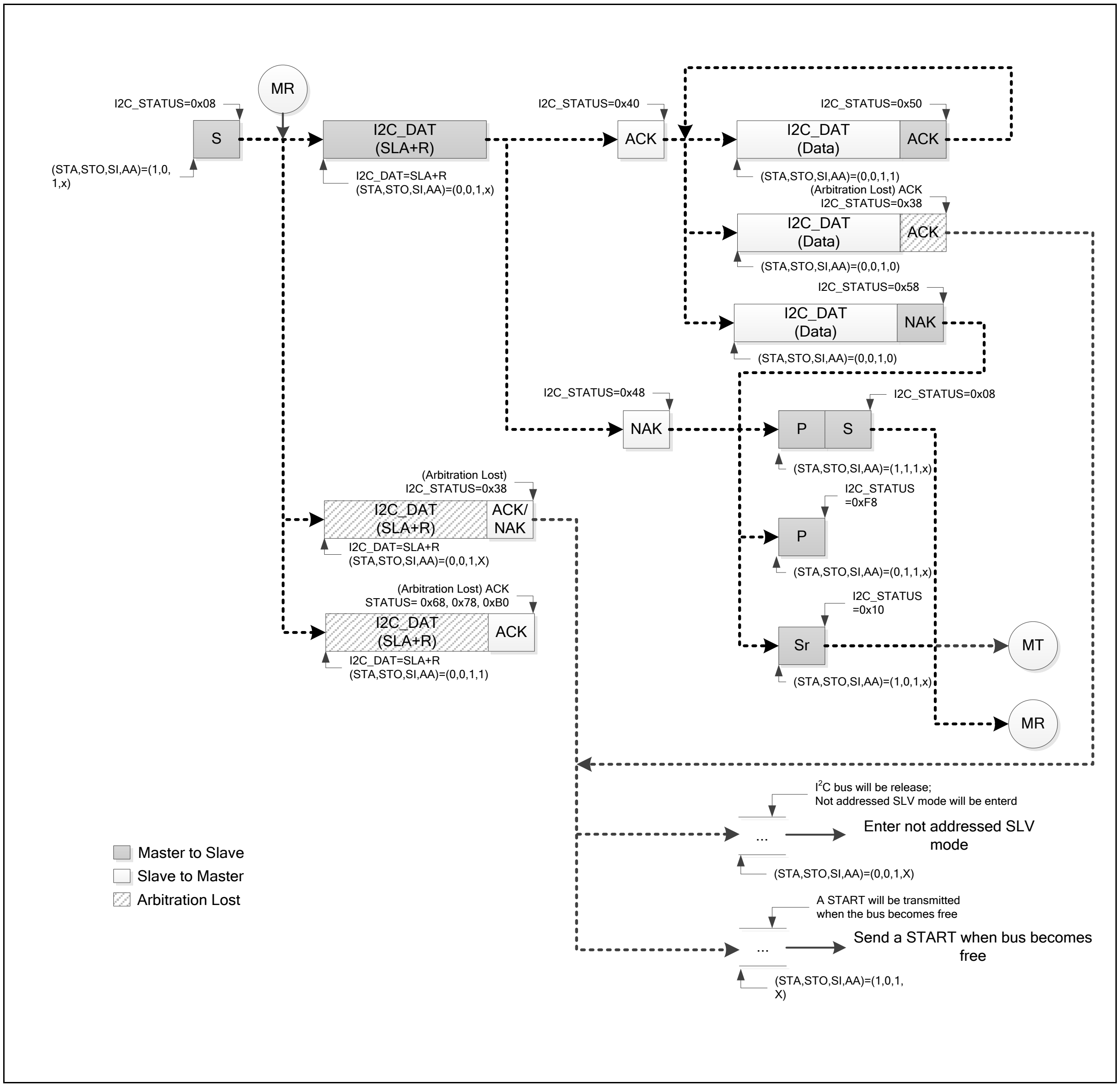

MR
I2C_STATUS=0x08
S
(STA,STO,SI,AA)=(1,0,1,x)
I2C_DAT (SLA+R)
I2C_DAT=SLA+R (STA,STO,SI,AA)=(0,0,1,x)
I2C_STATUS=0x40
ACK
I2C_STATUS=0x50
I2C_DAT (Data)
ACK
(STA,STO,SI,AA)=(0,0,1,1)
(Arbitration Lost) ACK
I2C_STATUS=0x38
I2C_DAT (Data)
ACK
(STA,STO,SI,AA)=(0,0,1,0)
I2C_STATUS=0x58
I2C_DAT (Data)
NAK
(STA,STO,SI,AA)=(0,0,1,0)
I2C_STATUS=0x48
NAK
I2C_STATUS=0x08
P
S
(STA,STO,SI,AA)=(1,1,1,x)
I2C_STATUS =0xF8
P
(STA,STO,SI,AA)=(0,1,1,x)
I2C_STATUS =0x10
Sr
MT
(STA,STO,SI,AA)=(1,0,1,x)
MR
(Arbitration Lost)
I2C_STATUS=0x38
I2C_DAT (SLA+R)
ACK/ NAK
I2C_DAT=SLA+R (STA,STO,SI,AA)=(0,0,1,X)
(Arbitration Lost) ACK
STATUS= 0x68, 0x78, 0xB0
I2C_DAT (SLA+R)
ACK
I2C_DAT=SLA+R (STA,STO,SI,AA)=(0,0,1,1)
I²C bus will be release;
Not addressed SLV mode will be enterd
...
Enter not addressed SLV mode
(STA,STO,SI,AA)=(0,0,1,X)
A START will be transmitted when the bus becomes free
...
Send a START when bus becomes free
(STA,STO,SI,AA)=(1,0,1, X)
▩ Master to Slave
□ Slave to Master
▨ Arbitration Lost

Figure 6.11-10 Master Receiver Mode Control Flow

If the I$^2$C is in Master mode and gets arbitration lost, the status code will be 0x38. In status 0x38, user may set (STA, STO, SI, AA) = (1, 0, 1, X) to send START to re-start Master operation when bus become free. Otherwise, user may set (STA, STO, SI, AA) = (0, 0, 1, X) to release I$^2$C bus and enter not addressed Slave mode.

### *6.11.5.2 Slave Mode*

When reset default, I$^2$C is not addressed and will not recognize the address on I$^2$C bus. User can set slave address by I2CADDRx and set (STA, STO, SI, AA) = (0, 0, 1, 1) to let I$^2$C recognize the address sent by master. The following figure shows all the possible flow for I$^2$C in Slave mode.

If bus arbitration is lost in Master mode, I$^2$C port switches to Slave mode immediately and can detect its own slave address in the same serial transfer. If the detected address is SLA+W

(Master want to write data to Slave) after arbitration lost, the status code is 0x68. If the detected address is SLA+R (Master want to read data from Slave) after arbitration lost, the status code is 0xB0.

During I²C communication, the SCL clock will be released when writing '1' to clear SI (I2C_CTL[3]) flag in Slave mode.

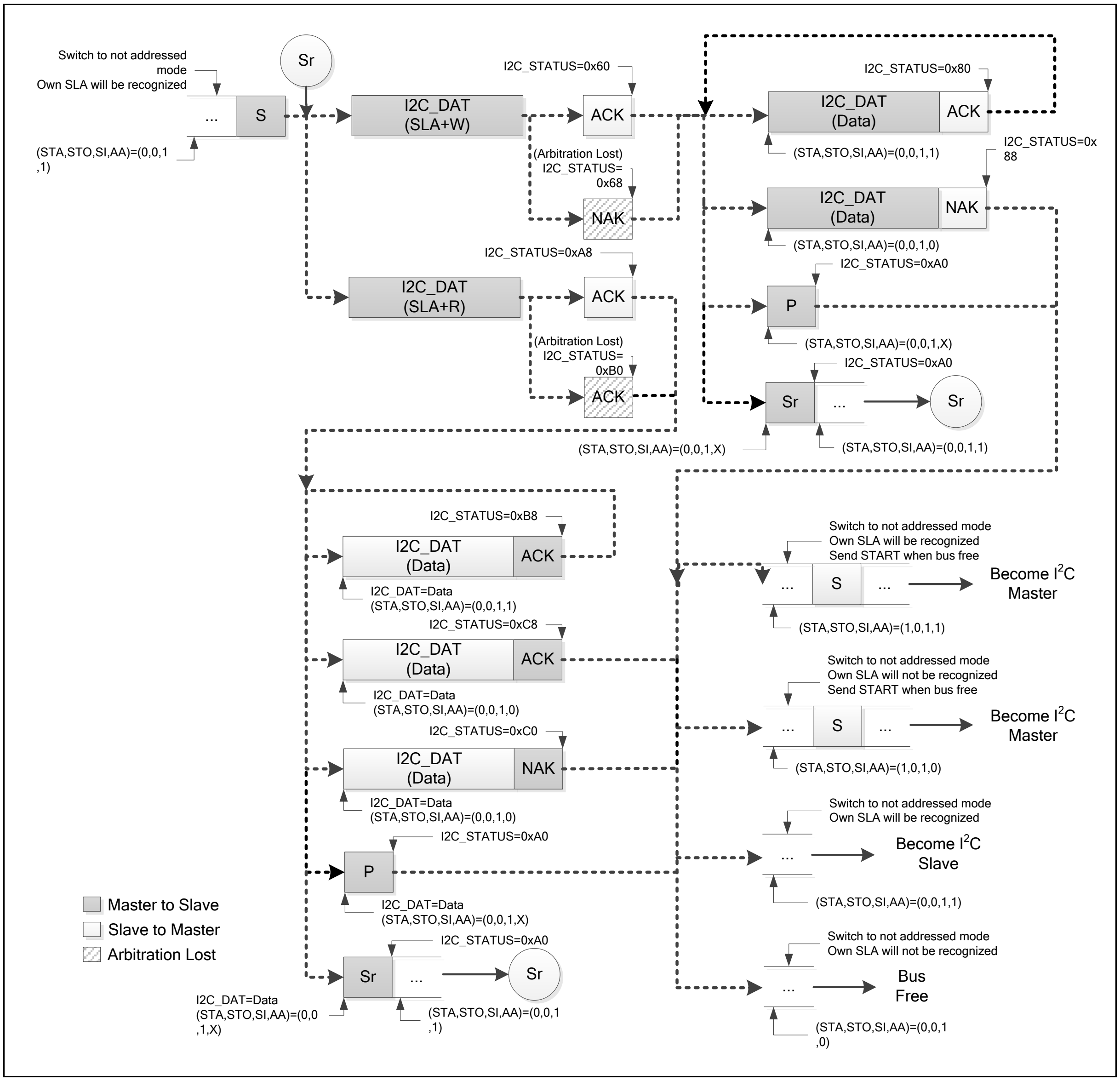

Switch to not addressed mode
Own SLA will be recognized
Sr
...
S
(STA,STO,SI,AA)=(0,0,1,1)
I2C_DAT (SLA+W)
I2C_STATUS=0x60
ACK
(Arbitration Lost) I2C_STATUS= 0x68
NAK
I2C_STATUS=0x80
I2C_DAT (Data)
ACK
(STA,STO,SI,AA)=(0,0,1,1)
I2C_STATUS=0x88
I2C_DAT (Data)
NAK
(STA,STO,SI,AA)=(0,0,1,0)
I2C_STATUS=0xA0
P
(STA,STO,SI,AA)=(0,0,1,X)
I2C_STATUS=0xA0
Sr
...
Sr
(STA,STO,SI,AA)=(0,0,1,X)
(STA,STO,SI,AA)=(0,0,1,1)
I2C_STATUS=0xA8
I2C_DAT (SLA+R)
ACK
(Arbitration Lost) I2C_STATUS= 0xB0
ACK
I2C_STATUS=0xB8
I2C_DAT (Data)
ACK
I2C_DAT=Data (STA,STO,SI,AA)=(0,0,1,1)
I2C_STATUS=0xC8
I2C_DAT (Data)
ACK
I2C_DAT=Data (STA,STO,SI,AA)=(0,0,1,0)
I2C_STATUS=0xC0
I2C_DAT (Data)
NAK
I2C_DAT=Data (STA,STO,SI,AA)=(0,0,1,0)
I2C_STATUS=0xA0
P
I2C_DAT=Data (STA,STO,SI,AA)=(0,0,1,X)
I2C_STATUS=0xA0
Sr
...
Sr
I2C_DAT=Data (STA,STO,SI,AA)=(0,0,1,X)
(STA,STO,SI,AA)=(0,0,1,1)
Switch to not addressed mode
Own SLA will be recognized
Send START when bus free
...
S
...
Become I²C Master
(STA,STO,SI,AA)=(1,0,1,1)
Switch to not addressed mode
Own SLA will not be recognized
Send START when bus free
...
S
...
Become I²C Master
(STA,STO,SI,AA)=(1,0,1,0)
Switch to not addressed mode
Own SLA will be recognized
...
Become I²C Slave
(STA,STO,SI,AA)=(0,0,1,1)
Switch to not addressed mode
Own SLA will not be recognized
...
Bus Free
(STA,STO,SI,AA)=(0,0,1,0)
▨ Master to Slave
□ Slave to Master
▨ Arbitration Lost

Figure 6.11-11 Slave Mode Control Flow

If I²C is still receiving data in addressed Slave mode but got a STOP or Repeat START, the status code will be 0xA0. User could follow the action for status code 0x88 as shown in the above figure when getting 0xA0 status.

If I²C is still transmitting data in addressed Slave mode but got a STOP or Repeat START, the status code will be 0xA0. User could follow the action for status code 0xC8 as shown in the above figure when getting 0xA0 status.

After slave gets status of 0x88, 0xC8, 0xC0 and 0xA0, slave can switch to not address mode and own SLA will not be recognized. If entering this status, slave will not receive any I$^2$C signal or address from master. At this status, I$^2$C should be reset to leave this status.

### *6.11.5.3 General Call (GC) Mode*

If the GC bit (I2C_ADDRn [0]) is set, the I$^2$C port hardware will respond to General Call address (00H). User can clear GC bit to disable general call function. When the GC bit is set and the I$^2$C is in Slave mode, it can receive the general call address by 0x00 after master send general call address to I$^2$C bus, then it will follow status of GC mode.

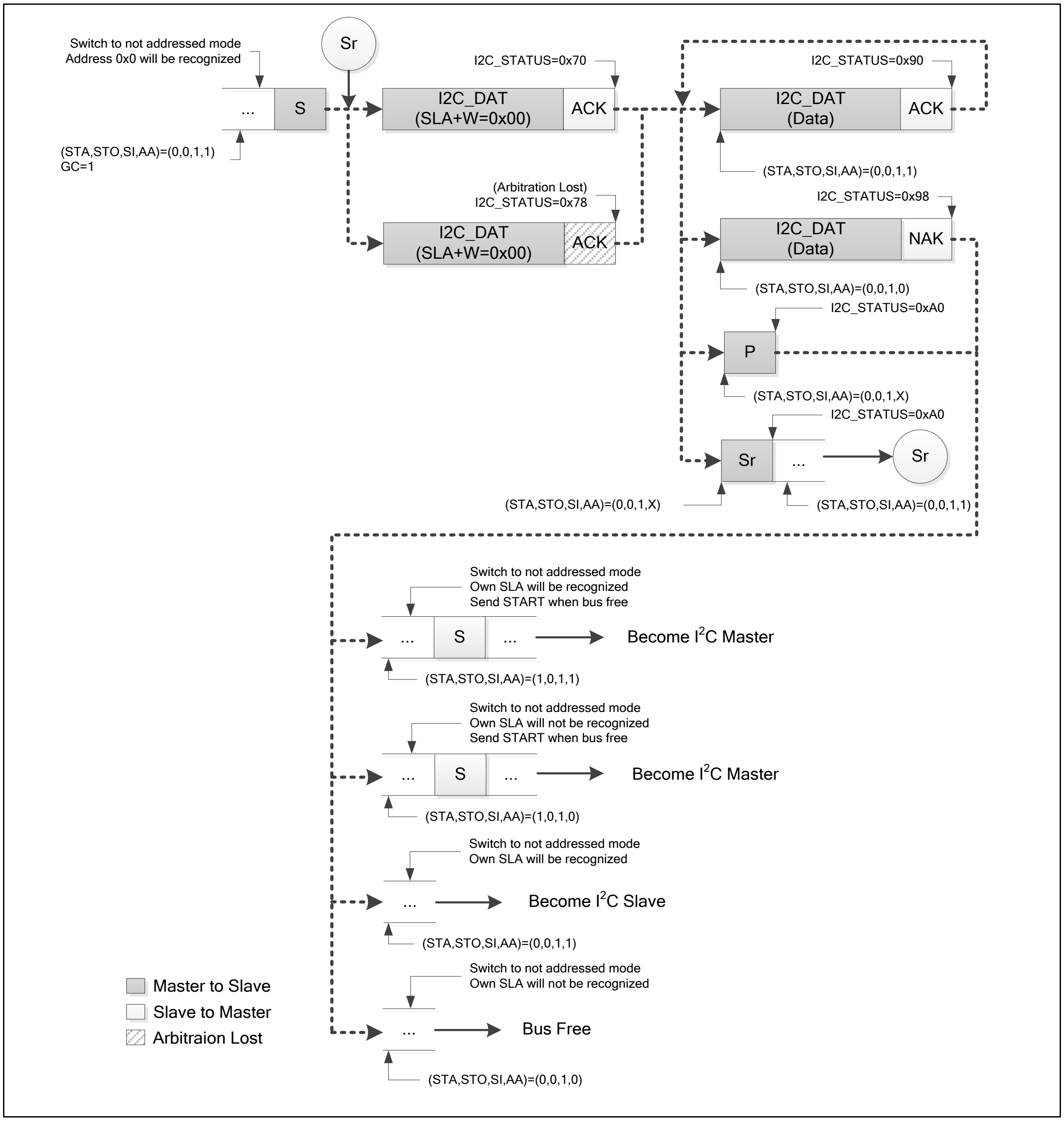

Switch to not addressed mode
Address 0x0 will be recognized
Sr
...
S
(STA,STO,SI,AA)=(0,0,1,1)
GC=1
I2C_STATUS=0x70
I2C_DAT
(SLA+W=0x00)
ACK
(Arbitration Lost)
I2C_STATUS=0x78
I2C_DAT
(SLA+W=0x00)
ACK
I2C_STATUS=0x90
I2C_DAT
(Data)
ACK
(STA,STO,SI,AA)=(0,0,1,1)
I2C_STATUS=0x98
I2C_DAT
(Data)
NAK
(STA,STO,SI,AA)=(0,0,1,0)
I2C_STATUS=0xA0
P
(STA,STO,SI,AA)=(0,0,1,X)
I2C_STATUS=0xA0
Sr
...
Sr
(STA,STO,SI,AA)=(0,0,1,X)
(STA,STO,SI,AA)=(0,0,1,1)
Switch to not addressed mode
Own SLA will be recognized
Send START when bus free
...
S
...
Become I2C Master
(STA,STO,SI,AA)=(1,0,1,1)
Switch to not addressed mode
Own SLA will not be recognized
Send START when bus free
...
S
...
Become I2C Master
(STA,STO,SI,AA)=(1,0,1,0)
Switch to not addressed mode
Own SLA will be recognized
...
Become I2C Slave
(STA,STO,SI,AA)=(0,0,1,1)
Switch to not addressed mode
Own SLA will not be recognized
...
Bus Free
(STA,STO,SI,AA)=(0,0,1,0)
Master to Slave
Slave to Master
Arbitraion Lost

Figure 6.11-12 GC Mode

If I$^2$C is still receiving data in GC mode but got a STOP or Repeat START, the status code will be

0xA0. User could follow the action for status code 0x98 in above figure when getting 0xA0 status.

After slave gets status of 0x98 and 0xA0, slave can switch to not address mode and own SLA will not be recognized. If entering this status, slave will not receive any I$^2$C signal or address from master. At this time, I$^2$C controller should be reset to leave this status.

#### 6.11.5.4 Multi-Master

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

- When I2C_STATUS = 0x38, an “Arbitration Lost” is received. Arbitration lost event maybe occur during the send START bit, data bits or STOP bit. User could set (STA, STO, SI, AA) = (1, 0, 1, X) to send START again when bus free, or set (STA, STO, SI, AA) = (0, 0, 1, X) to send STOP to back to not addressed Slave mode.
- When I2C_STATUS = 0x00, a “Bus Error” is received. To recover I$^2$C bus from a bus error, STO should be set and SI should be cleared, and then STO is cleared to release bus.
  - Set (STA, STO, SI, AA) = (0, 1, 1, X) to stop current transfer
  - Set (STA, STO, SI, AA) = (0, 0, 1, X) to release bus

#### 6.11.5.5 Example for Random Read on EEPROM

The following steps are used to configure the I$^2$C related registers when using I$^2$C to read data from EEPROM.

1. Set the multi-function pin in the “SYS_GPB_MFPL” register as SCL (PB.0) and SDA (PB.1) pins.
2. Enable I$^2$C APB clock, I2C0CKEN=1 in the “CLK_APBCLK[5]” register.
3. Set I2C0RST=1 to reset I$^2$C controller then set I$^2$C controller to normal operation, I2C0RST=0 in the “SYS_IPRST1[8]” register.
4. Set I2CEN (I2C_CTL[6])=1 to enable I$^2$C controller in the “I2C_CTL” register.
5. Give I$^2$C clock a divided register value for I$^2$C clock rate in the “I2C_CLKDIV”.
6. Set the “NVIC_ISER” register to enable the I$^2$C IRQ.
7. Set INTEN (I2C_CTL[7])=1 to enable I$^2$C Interrupt in the “I2C_CTL” register.
8. Set I$^2$C address registers which are “I2C_ADDR0~I2C_ADDR3”.

Random read operation is one of the methods of access EEPROM. The method allows the master to access any address of EEPROM space. The following figure shows the EEPROM random read operation.

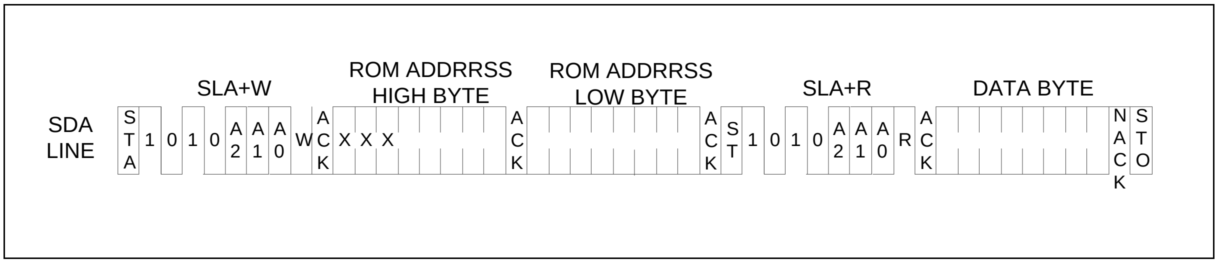


Figure 6.11-13 EEPROM Random Read

The following figure shows how to use I$^2$C controller to implement the protocol of EEPROM random read.

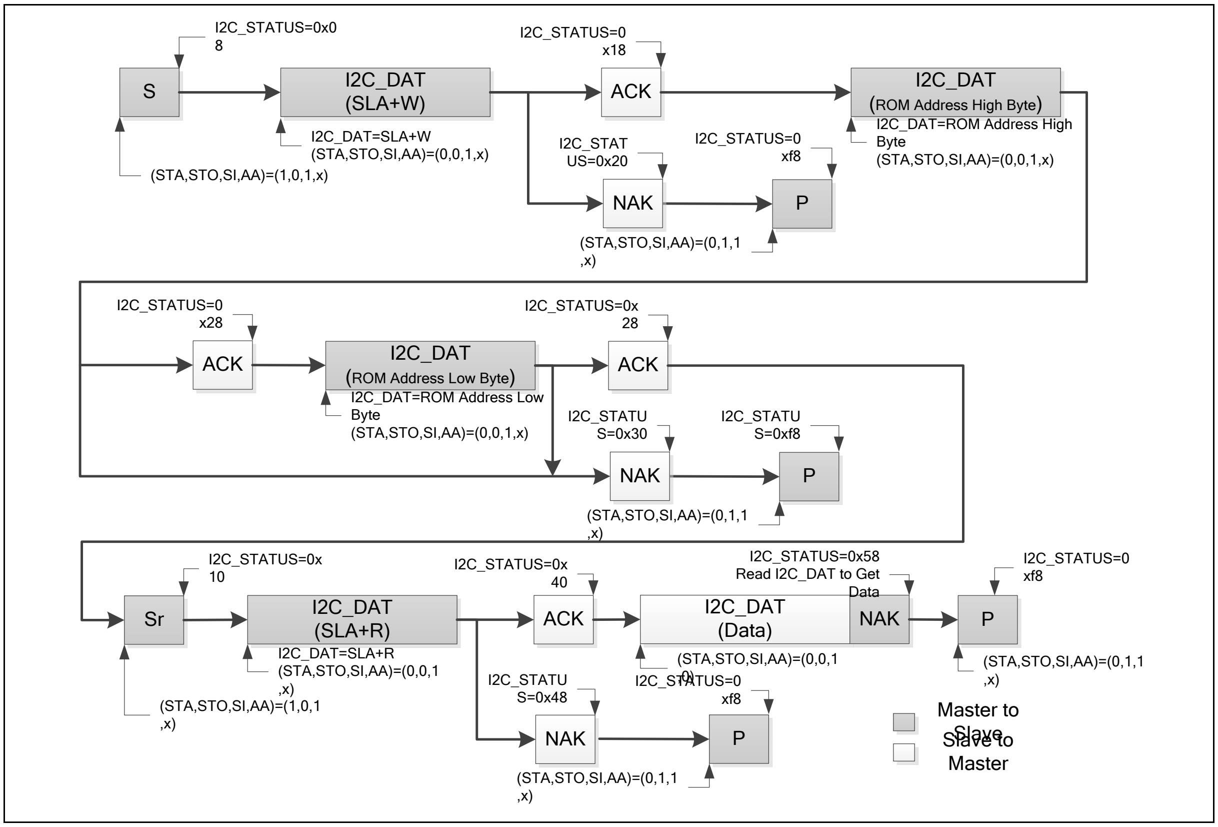

I2C_STATUS=0x08
S
I2C_DAT (SLA+W)
I2C_DAT=SLA+W (STA,STO,SI,AA)=(0,0,1,x)
(STA,STO,SI,AA)=(1,0,1,x)
I2C_STATUS=0x18
ACK
I2C_STATUS=0x20
NAK
I2C_STATUS=0xf8
P
(STA,STO,SI,AA)=(0,1,1,x)
I2C_DAT (ROM Address High Byte)
I2C_DAT=ROM Address High Byte (STA,STO,SI,AA)=(0,0,1,x)
I2C_STATUS=0x28
ACK
I2C_DAT (ROM Address Low Byte)
I2C_DAT=ROM Address Low Byte (STA,STO,SI,AA)=(0,0,1,x)
I2C_STATUS=0x28
ACK
I2C_STATUS=0x30
NAK
I2C_STATUS=0xf8
P
(STA,STO,SI,AA)=(0,1,1,x)
I2C_STATUS=0x10
Sr
I2C_DAT (SLA+R)
I2C_DAT=SLA+R (STA,STO,SI,AA)=(0,0,1,x)
(STA,STO,SI,AA)=(1,0,1,x)
I2C_STATUS=0x40
ACK
I2C_STATUS=0x48
NAK
I2C_STATUS=0xf8
P
(STA,STO,SI,AA)=(0,1,1,x)
I2C_STATUS=0x58 Read I2C_DAT to Get Data
I2C_DAT (Data)
NAK
(STA,STO,SI,AA)=(0,0,1,0)
I2C_STATUS=0xf8
P
(STA,STO,SI,AA)=(0,1,1,x)
Master to Slave
Slave to Master

Figure 6.11-14 Protocol of EEPROM Random Read

The I$^2$C controller sends START to bus to be a master. Then it sends a SLA+W (Slave address + Write bit) to EERPOM followed by two bytes data address to set the EEPROM address to read. Finally, a Repeat START followed by SLA+R is sent to read the data from EEPROM.

### 6.11.6 Protocol Registers

The CPU interfaces to the I$^2$C port through the following thirteen special function registers: I2C_CTL (control register), I2C_STATUS (status register), I2C_DAT (data register), I2C_ADDRn (address registers, n=0~3), I2C_ADDRMSKn (address mask registers, n=0~3), I2C_CLKDIV (clock rate register) and I2C_TOCTL (Time-out counter register). All bit 31~ bit 8 of these I$^2$C special function registers are reserved. These bits do not have any functions and are all zero if read back.

When I$^2$C port is enabled by setting I2CEN (I2C_CTL [6]) to high, the internal states will be controlled by I2C_CTL and I$^2$C logic hardware. Once a new status code is generated and stored in I2C_STATUS, the I$^2$C Interrupt Flag bit SI (I2C_CTL [3]) will be set automatically. If the Enable Interrupt bit INTEN (I2C_CTL [7]) is set high at this time, the I$^2$C interrupt will be generated. The bit field I2C_STATUS[7:3] stores the internal state code, the lowest 3 bits of I2C_STATUS are always zero and the content keeps stable until SI is cleared by software.

#### *6.11.6.1 Address Registers (I2C_ADDR)*

The I$^2$C port is equipped with four slave address registers I2C_ADDRn (n=0~3). The contents of the register are irrelevant when I$^2$C is in Master mode. In Slave mode, the bit field I2C_ADDRn[7:1] must be loaded with the chip's own slave address. The I$^2$C hardware will react if the contents of I2C_ADDRn are matched with the received slave address.

The I$^2$C ports support the "General Call" function. If the GC bit (I2C_ADDRn [0]) is set the I$^2$C port hardware will respond to General Call address (00H). Clear GC bit to disable general call function.

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

I$^2$C bus controllers support multiple address recognition with four address mask registers I2C_ADDRMSKn (n=0~3). When the bit in the address mask register is set to one, it means the received corresponding address bit is don't-care. If the bit is set to zero, that means the received corresponding register bit should be exact the same as address register.

#### *6.11.6.2 Data Register (I2C_DAT)*

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

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

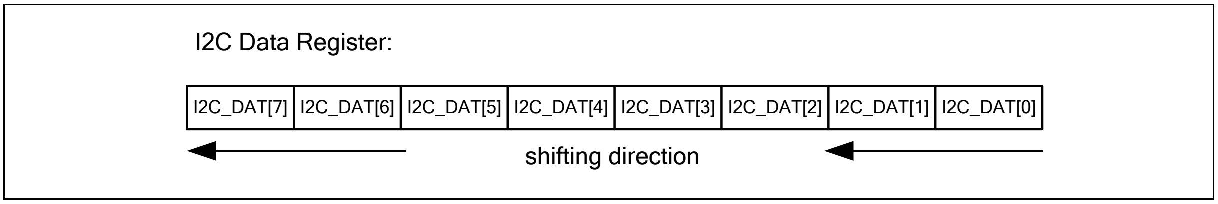


Figure 6.11-15 I$^2$C Data Shifting Direction

### *6.11.6.3 Control Register (I2C_CTL)*

The CPU can read from and write to this 8-bit field of I2C_CTL [7:0] directly. Two bits are affected by hardware: the SI bit is set when the I$^2$C hardware requests a serial interrupt, and the STO bit is cleared when a STOP condition is present on the bus. The STO bit is also cleared when I2CEN = 0.

INTEN Enable Interrupt.

I2CEN Set to enable I$^2$C serial function controller. When I2CEN=1 the I$^2$C serial function enables. The Multi-function pin function of SDA and SCL must be set to I$^2$C function.

STA I$^2$C START Control Bit. Setting STA to logic 1 to enter Master mode, the I$^2$C hardware sends a START or repeat START condition to bus when the bus is free.

STO I$^2$C STOP Control Bit. In Master mode, setting STO to transmit a STOP condition to bus then I$^2$C hardware will check the bus condition if a STOP condition is detected this flag will be cleared by hardware automatically. In Slave mode, setting STO resets I$^2$C hardware to the defined “not addressed” Slave mode. This means it is NO LONGER in the slave receiver mode to receive data from the master transmit device.

SI I$^2$C Interrupt Flag. When a new I$^2$C state is present in the I2C_STATUS register, the SI flag is set by hardware, and if bit INTEN (I2C_CTL [7]) is set, the I$^2$C interrupt is requested. SI must be cleared by software. Clear SI by writing 1 to this bit. All states are listed in I2C_STATUS Register section.

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

### *6.11.6.4 Status Register (I2C_STATUS)*

I2C_STATUS [7:0] is an 8-bit read only register. The three least significant bits are always 0. The bit field I2C_STATUS [7:3] contain the status code. There are 26 possible status codes, All states are listed in Table 6.11-1. When I2C_STATUS [7:0] contains F8H, no serial interrupt is requested. All other I2C_STATUS [7:3] values correspond to defined I2C states. When each of these states is entered, a status interrupt is requested (SI = 1). A valid status code is present in I2C_STATUS[7:3] one cycle after SI is set by hardware and is still present one cycle after SI has been reset by software.

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

| Master Mode | | Slave Mode | |
|---|---|---|---|
| **STATUS** | **Description** | **STATUS** | **Description** |
| 0x08 | Start | 0xA0 | Slave Transmit Repeat Start or Stop |
| 0x10 | Master Repeat Start | 0xA8 | Slave Transmit Address ACK |
| 0x18 | Master Transmit Address ACK | 0xB0 | Slave Transmit Arbitration Lost |
| 0x20 | Master Transmit Address NACK | 0xB8 | Slave Transmit Data ACK |
| 0x28 | Master Transmit Data ACK | 0xC0 | Slave Transmit Data NACK |
| 0x30 | Master Transmit Data NACK | 0xC8 | Slave Transmit Last Data ACK |
| 0x38 | Master Arbitration Lost | 0x60 | Slave Receive Address ACK |
| 0x40 | Master Receive Address ACK | 0x68 | Slave Receive Arbitration Lost |
| 0x48 | Master Receive Address NACK | 0x80 | Slave Receive Data ACK |
| 0x50 | Master Receive Data ACK | 0x88 | Slave Receive Data NACK |
| 0x58 | Master Receive Data NACK | 0x70 | GC mode Address ACK |
| 0x00 | Bus error | 0x78 | GC mode Arbitration Lost |
| - | - | 0x90 | GC mode Data ACK |
| - | - | 0x98 | GC mode Data NACK |
| 0xF8 | Bus Released<br>Status “0xF8” exists in both Master/Slave modes, and it will not raise interrupt. | | |

Table 6.11-1 I$^2$C Status Code Description Table

### 6.11.6.5 *I2C Clock Baud Rate Bits (I2C_CLKDIV)*

The data baud rate of I$^2$C is determines by I2C_CLKDIV [7:0] register when I$^2$C is in Master mode. It is not important when I$^2$C is in Slave mode. In Slave mode, I$^2$C will automatically synchronize with any clock frequency from master I$^2$C device.

The data baud rate of I$^2$C setting is Data Baud Rate of I$^2$C = (system clock) / (4x (I2C_CLKDIV [7:0] +1)). If system clock = 16 MHz, the I2C_CLKDIV [7:0] = 40 (28H), so data baud rate of I$^2$C = 16 MHz/ (4x (40 +1)) = 97.5 Kbits/sec.

### 6.11.6.6 *I$^2$C Time-out Counter Register (I2C_TOCTL)*

There is a 14-bit time-out counter which can be used to deal with the I$^2$C bus hang-up. If the time-out counter is enabled, the counter starts up counting until it overflows (TOIF (I2C_TOCTL[0])=1) and generates I$^2$C interrupt to CPU or stops counting by clearing TOCEN (I2C_TOCTL[2]) to 0. When the time-out counter is enabled, setting flag SI (I2C_CTL[3]) to high will reset counter and re-start up counting after SI is cleared. If I$^2$C bus hangs up, it causes the I2C_STATUS and flag SI are not updated for a period, the 14-bit time-out counter may overflow and acknowledge CPU the I$^2$C interrupt. Refer to the following figure for the 14-bit time-out counter. User may write 1 to clear TOIF to 0.

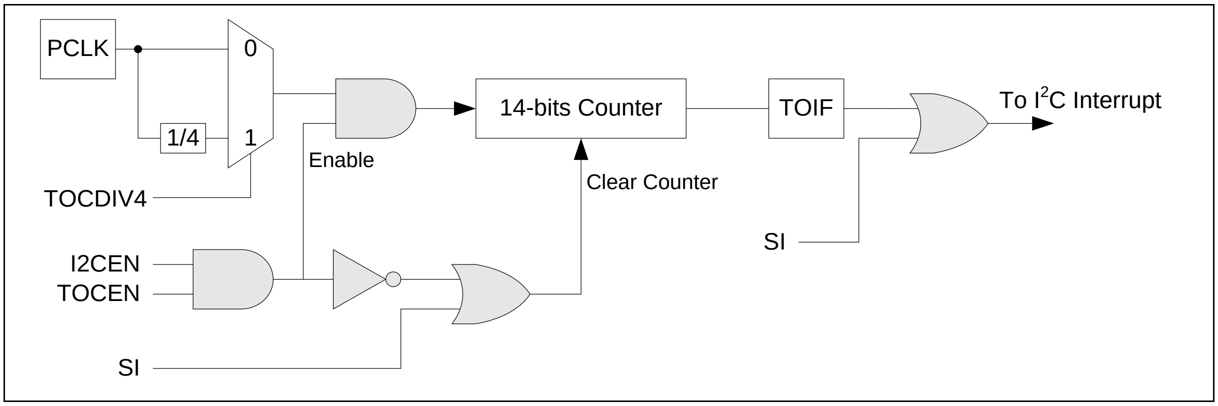


Figure 6.11-16 I$^2$C Time-out Count Block Diagram

### *6.11.6.7 The I$^2$C wake-up control Register (I2C_WKCTL)*

When entering Power-down mode, other I$^2$C master can wake up the chip by addressing the I$^2$C device, and user must configure the related settings before entering Power-down mode.

WKUPEN enables I$^2$C wake-up function

### *6.11.6.8 The I$^2$C wake-up status Register (I2C_WKSTS)*

When system is woken up by other I$^2$C master device, WKIF is set to indicate this event

WKIF Wake-up interrupt flag

### 6.11.7 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2C Base Address:**<br>**I2Cx_BA = 0x400E_4000 + x * 0x1000**<br>**x=0,1** | | | | |
| **I2C_CTL** | I2Cx_BA+0x00 | R/W | I$^2$C Control Register | 0x0000_0000 |
| **I2C_ADDR0** | I2Cx_BA+0x04 | R/W | I$^2$C Slave Address Register0 | 0x0000_0000 |
| **I2C_DAT** | I2Cx_BA+0x08 | R/W | I$^2$C Data Register | 0x0000_0000 |
| **I2C_STATUS** | I2Cx_BA+0x0C | R | I$^2$C Status Register | 0x0000_00F8 |
| **I2C_CLKDIV** | I2Cx_BA+0x10 | R/W | I$^2$C Clock Divided Register | 0x0000_0000 |
| **I2C_TOCTL** | I2Cx_BA+0x14 | R/W | I$^2$C Time-out Control Register | 0x0000_0000 |
| **I2C_ADDR1** | I2Cx_BA+0x18 | R/W | I$^2$C Slave Address Register1 | 0x0000_0000 |
| **I2C_ADDR2** | I2Cx_BA+0x1C | R/W | I$^2$C Slave Address Register2 | 0x0000_0000 |
| **I2C_ADDR3** | I2Cx_BA+0x20 | R/W | I$^2$C Slave Address Register3 | 0x0000_0000 |
| **I2C_ADDRMSK0** | I2Cx_BA+0x24 | R/W | I$^2$C Slave Address Mask Register0 | 0x0000_0000 |
| **I2C_ADDRMSK1** | I2Cx_BA+0x28 | R/W | I$^2$C Slave Address Mask Register1 | 0x0000_0000 |
| **I2C_ADDRMSK2** | I2Cx_BA+0x2C | R/W | I$^2$C Slave Address Mask Register2 | 0x0000_0000 |
| **I2C_ADDRMSK3** | I2Cx_BA+0x30 | R/W | I$^2$C Slave Address Mask Register3 | 0x0000_0000 |
| **I2C_WKCTL** | I2Cx_BA+0x3C | R/W | I$^2$C Wake-up Control Register | 0x0000_0000 |
| **I2C_WKSTS** | I2Cx_BA+0x40 | R/W | I$^2$C Wake-up Status Register | 0x0000_0000 |

### 6.11.8 Register Description

#### I²C Control Register (I2C_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| I2C_CTL | I2Cx_BA+0x00 | R/W | I²C Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| Reserved | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| INTEN | I2CEN | STA | STO | SI | AA | Reserved | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7] | **INTEN** | **I²C Interrupt Enable Control**<br>0 = I²C interrupt Disabled.<br>1 = I²C interrupt Enabled. |
| [6] | **I2CEN** | **I²C Controller Enable Control**<br>0 = Disabled.<br>1 = Enabled.<br>Set to enable I²C serial function controller. When I2CEN=1 the I²C serial function enables. The multi-function pin function of SDA and SCL must set to I²C function first. |
| [5] | **STA** | **I²C START Control**<br>Setting STA to logic 1 to enter Master mode, the I²C hardware sends a START or repeat START condition to bus when the bus is free. |
| [4] | **STO** | **I²C STOP Control**<br>In Master mode, setting STO to transmit a STOP condition to bus then I²C hardware will check the bus condition if a STOP condition is detected this bit will be cleared by hardware automatically. In Slave mode, setting STO resets I²C hardware to the defined “not addressed” Slave mode. This means it is NO LONGER in the slave receiver mode to receive data from the master transmit device. |
| [3] | **SI** | **I²C Interrupt Flag**<br>When a new I²C state is present in the I2C_STATUS register, the SI flag is set by hardware, and if bit INTEN (I2C_CTL [7]) is set, the I²C interrupt is requested. SI must be cleared by software. Clear SI by writing 1 to this bit. |
| [2] | **AA** | **Assert Acknowledge Control**<br>When AA =1 prior to address or data received, an acknowledged (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line when 1.). A slave is acknowledging the address sent from master, 2.) The receiver devices are acknowledging the data sent by transmitter. When AA=0 prior to address or data received, a Not |

| | | |
|---|---|---|
| | | acknowledged (high level to SDA) will be returned during the acknowledge clock pulse on the SCL line. |
| [1:0] | **Reserved** | Reserved. |

### I$^2$C Data Register (I2C_DAT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2C_DAT** | I2Cx_BA+0x08 | R/W | I$^2$C Data Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **DAT** | **I$^2$C Data Bits**<br>Bit [7:0] is located with the 8-bit transferred data of I$^2$C serial port. |

### $I^2C$ Status Register (I2C_STATUS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2C_STATUS** | I2Cx_BA+0x0C | R | $I^2C$ Status Register | 0x0000_00F8 |

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

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **STATUS** | **$I^2C$ Status Bits**<br>The status register of $I^2C$:<br>The three least significant bits are always 0. The five most significant bits contain the status code. Refer to section 6.11.6.4 for detail description. |

### I$^2$C Clock Divided Register (I2C_CLKDIV)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2C_CLKDIV** | I2Cx_BA+0x10 | R/W | I$^2$C Clock Divided Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **DIVIDER** | **I$^2$C Clock Divided Bits**<br>The I$^2$C clock rate bits: Data Baud Rate of I$^2$C = (PCLK) / (4x (DIVIDER+1)).<br>The minimum value of DIVIDER is 4. |

### I$^2$C Time-out Counter Register (I2C_TOCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2C_TOCTL** | I2Cx_BA+0x14 | R/W | I$^2$C Time-out Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | TOCEN | TOCDIV4 | TOIF |

| Bits | Description | |
|---|---|---|
| [31:3] | **Reserved** | Reserved. |
| [2] | **TOCEN** | **Time-out Counter Enable Control**<br>0 = Disabled.<br>1 = Enabled.<br>**Note:** When Enabled, the 14-bit time-out counter will start counting when SI (I2C_CTL[3]) is cleared. Setting flag SI to high will reset counter and re-start up counting after SI is cleared. |
| [1] | **TOCDIV4** | **Time-out Counter Input Clock Divided by 4**<br>0 = Disabled.<br>1 = Enabled.<br>**Note** When Enabled, The time-out period is extend 4 times. |
| [0] | **TOIF** | **Time-out Flag**<br>This bit is set by H/W when I$^2$C time-out happened and it can interrupt CPU if I$^2$C interrupt enable bit (INTEN (I2C_CTL[7])) is set to 1.<br>Write 1 to clear this bit. |

### I$^2$C Slave Address Register 0~3 (I2C_ADDRx)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2C_ADDR0** | I2Cx_BA+0x04 | R/W | I$^2$C Slave Address Register0 | 0x0000_0000 |
| **I2C_ADDR1** | I2Cx_BA+0x18 | R/W | I$^2$C Slave Address Register1 | 0x0000_0000 |
| **I2C_ADDR2** | I2Cx_BA+0x1C | R/W | I$^2$C Slave Address Register2 | 0x0000_0000 |
| **I2C_ADDR3** | I2Cx_BA+0x20 | R/W | I$^2$C Slave Address Register3 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:1] | **ADDR** | **I$^2$C Address Bits**<br>The content of this register is irrelevant when I$^2$C is in Master mode. In Slave mode, the seven most significant bits must be loaded with the chip's own address. The I$^2$C hardware will react if either of the address is matched. |
| [0] | **GC** | **General Call Function**<br>0 = General Call Function Disabled.<br>1 = General Call Function Enabled. |

### I$^2$C Slave Address Mask Register 0~3 (I2C_ADDRMSKx)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2C_ADDRMSK0** | I2Cx_BA+0x24 | R/W | I$^2$C Slave Address Mask Register0 | 0x0000_0000 |
| **I2C_ADDRMSK1** | I2Cx_BA+0x28 | R/W | I$^2$C Slave Address Mask Register1 | 0x0000_0000 |
| **I2C_ADDRMSK2** | I2Cx_BA+0x2C | R/W | I$^2$C Slave Address Mask Register2 | 0x0000_0000 |
| **I2C_ADDRMSK3** | I2Cx_BA+0x30 | R/W | I$^2$C Slave Address Mask Register3 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:1] | **ADDRMSK** | **I$^2$C Address Mask Bits**<br>0 = Mask Disabled (the received corresponding register bit should be exact the same as address register.).<br>1 = Mask Enabled (the received corresponding address bit is don't care.).<br>I$^2$C bus controllers support multiple address recognition with four address mask register. When the bit in the address mask register is set to one, it means the received corresponding address bit is don't-care. If the bit is set to zero, that means the received corresponding register bit should be exact the same as address register. |
| [0] | **Reserved** | Reserved. |

### I$^2$C Wake-up Control Register (I2C_WKCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2C_WKCTL** | I2Cx_BA+0x3C | R/W | I$^2$C Wake-up Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **WKEN** | **I$^2$C Wake-up Enable Control**<br>0 = I$^2$C wake-up function Disabled.<br>1 = I$^2$C wake-up function Enabled. |

### I$^2$C Wake-up Status Register (I2C_WKSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2C_WKSTS** | I2Cx_BA+0x40 | R/W | I$^2$C Wake-up Status Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **WKIF** | **I$^2$C Wake-up Flag**<br>0 = No wake up occurred.<br>1 = Wake up from Power-down mode.<br>**Note:** Software can write 1 to clear this bit. |

## 6.12 Serial Peripheral Interface (SPI)

### 6.12.1 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 NUC505 series contains one set of SPI controller performing a serial-to-parallel conversion on data received from a peripheral device, and a parallel-to-serial conversion on data transmitted to a peripheral device. Also, the SPI controller can be configured as a master or a slave device.

### 6.12.2 Features

- Supports Master or Slave mode operation
- Configurable bit length of a transfer word from 8 to 32-bit
- Provides separate 8-level depth transmit and receive FIFO buffers
- Supports MSB first or LSB first transfer sequence
- Supports byte reorder function
- Supports Byte or Word Suspend mode
- Supports 3-wire, no slave select signal, bi-direction interface
- Up to 2 sets of SPI controllers

### 6.12.3 Block Diagram

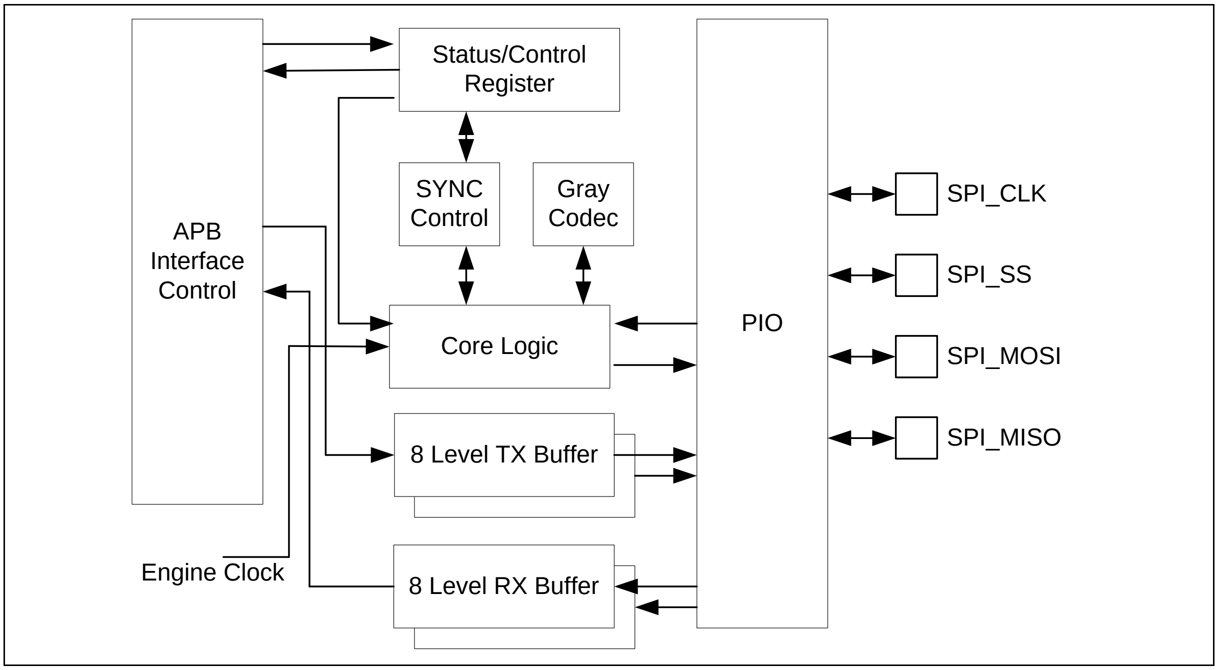


Figure 6.12-1 SPI Block Diagram

### 6.12.4 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 divisor. The SPI0SEL (CLK_CLKDIV2[28]) and SPI1SEL (CLK_CLKDIV2[29]) register determine the clock source of the SPI0 and SPI1 engine clock. The clock source can be HXT or PLL_FOUT. The DIVIDER (SPI_CLKDIV[7:0]) register determines the divisor of the clock rate calculation.

In Master mode, 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. (If the clock source of SPI engine clock and APB clock are different, the frequency of SPI engine clock shall slower 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 (SPI_CTL[18]) to communicate with the off-chip SPI Slave or Master device. The application block diagrams in Master and Slave mode are shown below.

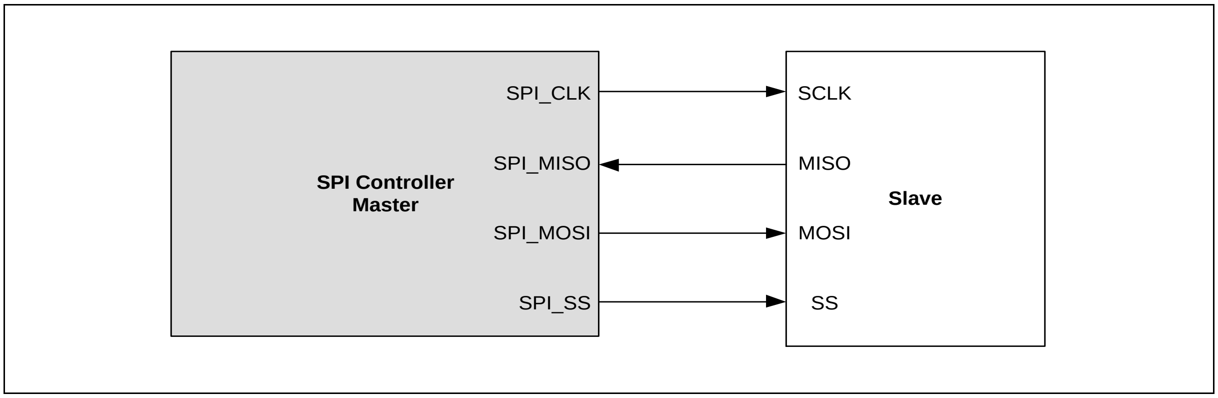


Figure 6.12-2 SPI Master Mode Application Block Diagram

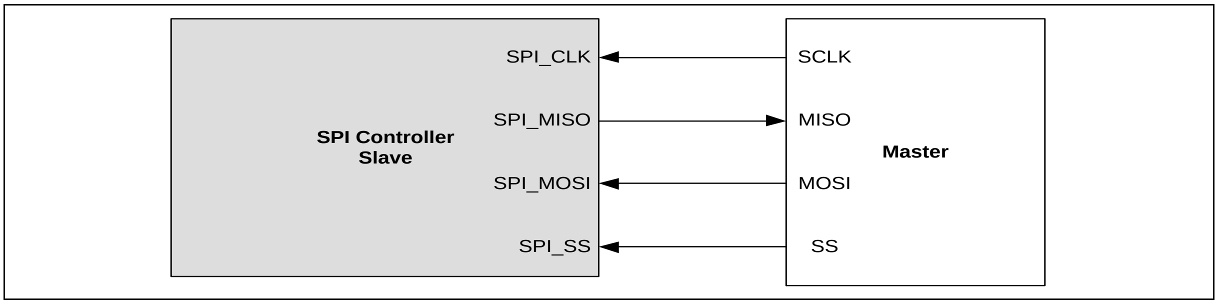


Figure 6.12-3 SPI Slave Mode Application Block Diagram

#### Slave Selection

In Master mode, the SPI controller can drive up to 1 off-chip slave device through the slave select output pins SPI_SS. In Slave mode, the off-chip master device drives the slave select signal from the SPI_SS 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 SSACTPOL (SPI_SSCTL[2]). The selection of slave select conditions depends on what type of peripheral slave/master device is connected.

### Automatic Slave Selection

In Master mode, if the bit AUTOSS (SPI_SSCTL[3]) is set, the slave select signals will be generated automatically and output to the SPI_SS pin according to whether SS[0] (SPI_SSCTL[0]) is enabled or not. This means that the slave select signal, which is selected in SPI_SSCTL[0], will be asserted by the SPI controller when the SPI data transfer is started by writing the transfer data into the FIFO and will be de-asserted after one transaction is finished. If the AUTOSS (SPI_SSCTL[3]) bit is cleared, the slave select output signals will be asserted/de-asserted by manual setting/clearing the related bit of SPI_SSCTL[0]. The active state of the slave select output signals is specified in SSACTPOL (SPI_SSCTL[2]).

In Master mode, if the value of SUSPITV[3:0] is less than 3 and the AUTOSS (SPI_SSCTL[3]) 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 3 engine clock periods between two successive transactions.

### Clock Polarity

The CLKPOL (SPI_CTL[3]) defines the SPI clock idle state. If CLKPOL (SPI_CTL[3]) = 1, the output SPI clock is idle at high state; if CLKPOL (SPI_CTL[3]) = 0, it is idle at low state.

### Transmit/Receive Bit Length

The bit length of a transaction word is defined in DWIDTH (SPI_CTL[12:8]) and can be configured up to 32-bit length in a transaction word for transmitting and receiving.

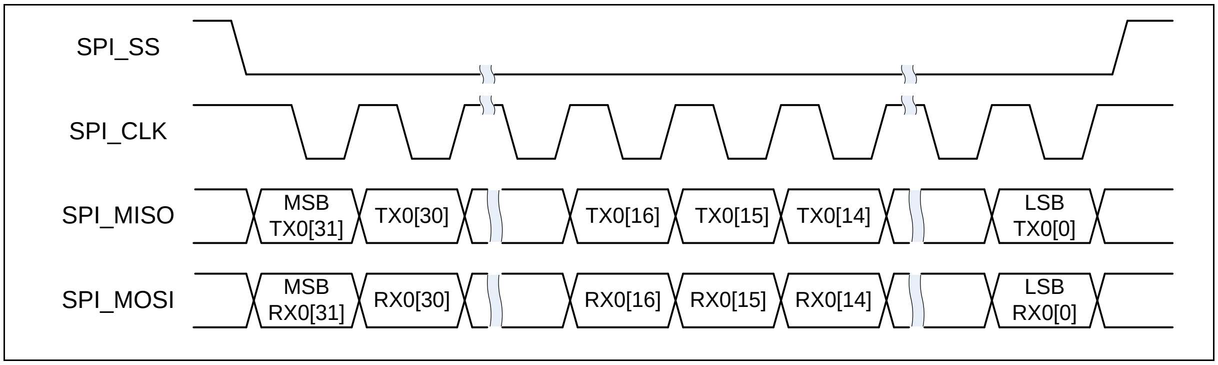


Figure 6.12-4 Bit in One Transaction

### LSB/MSB First

The LSB (SPI_CTL[13]) 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 TXNEG (SPI_CTL[2]) defines the data transmitted out either on negative edge or on positive edge of SPI clock.

### Receive Edge

The RXNEG (SPI_CTL[1]) defines the data received either on negative edge or on positive edge of SPI clock.

The settings of TXNEG (SPI_CTL[2]) and RXNEG (SPI_CTL[1]) are mutual exclusive. In other words, do not transmit and receive data at the same clock edge.

### Word Suspend

The four bit fields of SUSPITV (SPI_CTL[7:4]) provide a configurable suspend interval, 0.5 ~ 15.5 SPI clock periods, between two successive transaction words in Master mode. The definition of the suspend interval is the interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value of SUSPITV (SPI_CTL[7:4]) is 0x3 (3.5 SPI clock cycles).

### Byte Reorder

When the transfer is set as MSB first (LSB = 0) and the REORDER (SPI_CTL[19]) 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 (DWIDTH = 0). The sequence of transmitted/received data will be BYTE0, BYTE1, BYTE2, and then BYTE3. If the DWIDTH is set as 24-bit transfer mode, the data in TX buffer and RX buffer will be rearranged as [unknown byte, BYTE0, BYTE1, BYTE2]. The SPI controller will transmit/receive data with the sequence of BYTE0, BYTE1 and then BYTE2. Each byte will be transmitted/received with MSB first. The rule of 16-bit mode is the same as above. Byte reorder function is only available when DWIDTH is configured as 16, 24, and 32 bits.

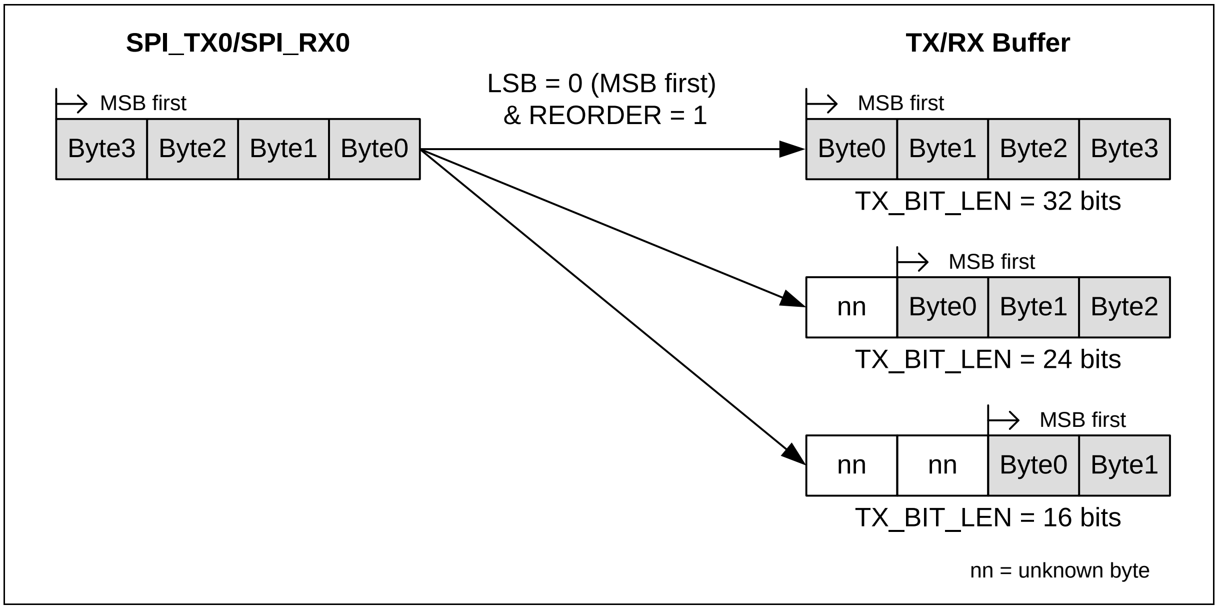


Figure 6.12-5 Byte Reorder Function

### Byte Suspend

In Master mode, if REORDER (SPI_CTL[19]) is set to 1, a suspend interval of 0.5 ~ 15.5 SPI clock periods will be inserted by hardware between two successive bytes in a transaction word. Both settings of byte suspend interval and word suspend interval are configured in SUSPITV (SPI_CTL[7:4]).

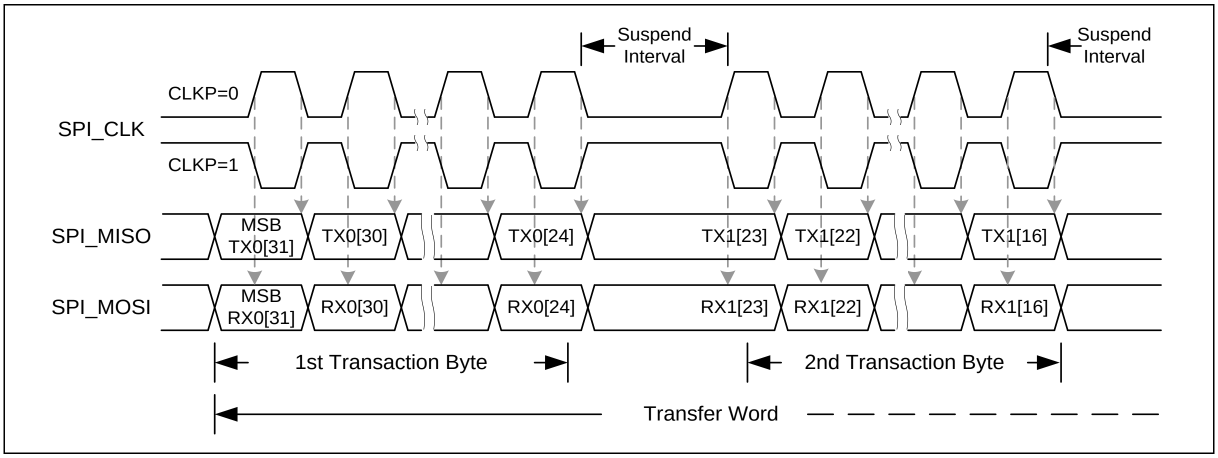


Figure 6.12-6 Timing Waveform for Byte Suspend

**3-Wire Mode**

When the SLV3WIRE (SPI_SSCTL[4]) 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 SLV3WIRE (SPI_SSCTL[4]) 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 SPI_SS pin can be configured as a GPIO. When the SLV3WIRE (SPI_SSCTL[4]) bit is set to 1, the SPI slave will be ready to transmit/receive data after the SPIEN (SPI_CTL [0]) bit is set to 1.

**8-Level FIFO Buffer**

The SPI controllers equip with eight 32-bit wide transmit and receive FIFO buffers.

The transmit FIFO buffer is an 8-level depth, 32-bit wide, first-in, first-out register buffer. 8 data can be written to the transmit FIFO buffer in advance through software by writing the SPI_TX register. The data stored in the transmit FIFO buffer will be read and sent out by the transmission control logic. If the 8-level transmit FIFO buffer is full, the TXFULL **(**SPI_STATUS[17]) 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-level transmit FIFO buffer is empty, the TXEMPTY **(**SPI_STATUS[16]) bit will be set to 1. Notice that the TXEMPTY **(**SPI_STATUS[16]) flag is set to 1 while the last transaction is still in progress. In Master mode, both the BUSY (SPI_STATUS[0]) and TXEMPTY **(**SPI_STATUS[16]) 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-level depth, 32-bit wide, first-in, first-out register buffer. The receive control logic will store the received data to this buffer. The FIFO buffer data can be read from SPI_RX register by software. There are FIFO related status bits, like RXEMPTY **(**SPI_STATUS[8]) and RXFULL **(**SPI_STATUS[9]), to indicate the current status of FIFO buffer.

The transmitting and receiving threshold can be set through software by setting the TXTH (SPI_FFCTL[30:24]), and RXTH (SPI_FIFOCTL[26:24]) settings. When the count of valid data stored in transmit FIFO buffer is less than or equal to TXTH (SPI_FFCTL[30:24]) setting, the TXTHIF, SPI_STATUS[18], bit will be set to 1. When the count of valid data stored in receive FIFO buffer is larger than RXTH (SPI_FIFOCTL[26:24]) setting, the RXTHIF (SPI_STATUS[10]), bit will be set to 1.

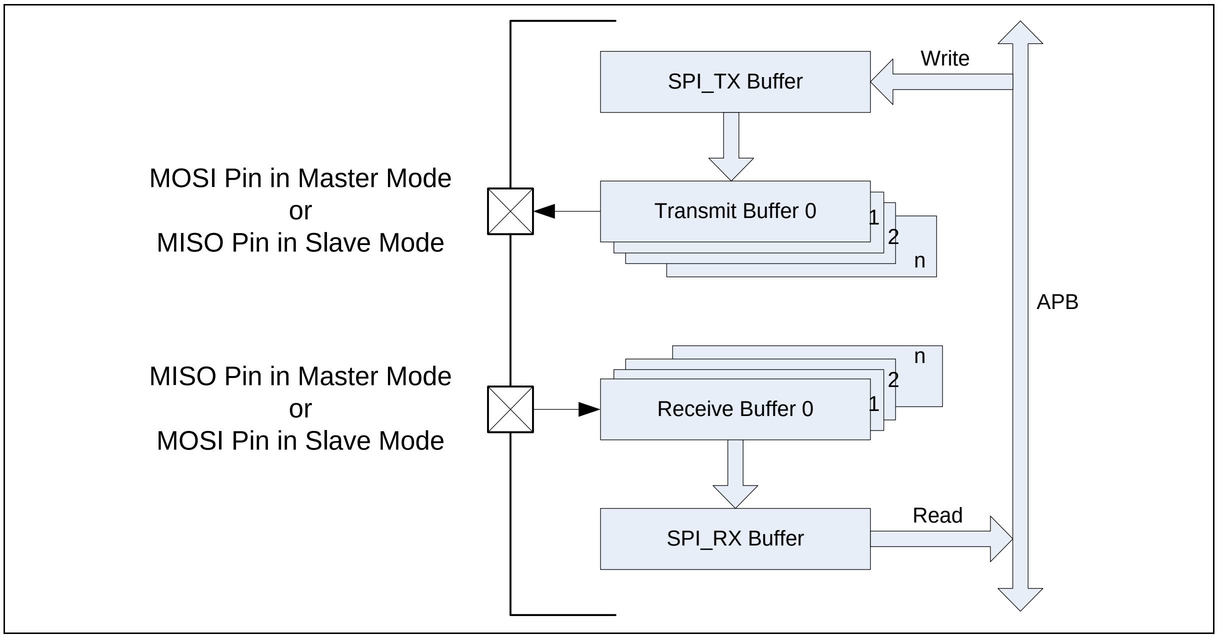


Figure 6.12-7 FIFO Mode Block Diagram

In Master mode, the first datum is written to the SPI_TX register, the TXEMPTY **(**SPI_STATUS[16]) 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_TX register immediately. The SPI controller will insert a suspend interval between two successive transactions and the period of suspend interval is decided by the setting of SUSPITV (SPI_CTL [7:4]). User can write data into SPI_TX register as long as the TXFULL **(**SPI_STATUS[17]) flag is 0.

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

In Master mode, during receiving operation, the serial data are received from SPI_MISO0/1 pin and stored to receive FIFO buffer. The RXEMPTY **(**SPI_STATUS[8]) flag will be cleared to 0 while the receive FIFO buffer contains unread data. The received data can be read by software from SPI_RX register as long as the RXEMPTY **(**SPI_STATUS[8]) flag is 0. If the receive FIFO buffer contains 8 unread data, the RXFULL **(**SPI_STATUS[9]) flag will be set to 1. The SPI controller will stop receiving data until the SPI_RX register is read by software.

In Slave mode, during transmission operation, when data is written to the SPI_TX register by software, the data will be loaded into transmit FIFO buffer and the TXEMPTY **(**SPI_STATUS[16]) 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_TX register as long as the TXFULL **(**SPI_STATUS[17]) flag is 0. After all data have been drawn out by the SPI transmission logic unit and the SPI_TX register is not updated by software, the TXEMPTY **(**SPI_STATUS[16]) flag will be set to 1.

If there is no any data is written to the SPI_TX register, the under-run event, TXUFIF (SPI_STATUS[19]) will active when the slave select active and the serial clock input this controller. Under the previous condition, the Slave mode error 1, SLVURIF (SPI_STATUS[7]), will be set to 1 when SS goes to inactive state and transmit under-run occurs.

In Slave mode, during receiving operation, the serial data is received from SPI_MOSI0/1 pin and stored to SPI_RX register. The reception mechanism is similar to Master mode reception operation. If the receive FIFO buffer contains 8 unread data, the RXFULL **(**SPI_STATUS[9]) flag will be set to 1 and the RXOVIF **(**SPI_STATUS[11]) will be set 1 if there is more serial data is received from SPI0OSI and follow-up data will be dropped. If the receive bit counter mismatch with the DWIDTH (SPI_CTL[12:8]) when the slave select line goes to inactive state, the Slave mode error 0, SLVBEIF (SPI_STATUS[6]), will be set to 1.

When the Slave select is active and the value of SLVTOCNT (SPI_SSCTL[31:16]) is not 0, the Slave time-out counter in the SPI controller logic will start after the serial clock input. This counter will be cleared after one transaction is done or the SLVTOCNT (SPI_SSCTL[31:16]) is set to 0. If the value of the time-out counter greater or equal than the value of SLVTOCNT (SPI_SSCTL[31:16]) before one transaction done, the slave time-out event occurs and the SLVTOIF (SPI_STATUS[5]) will be set to 1.

A receive time-out function is built-in in this controller. When the receive FIFO is not empty and no read operation in receive FIFO over 64 SPI clock period in Master mode or over 576 SPI engine clock period in Slave mode, the receive time-out occurs and the SLVTOIF (SPI_STATUS[5]) be set to 1. When the receive FIFO is read by user, the time-out status will be cleared automatically.

**SPI unit transfer interrupt**

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

**SPI slave select interrupt**

In Slave mode, there are slave select active and in-active interrupt flag, SSACTIF (SPI_STATUS[2]) and SSINAIF (SPI_STATUS[3]), will be set to 1 when the SPIEN (SPI_CTL[0]) and SLAVE (SPI_CTL[18]) bits were set to 1 and slave senses the slave select signal active or inactive. The SPI controller will issue an interrupt if the SSINAIEN (SPI_SSCTL[13]) or SSACTIEN (SPI_SSCTL[12]) are set to 1.

**Slave Time-out interrupt**

In Slave mode, there is slave time-out function for user to know that there is serial clock input but one transaction is not finished over the period of SLVTOCNT (SPI_SSCTL[31:16]) based on engine clock.

When the Slave select is active and the value of SLVTOCNT (SPI_SSCTL[31:16]) is not 0, the Slave time-out counter in the SPI controller logic will start after the serial clock input. This counter will be cleared after one transaction is done or the SLVTOCNT (SPI_SSCTL[31:16]) is set to 0. If the value of the time-out counter is greater than or equal to the value of SLVTOCNT (SPI_SSCTL[31:16]) before one transaction done, the slave time-out event occurs and the SLVTOIF (SPI_STATUS[5]) will be set to 1. The SPI controller will issue an interrupt if the SLVTOIEN (SPI_SSCTL[5]) is set to 1.

**Slave Error 0 interrupt**

In Slave mode, if the transmit/ receive bit count mismatch with the DWIDTH (SPI_CTL[12:8]) when the slave select line goes to inactive state, the Slave mode error 0, SLVBEIF (SPI_STATUS[6]) will be set to 1. The SPI controller will issue an interrupt if the SLVBEIEN (SPI_SSCTL[8]) is set to 1.

1. In Slave transmit mode, if there is bit length transmit error (bit count mismatch), the user shall set the TXRST (SPI_FIFOCTL[1]) bit and write the transmit datum again to restart the next transaction.

2. If the slave select active but there is no any serial clock input, the SLVBEIF (SPI_STATUS[6]) also active when the slave select goes to inactive state.

**Slave Under-run and Slave Error 1 interrupts**

In Slave mode, if there is no any data is written to the SPI_TX register, the under-run event, TXUFIF (SPI_STATUS[19]) will active when the slave select active and the serial clock input this controller.

Under the previous condition, the Slave mode error 1, SLVURIF (SPI_STATUS[7]), will be set to 1 when SS goes to inactive state and transmit under-run occurs. The SPI controller will issue an interrupt if the SLVURIEN (SPI_SSCTL[9]) is set to 1.

In SLV3WIRE mode, the slave select bus active all the time so that the user shall polling the TXUFIF bit to know if there is transmit under-run event or not.

**Receive Over-run interrupt**

In Slave mode, if the receive FIFO buffer contains 8 unread data, the RXFULL **(**SPI_STATUS[9]) flag will be set to 1 and the RXOVIF **(**SPI_STATUS[11]) will be set 1 if there is more serial data is received from SPI0OSI and the RXOVIF **(**SPI_STATUS[11]) will be set to 1 and follow-up data will be dropped. The SPI controller will issue an interrupt if the SLVOVR_INTEN (SPI_FIFOCTL[5]) is set to 1.

**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, RXTOIEN (SPI_FIFOCTL[4]) is set to 1.

**Transmit FIFO interrupt**

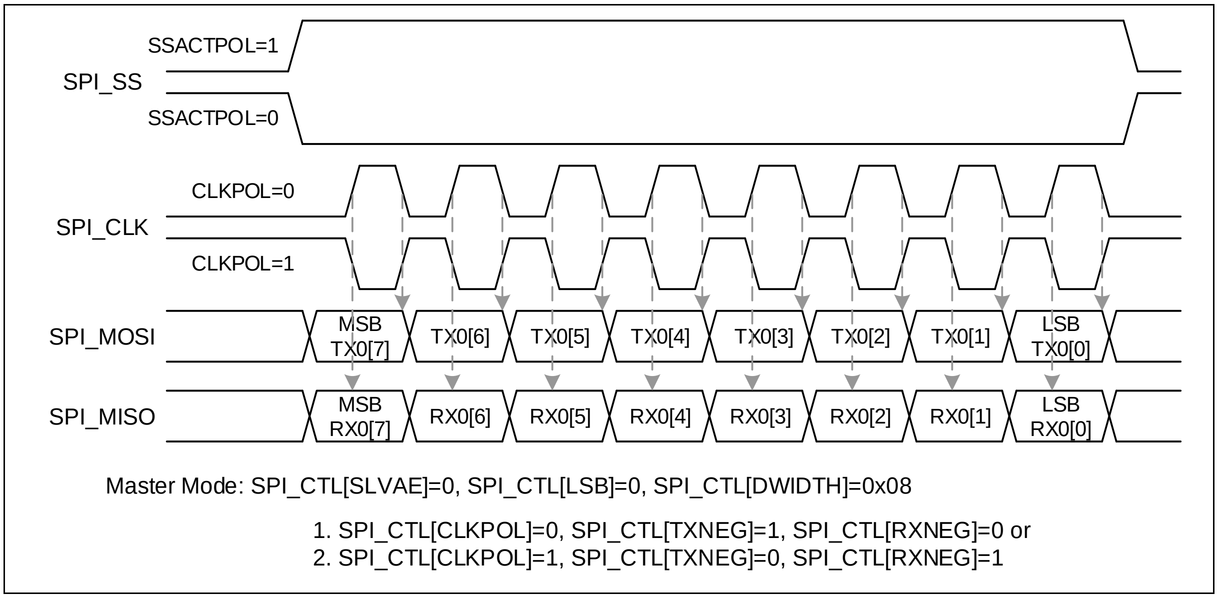
In FIFO mode, if the valid data count of the transmit FIFO buffer is less than or equal to the setting value of TXTH, the transmit FIFO interrupt flag will be set to 1. The SPI controller will generate a transmit FIFO interrupt to the system if the transmit FIFO interrupt enable bit, TXTHIEN (SPI_FIFOCTL[3]),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

**Receive FIFO interrupt**

In FIFO mode, if the valid data count of the receive FIFO buffer is larger than the setting value of RXTH, 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, RXTHIEN (SPI_FIFOCTL[2]), is set to 1.

### 6.12.5 Timing Diagram

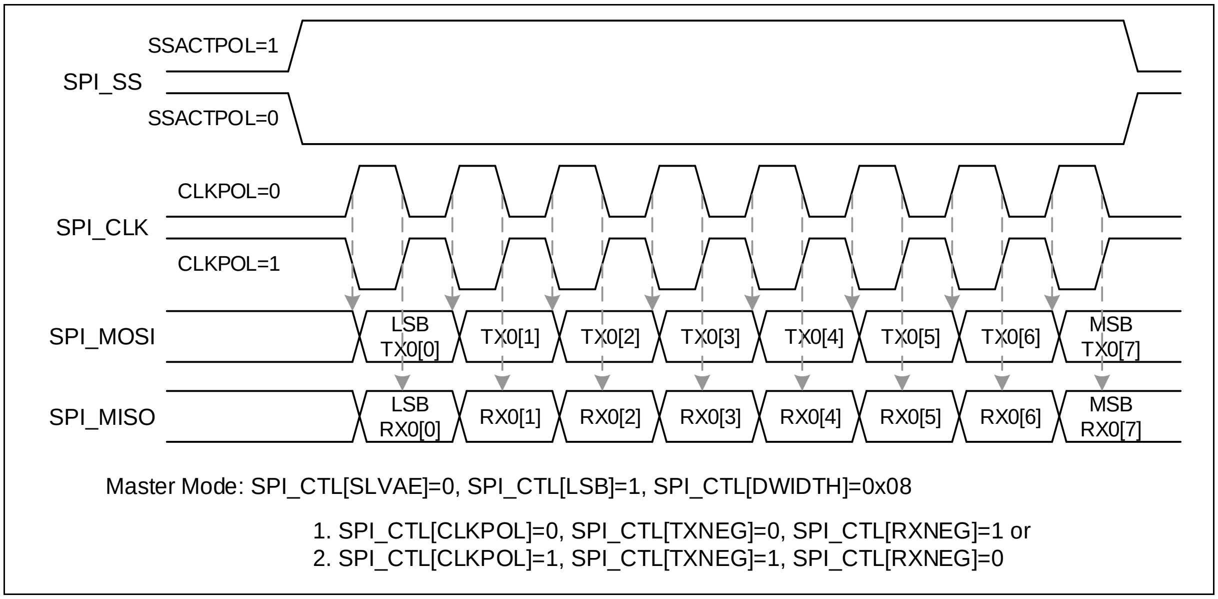
The active state of slave select signal can be defined by setting the SSACTPOL (SPI_SSCTL[2]). The SPI clock which is in idle state can be configured as high or low state by setting the CLKPOL (SPI_CTL[3]). It also provides the bit length of a transaction word in DWIDTH (SPI_CTL[12:8]), and transmitting/receiving data from MSB or LSB first in LSB (SPI_CTL[13]). User can also select which edge of SPI clock to transmit/receive data in TXNEG/RXNEG (SPI_CTL[2:1]). Four SPI timing diagrams for master/slave operations and the related settings are shown below.



Master Mode: SPI_CTL[SLVAE]=0, SPI_CTL[LSB]=0, SPI_CTL[DWIDTH]=0x08

1. SPI_CTL[CLKPOL]=0, SPI_CTL[TXNEG]=1, SPI_CTL[RXNEG]=0 or
2. SPI_CTL[CLKPOL]=1, SPI_CTL[TXNEG]=0, SPI_CTL[RXNEG]=1

Figure 6.12-8 SPI Timing in Master Mode



Master Mode: SPI_CTL[SLVAE]=0, SPI_CTL[LSB]=1, SPI_CTL[DWIDTH]=0x08

1. SPI_CTL[CLKPOL]=0, SPI_CTL[TXNEG]=0, SPI_CTL[RXNEG]=1 or
2. SPI_CTL[CLKPOL]=1, SPI_CTL[TXNEG]=1, SPI_CTL[RXNEG]=0

Figure 6.12-9 SPI Timing in Master Mode (Alternate Phase of SPICLK)

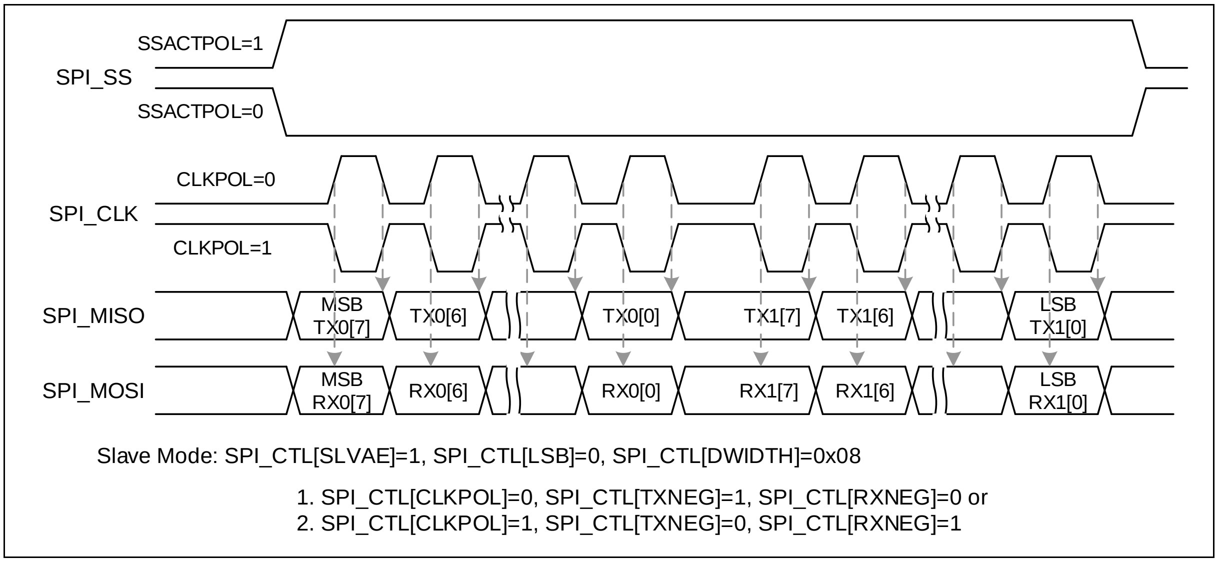

SSACTPOL=1
SPI_SS
SSACTPOL=0
CLKPOL=0
SPI_CLK
CLKPOL=1
SPI_MISO
MSB
TX0[7]
TX0[6]
TX0[0]
TX1[7]
TX1[6]
LSB
TX1[0]
SPI_MOSI
MSB
RX0[7]
RX0[6]
RX0[0]
RX1[7]
RX1[6]
LSB
RX1[0]
Slave Mode: SPI_CTL[SLVAE]=1, SPI_CTL[LSB]=0, SPI_CTL[DWIDTH]=0x08
1. SPI_CTL[CLKPOL]=0, SPI_CTL[TXNEG]=1, SPI_CTL[RXNEG]=0 or
2. SPI_CTL[CLKPOL]=1, SPI_CTL[TXNEG]=0, SPI_CTL[RXNEG]=1

Figure 6.12-10 SPI Timing in Slave Mode

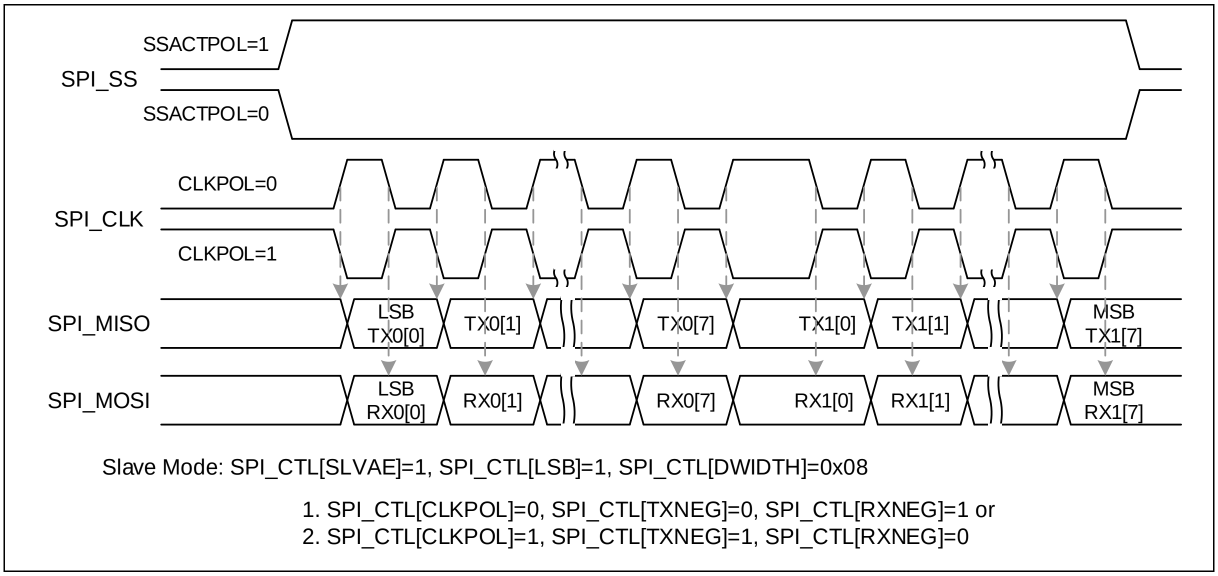

SSACTPOL=1
SPI_SS
SSACTPOL=0
CLKPOL=0
SPI_CLK
CLKPOL=1
SPI_MISO
LSB
TX0[0]
TX0[1]
TX0[7]
TX1[0]
TX1[1]
MSB
TX1[7]
SPI_MOSI
LSB
RX0[0]
RX0[1]
RX0[7]
RX1[0]
RX1[1]
MSB
RX1[7]
Slave Mode: SPI_CTL[SLVAE]=1, SPI_CTL[LSB]=1, SPI_CTL[DWIDTH]=0x08
1. SPI_CTL[CLKPOL]=0, SPI_CTL[TXNEG]=0, SPI_CTL[RXNEG]=1 or
2. SPI_CTL[CLKPOL]=1, SPI_CTL[TXNEG]=1, SPI_CTL[RXNEG]=0

Figure 6.12-11 SPI Timing in Slave Mode (Alternate Phase of SPICLK)

### 6.12.6 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_CLKDIV [7:0]) register to determine the output frequency of SPI clock.
2) Write the SPI_SSCTL register a proper value for the related settings of Master mode:
   1. Disable the Automatic Slave Select bit AUTOSS (SPI_SSCTL[3] = 0).
   2. Select low level trigger output of slave select signal in the Slave Select Active Level bit SSACTPOL (SPI_SSCTL[2] = 0).
   3. Select slave select signal to be output active at the I/O pin by setting the Slave Select Register bits SS[0] (SPI_SSCTL[0]) to active the off-chip slave device.
3) Write the related settings into the SPI_CTL register to control the SPI master actions
   1. Set this SPI controller as master device in SLAVE (SPI_CTL[18] = 0).
   2. Force the SPI clock idle state at low in CLKPOL bit (SPI_CTL[3] = 0).
   3. Select data transmitted at negative edge of SPI clock in TXNEG bit (SPI_CTL[2] = 1).
   4. Select data latched at positive edge of SPI clock in RXNEG (SPI_CTL[1]) to 0.
   5. Set the bit length of word transfer as 8-bit in DWIDTH (SPI_CTL[12:8]) to 0x08.
   6. Set MSB transfer first in MSB (SPI_CTL[13]) to 0.
4) Enable the SPIEN (SPI_CTL[0]) to start the data transfer with the SPI interface.
5) If this SPI master attempts to transmit (write) one byte data to the off-chip slave device, write the byte data that will be transmitted into the SPI_TX register.
6) Waiting for SPI interrupt (if the Interrupt Enable IE bit is set) or just polling the BUSY (SPI_STATUS[0]) till it is cleared to 0 by hardware automatically.
7) Read out the received one byte data from SPI_RX[7:0].
8) Go to 5) to continue another data transfer or set SS [0] (SPI_SSCTL[0]) to 0 to inactivate the off-chip slave device.

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

- Data bit is latched on positive edge of SPI 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_SSCTL register a proper value for the related settings of Slave mode:

   Select high level for the input of slave select signal by setting the Slave Select Active Level bit SSACTPOL (SPI_SSCTL[2] = 1).

2) Write the related settings into the SPI_CTL register to control this SPI slave actions
   1. Set the SPI controller as slave device in SLAVE (SPI_CTL[18]) to 1.
   2. Select the SPI clock idle state at high in CLKPOL (SPI_CTL[3]) to 1.
   3. Select data transmitted at negative edge of SPI clock in TXNEG (SPI_CTL[2]) to 1.
   4. Select data latched at positive edge of SPI clock in RXNEG (SPI_CTL[1]) to 0.
   5. Set the bit length of word transfer as 8-bit in DWIDTH (SPI_CTL[12:8]) to 0x08.
   6. Set LSB transfer first in LSB (SPI_CTL[13]) to 1.

3) Enable the SPIEN (SPI_CTL[0]) 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.

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

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

6) Waiting for SPI interrupt (if the Interrupt Enable UNITIEN (SPI_CTL[17]) is set).

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

8) Go to 4) to continue another data transfer or stop data transfer.

### 6.12.7 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPI Base Address:**<br>**SPI0_BA = 0x400E_9000**<br>**SPI1_BA = 0x400E_1000** | | | | |
| **SPI_CTL**<br>**x = 0,1** | SPIx_BA+0x00 | R/W | SPI Control Register | 0x0000_0034 |
| **SPI_CLKDIV**<br>**x = 0,1** | SPIx_BA+0x04 | R/W | SPI Clock Divider Register | 0x0000_0000 |
| **SPI_SSCTL**<br>**x = 0,1** | SPIx_BA+0x08 | R/W | SPI Slave Select Control Register | 0x0000_0000 |
| **SPI_FIFOCTL**<br>**x = 0,1** | SPIx_BA+0x10 | R/W | SPI FIFO Control Register | 0x4400_0000 |
| **SPI_STATUS**<br>**x = 0,1** | SPIx_BA+0x14 | R/W | SPI Status Register | 0x0005_0110 |
| **SPI_TX**<br>**x = 0,1** | SPIx_BA+0x20 | W | SPI Data Transmit Register | 0x0000_0000 |
| **SPI_RX**<br>**x = 0,1** | SPIx_BA+0x30 | R | SPI Data Receive Register | 0x0000_0000 |

### 6.12.8 Register Description

#### SPI Control Register (SPI_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SPI_CTL | SPIx_BA+0x00 | R/W | SPI Control Register | 0x0000_0034 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | REORDER | SLAVE | UNITIEN | Reserved |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | LSB | DWIDTH | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| SUSPITV | | | | CLKPOL | TXNEG | RXNEG | SPIEN |

| Bits | Description | |
|---|---|---|
| [31:20] | Reserved | Reserved. |
| [19] | REORDER | **Byte Reorder Function Enable Control**<br>0 = Byte reorder function Disabled.<br>1 = Byte reorder function Enabled. A byte suspend interval will be inserted among each byte. The period of the byte suspend interval depends on the setting of SUSPITV (SPI_CTL[7:4]).<br>**Note:** Byte reorder function is only available if DWIDTH is defined as 16, 24, and 32 bits. |
| [18] | SLAVE | **Slave Mode Enable Control**<br>0 = Master mode.<br>1 = Slave mode. |
| [17] | UNITIEN | **Unit Transfer Interrupt Enable Control**<br>0 = SPI unit transfer interrupt Disabled.<br>1 = SPI unit transfer interrupt Enabled. |
| [16:14] | Reserved | Reserved. |
| [13] | LSB | **Send LSB First**<br>0 = The MSB, which bit of transmit/receive register depends on the setting of DWIDTH, is transmitted/received first.<br>1 = The LSB, bit 0 of the SPI TX register, is sent first to the SPI data output pin, and the first bit received from the SPI data input pin will be put in the LSB position of the RX register (bit 0 of SPI_RX). |
| [12:8] | DWIDTH | **Data Transmit Bit Width**<br>This field specifies how many bits can be transmitted / received in one transaction. The minimum bit length is 8 bits and can up to 32 bits.<br>DWIDTH = 0x08 …. 8 bits.<br>DWIDTH = 0x09 …. 9 bits. |

| | | |
|---|---|---|
| | | ……<br>DWIDTH = 0x1F …. 31 bits.<br>DWIDTH = 0x00 …. 32 bits. |
| [7:4] | **SUSPITV** | **Suspend Interval (Master Only)**<br>The four bits provide configurable suspend interval between two successive transmit/receive transaction in a transfer. The definition of the suspend interval is the interval between the last clock edge of the preceding transaction word and the first clock edge of the following transaction word. The default value is 0x3. The period of the suspend interval is obtained according to the following equation.<br>(SUSPITV[3:0]+0.5) * period of SPICLK clock cycle<br>Example:<br>SUSPITV = 0x0 …. 0.5 SPICLK clock cycle.<br>SUSPITV = 0x1 …. 1.5 SPICLK clock cycle.<br>……<br>SUSPITV = 0xE …. 14.5 SPICLK clock cycle.<br>SUSPITV = 0xF …. 15.5 SPICLK clock cycle. |
| [3] | **CLKPOL** | **Clock Polarity**<br>0 = SPICLK is idle low.<br>1 = SPICLK is idle high. |
| [2] | **TXNEG** | **Transmit on Negative Edge**<br>0 = Transmitted data output signal is changed on the rising edge of SPICLK.<br>1 = Transmitted data output signal is changed on the falling edge of SPICLK. |
| [1] | **RXNEG** | **Receive on Negative Edge**<br>0 = Received data input signal is latched on the rising edge of SPICLK.<br>1 = Received data input signal is latched on the falling edge of SPICLK. |
| [0] | **SPIEN** | **SPI Transfer Control Enable Bit**<br>0 = Transfer control bit Disabled.<br>1 = Transfer control bit Enabled.<br>**Note1:** In Master mode, the transfer will start when there is data in the FIFO buffer after this is set to 1. In Slave mode, this device is ready to receive data when this bit is set to 1.<br>**Note2:** The byte reorder function is not supported when the Quad or Dual I/O mode is enabled. |

### SPI Divider Register (SPI_CLKDIV)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPI_CLKDIV** | SPIx_BA+0x04 | R/W | SPI Clock Divider Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **DIVIDER** | **Clock Divider 1 Register**<br>The value in this field is the frequency divider for generating the SPI engine clock, $f_{spi\_eclk}$, and the SPI serial clock of SPI master. The frequency is obtained according to the following equation.<br>$f_{spi\_eclk} = \frac{f_{spi\_clock\_src}}{(DIVIDER+1)}$<br>where<br>$f_{spi\_clock\_src}$ is the SPI engine clock source, which is defined in the clock control, clock control register. |

### SPI Slave Select Register (SPI_SSCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SPI_SSCTL | SPIx_BA+0x08 | R/W | SPI Slave Select Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| SLVTOCNT | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| SLVTOCNT | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | SSINAIEN | SSACTIEN | Reserved | | SLVURIEN | SLVBEIEN |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | SLVTORST | SLVTOIEN | SLV3WIRE | AUTOSS | SSACTPOL | Reserved | SS |

| Bits | Description | |
|---|---|---|
| [31:16] | **SLVTOCNT** | **Slave Mode Time-out Period**<br>In Slave mode, these bits indicate the time-out period when there is serial clock input during slave select active. The clock source of the time-out counter is Slave engine clock. If the value is 0, it indicates the slave mode time-out function is disabled. |
| [15:14] | **Reserved** | Reserved. |
| [13] | **SSINAIEN** | **Slave Select Inactive Interrupt Enable Control**<br>0 = Slave select inactive interrupt Disable.<br>1 = Slave select inactive interrupt Enable. |
| [12] | **SSACTIEN** | **Slave Select Active Interrupt Enable Control**<br>0 = Slave select active interrupt Disable.<br>1 = Slave select active interrupt Enable. |
| [11:10] | **Reserved** | Reserved. |
| [9] | **SLVURIEN** | **Slave Mode Error 1 Interrupt Enable Control**<br>0 = Slave mode error 1 interrupt Disable.<br>1 = Slave mode error 1 interrupt Enable. |
| [8] | **SLVBEIEN** | **Slave Mode Error 0 Interrupt Enable Control**<br>0 = Slave mode error 0 interrupt Disable.<br>1 = Slave mode error 0 interrupt Enable. |
| [7] | **Reserved** | Reserved. |
| [6] | **SLVTORST** | **Slave Mode Time-out FIFO Clear**<br>0 = Slave mode Time-out FIFO Clear Disable.<br>1 = Slave mode Time-out FIFO Clear Enable.<br>**Note:** Both the FIFO clear function, TX_CLK and RXRST, active automatically when there is a slave mode time-out event. |

| | | |
|---|---|---|
| [5] | **SLVTOIEN** | **Slave Mode Time-out Interrupt Enable Control**<br>0 = Slave mode time-out interrupt Disabled.<br>1 = Slave mode time-out interrupt Enabled. |
| [4] | **SLV3WIRE** | **Slave 3-wire Mode Enable Control**<br>This is used to ignore the slave select signal in Slave mode. The SPI controller can work with 3-wire interface including SPI_CLK, SPI_MISO, and SPI_MOSI.<br>0 = 2-wire bi-direction interface.<br>1 = 3-wire bi-direction interface. |
| [3] | **AUTOSS** | **Automatic Slave Select Function Enable (Master Only)**<br>0 = If this bit is cleared, slave select signals will be asserted/de-asserted by setting /clearing the corresponding bit of SPI_SSCTL[0].<br>1 = If this bit is set, SS signal will be generated automatically. It means that device/slave select signal, which is set in SPI_SSCTL[0], will be asserted by the SPI controller when transmit/receive is started, and will be de-asserted after each transmit/receive is finished. |
| [2] | **SSACTPOL** | **Slave Select Active Level**<br>This bit defines the active status of slave select signal (SS).<br>0 = The slave select signal SS is active on low-level.<br>1 = The slave select signal SS is active on high-level. |
| [1] | **Reserved** | Reserved. |
| [0] | **SS** | **Slave Selection Control (Master Only)**<br>If AUTOSS bit is cleared to 0,<br>0 = set the SPIn_SS line to inactive state.<br>1 = set the SPIn_SS line to active state.<br>If the AUTOSS bit is set to 1,<br>0 = Keep the SPIn_SS line at inactive state.<br>1 = SPIn_SS line will be automatically driven to active state for the duration of data transfer, and will be driven to inactive state for the rest of the time. The active state of SPIn_SS is specified in SSACTPOL (SPI_SSCTL[2]). |

### SPI FIFO Control Register (SPI_FIFOCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPI_FIFOCTL** | SPIx_BA+0x10 | R/W | SPI FIFO Control Register | 0x4400_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | TXTH | | | Reserved | RXTH | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| Reserved | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | | | | | | TXFBCLR | RXFBCLR |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| TXUFIEN | TXUFPOL | RXOVIEN | RXTOIEN | TXTHIEN | RXTHIEN | TXRST | RXRST |

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30:28] | **TXTH** | **Transmit FIFO Threshold**<br>If the valid data count of the transmit FIFO buffer is less than or equal to the TXTH setting, the TXTHIF (SPI_STATUS[18]) will be set to 1, else the TXTHIF (SPI_STATUS[18]) will be cleared to 0. |
| [27] | **Reserved** | Reserved. |
| [26:24] | **RXTH** | **Receive FIFO Threshold**<br>If the valid data count of the receive FIFO buffer is larger than the RXTH setting, the RXTHIF (SPI_STATUS[10]) will be set to 1, else the RXTHIF (SPI_STATUS[10]) will be cleared to 0. |
| [23:10] | **Reserved** | Reserved. |
| [9] | **TXFBCLR** | **Clear Transmit FIFO**<br>0 = No effect.<br>1 = Clear transmit FIFO only.<br>**Note:** Auto cleared by Hardware**.** |
| [8] | **RXFBCLR** | **Clear Receive FIFO**<br>0 = No effect.<br>1 = Clear receive FIFO only.<br>**Note:** Auto cleared by Hardware**.** |
| [7] | **TXUFIEN** | **Slave Transmit Under Run Interrupt Enable Control**<br>0 = Slave Transmit FIFO under-run interrupt Disabled.<br>1 = Slave Transmit FIFO under-run interrupt Enabled. |
| [6] | **TXUFPOL** | **Transmit Under-run Data Out**<br>0 = The SPI data out is keep 0 if there is transmit under-run event in Slave mode.<br>1 = The SPI data out is keep 1 if there is transmit under-run event in Slave mode. |

| | | |
|---|---|---|
| | | **Note1:** The under run event is active after the serial clock input and the hardware synchronous, so that the first 1~3 bit (depending on the relation between system clock and the engine clock) data out will be the last transaction data.<br>**Note2:** If the frequency of system clock approach to engine clock, they may need 3-bit time to report the transmit under-run data out. |
| [5] | **RXOVIEN** | **Receive FIFO Overrun Interrupt Enable Control**<br>0 = Receive FIFO overrun interrupt Disabled.<br>1 = Receive FIFO overrun interrupt Enabled. |
| [4] | **RXTOIEN** | **Slave Receive Time-out Interrupt Enable Control**<br>0 = Receive time-out interrupt Disabled.<br>1 = Receive time-out interrupt Enabled. |
| [3] | **TXTHIEN** | **Transmit FIFO Threshold Interrupt Enable Control**<br>0 = TX FIFO threshold interrupt Disabled.<br>1 = TX FIFO threshold interrupt Enabled. |
| [2] | **RXTHIEN** | **Receive FIFO Threshold Interrupt Enable Control**<br>0 = RX FIFO threshold interrupt Disabled.<br>1 = RX FIFO threshold interrupt Enabled. |
| [1] | **TXRST** | **Clear Transmit FIFO Control**<br>0 = No effect.<br>1 = Clear transmit FIFO control. The TXFULL (SPI_STATUS[17]) will be cleared to 0 and the TXEMPTY (SPI_STATUS[16]) will be set to 1. This bit will be cleared to 0 by hardware about 3 system clocks+3 SPI engine clock after it is set to 1.<br>**Note:** If there is slave receive time-out event, the TXRST will be set 1 when the SLVTORST (SPI_SSCTL[6]) is enabled. |
| [0] | **RXRST** | **Clear Receive FIFO Control**<br>0 = No effect.<br>1 = Clear receive FIFO control. The RXFULL (SPI_STATUS[9]) will be cleared to 0 and the RXEMPTY (SPI_STATUS[8]) will be set to 1. This bit will be cleared to 0 by hardware about 3 system clocks+3 SPI engine clock after it is set to 1.<br>**Note:** If there is slave receive time-out event, the RXRST will be set 1 when the SLVTORST (SPI_SSCTL[6]) is enabled. |

### SPI Status Register (SPI_STATUS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPI_STATUS** | SPIx_BA+0x14 | R/W | SPI Status Register | 0x0005_0110 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| TXCNT | | | | RXCNT | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| TXRXRST | Reserved | | | TXUFIF | TXTHIF | TXFULL | TXEMPTY |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| SPIENSTS | Reserved | | RXTOIF | RXOVIF | RXTHIF | RXFULL | RXEMPTY |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| SLVURIF | SLVBEIF | SLVTOIF | SSLINE | SSINAIF | SSACTIF | UNITIF | BUSY |

| Bits | Description | |
|---|---|---|
| [31:28] | **TXCNT** | **Transmit FIFO Data Count (Read Only)**<br>This bit field indicates the valid data count of transmit FIFO buffer. |
| [27:24] | **RXCNT** | **Receive FIFO Data Count (Read Only)**<br>This bit field indicates the valid data count of receive FIFO buffer. |
| [23] | **TXRXRST** | **FIFO CLR Status (Read Only)**<br>0 = Done the FIFO buffer clear function of TXRST (SPI_FIFOCTL[1]) or RXRST (SPI_FIFOCTL[0]).<br>1 = Doing the FIFO buffer clear function of TXRST (SPI_FIFOCTL[1])or RXRST (SPI_FIFOCTL[0]).<br>**Note:** Both the TXRST (SPI_FIFOCTL[1]), RXRST (SPI_FIFOCTL[0]), need 3 system clock+3 engine clock , the status of this bit support the user to monitor the clear function is doing or done. |
| [22:20] | **Reserved** | Reserved. |
| [19] | **TXUFIF** | **TX Underflow Interrupt Flag**<br>When the TX underflow event occurs, this bit will be set to 1, the state of data output pin depends on the setting of TXUFPOL.<br>0 = No effect.<br>1 = No data in Transmit FIFO and TX shift register when the slave selection signal is active.<br>**Note 1:** This bit will be cleared by writing 1 to it.<br>**Note 2:** If reset slave’s transmission circuit when slave selection signal is active, this flag will be set to 1 after 3 system clock cycles + 2 peripheral clock cycles since the reset operation is done. |
| [18] | **TXTHIF** | **Transmit FIFO Threshold Interrupt Status (Read Only)**<br>0 = The valid data count within the transmit FIFO buffer is larger than the setting value of TXTH (SPI_FIFOCTL[30:28]).<br>1 = The valid data count within the transmit FIFO buffer is less than or equal to the setting value of TXTH (SPI_FIFOCTL[30:28]). |

| | | |
|---|---|---|
| | | **Note:** If TXTHIEN (SPI_FIFOCTL[3]) = 1 and TXTHIF = 1, the SPI controller will generate a SPI interrupt request. |
| [17] | **TXFULL** | **Transmit FIFO Buffer Full Indicator (Read Only)**<br>0 = Transmit FIFO buffer is not full.<br>1 = Transmit FIFO buffer is full. |
| [16] | **TXEMPTY** | **Transmit FIFO Buffer Empty Indicator (Read Only)**<br>0 = Transmit FIFO buffer is not empty.<br>1 = Transmit FIFO buffer is empty. |
| [15] | **SPIENSTS** | **SPI Enable Bit Status (Read Only)**<br>0 = Indicate the transmit control bit is disabled.<br>1 = Indicate the transfer control bit is active.<br>The clock source of SPI controller logic is engine clock, it is asynchronous with the system clock. In order to make sure the function is disabled in SPI controller logic, this bit indicates the real status of SPIEN (SPI_CTL[0]) in SPI controller logic for user. |
| [14:13] | **Reserved** | Reserved. |
| [12] | **RXTOIF** | **Receive Time-out Interrupt Status**<br>0 = No receive FIFO time-out event.<br>1 = Receive FIFO buffer is not empty and no read operation on receive FIFO buffer over 64 SPI clock period in Master mode or over 576 SPI engine clock period in Slave mode. When the received FIFO buffer is read by software, the time-out status will be cleared automatically.<br>**Note:** This bit will be cleared by writing 1 to itself. |
| [11] | **RXOVIF** | **Receive FIFO Overrun Status**<br>0= No receiver FIFO overrun status.<br>1= Receive FIFO buffer is full, the follow-up data will be dropped and this bit will be set to 1.<br>**Note:** This bit will be cleared by writing 1 to itself. |
| [10] | **RXTHIF** | **Receive FIFO Threshold Interrupt Status (Read Only)**<br>0 = The valid data count within the RX FIFO buffer is smaller than or equal to the setting value of RXTH (SPI_FIFOCTL[26:24]).<br>1 = The valid data count within the receive FIFO buffer is larger than the setting value of RXTH (SPI_FIFOCTL[26:24]).<br>**Note:** If RXTHIEN (SPI_FIFOCTL[2]) = 1 and RXTHIF = 1, the SPI controller will generate a SPI interrupt request. |
| [9] | **RXFULL** | **Receive FIFO Buffer Empty Indicator (Read Only)**<br>0 = Receive FIFO buffer is not empty.<br>1 = Receive FIFO buffer is empty. |
| [8] | **RXEMPTY** | **Receive FIFO Buffer Empty Indicator (Read Only)**<br>0 = Receive FIFO buffer is not empty.<br>1 = Receive FIFO buffer is empty. |
| [7] | **SLVURIF** | **Slave Mode Error 1 Interrupt Status (Read Only)**<br>In Slave mode, transmit under-run occurs when the slave select line goes to inactive state.<br>0 = No Slave mode error 1 event.<br>1 = Slave mode error 1 occurs. |
| [6] | **SLVBEIF** | **Slave Mode Error 0 Interrupt Status (Read Only)**<br>In Slave mode, there is bit counter mismatch with DWIDTH (SPI_CTL[12:8]) when the slave select line goes to inactive state. |

| | | |
|---|---|---|
| | | 0 = No Slave mode error 0 event.<br>1 = Slave mode error 0 occurs.<br>**Note:** If the slave select active but there is no any serial clock input, the SLVBEIF (SPI_STATUS[6]) also active when the slave select goes to inactive state. |
| [5] | **SLVTOIF** | **Slave Time-out Interrupt Status (Read Only)**<br>When the Slave Select is active and the value of SLVTOCNT (SPI_SSCTL[31:16]) is not 0 and the serial clock input, the slave time-out counter in SPI controller logic will be start. When the value of time-out counter greater or equal than the value of SLVTOCNT (SPI_SSCTL[31:16]) during before one transaction done, the slave time-out interrupt event will active.<br>0 = Slave time-out is not active.<br>1 = Slave time-out is active.<br>**Note:** If the DWIDTH (SPI_CTL[12:8]) is set 0x10, one transaction is equal 16 bits serial clock period. |
| [4] | **SSLINE** | **Slave Select Line Bus Status (Read Only)**<br>0 = Indicates the slave select line bus status is 0.<br>1 = Indicates the slave select line bus status is 1.<br>**Note:** If SSACTPOL (SPI_SSCTL[2]) is set 0, and the SSLINE is 1, the SPI slave select is in inactive status. |
| [3] | **SSINAIF** | **Slave Select Inactive Interrupt Status**<br>0 = Slave select inactive interrupt is cleared or did not occur.<br>1 = Slave select inactive interrupt event has occurred.<br>**Note:** This bit will be cleared by writing 1 to itself. |
| [2] | **SSACTIF** | **Slave Select Active Interrupt Status**<br>0 = Slave select active interrupt is cleared or did not occur.<br>1 = Slave select active interrupt event has occurred.<br>**Note:** This bit will be cleared by writing 1 to itself. |
| [1] | **UNITIF** | **Unit Transfer Interrupt Status**<br>0 = No transaction has been finished since this bit was cleared to 0.<br>1 = SPI controller has finished one unit transfer.<br>**Note:** This bit will be cleared by writing 1 to itself. |
| [0] | **BUSY** | **SPI Unit Bus Status (Read Only)**<br>0 = No transaction in the SPI bus.<br>1 = SPI controller unit in busy state.<br>The following lists the bus busy conditions:<br>a. SPIEN (SPI_CTL[0]) = 1 and the TXEMPTY (SPI_STATUS[16]) = 0.<br>b. For SPI Master, the TXEMPTY (SPI_STATUS[16]) = 1 but the current transaction is not finished yet.<br>c. For SPI Slave receive mode, the SPIEN (SPI_CTL[0]) = 1 and there is serial clock input into the SPI core logic when slave select is active.<br>d. For SPI Slave transmit mode, the SPIEN (SPI_CTL[0]) = 1 and the transmit buffer is not empty in SPI core logic event if the slave select is inactive. |

### SPI Data Transmit Register (SPI_TX)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPI_TX** | SPIx_BA+0x20 | W | SPI Data Transmit Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **TX** | **Data Transmit Register**<br>The data transmit registers pass through the transmitted data into the 8-level transmit FIFO buffer. The number of valid bits depends on the setting of transmit bit width field of the SPI_CTL register.<br>For example, if DWIDTH (SPI_CTL[12:8]) is set to 0x08, the bits TX[7:0] will be transmitted. If DWIDTH (SPI_CTL[12:8]) is set to 0x00, the SPI controller will perform a 32-bit transfer. |

### SPI Data Receive Register (SPI_RX)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPI_RX** | SPIx_BA+0x30 | R | SPI Data Receive Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **RX** | **Data Receive Register**<br>There is 8-level FIFO buffer in this controller. The data receive register holds the earliest datum received from SPI data input pin. If the RXEMPTY (SPI_STATUS[8]) is not set to 1, the receive FIFO buffer can be accessed through software by reading this register. This is a read-only register. |

## 6.13 SPI Memory Interface Controller (SPIM)

### 6.13.1 Overview

The SPI Memory Interface Controller performs a serial-to-parallel conversion on data received from the peripheral, and a parallel-to-serial conversion on data received from CPU. This controller can drive up to 2 external peripherals (embedded SPI Flash or external SPI Flash) and act as a SPI master. It can generate an interrupt signal when data transfer is finished and can be cleared by writing 1 to the interrupt flag. The active level of device/slave select signal can be chosen to low active or high active, which depends on the peripheral. Writing a divisor into the SPIM_CTL1 register can program the frequency of serial clock output to the peripheral. This controller contains four 32-bit transmit/receive buffers, and can provide 1 to 4 burst mode operation. The number of bits in each transaction can be 8, 16, 24, or 32; data can be transmitted/received up to four successive transactions in one transfer.

### 6.13.2 Features

- Supports SPI master mode
- Supports DMA mode (DMA Write and DMA Read), Direct Memory Map (DMM) mode, and I/O mode
- 8-, 16-, 24-, and 32-bit length of transaction
- Supports standard (1-bit), dual (2-bit), and quad (4-bit) I/O transfer mode
- Provides burst mode operation, which can transmit/receive data up to four successive transactions in one transfer
- Two slave/device select lines (embedded SPI Flash or external SPI Flash)
- Fully static synchronous design with one clock domain

### 6.13.3 Block Diagram

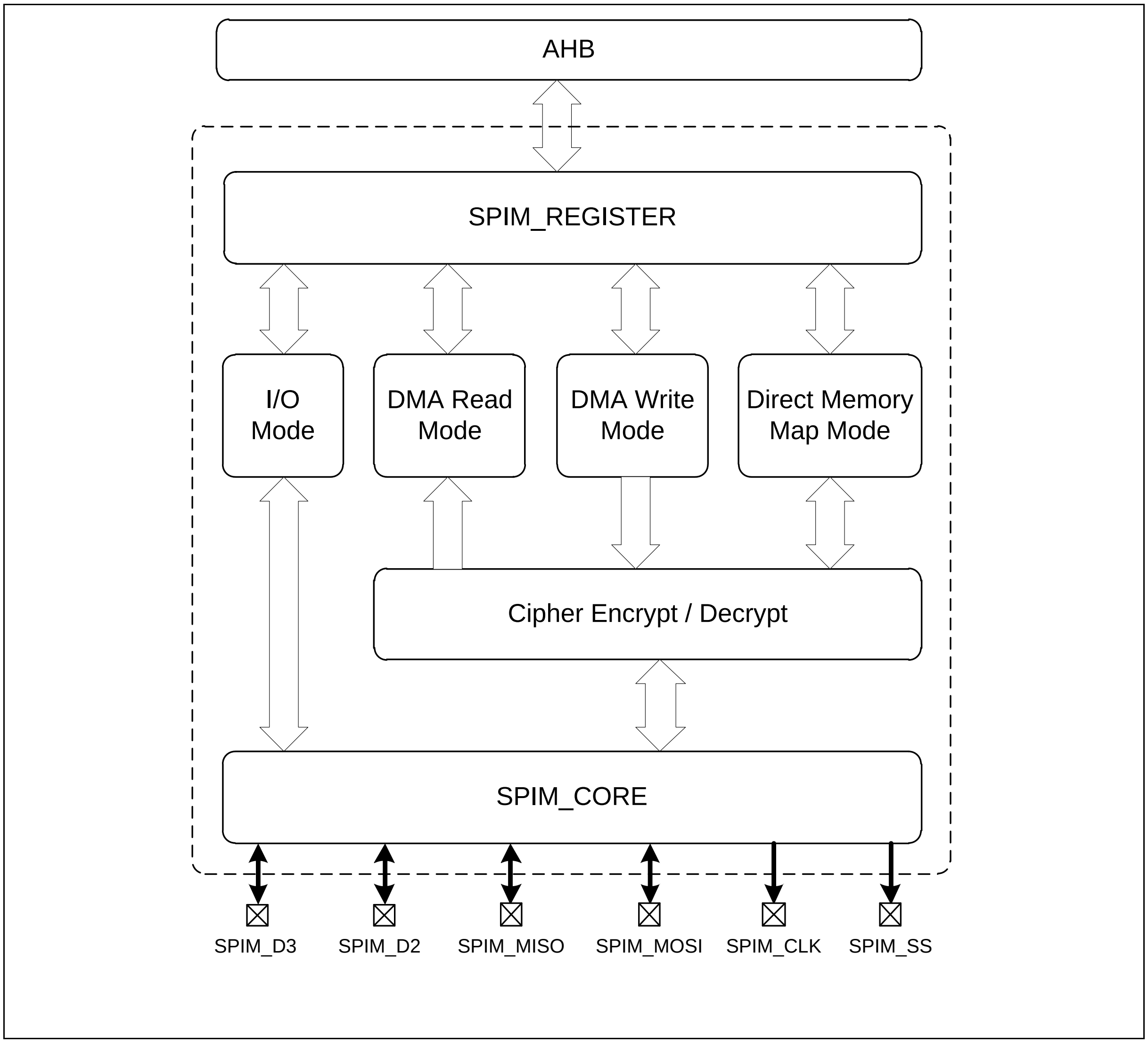


Figure 6.13-1 SPIM Block Diagram

### 6.13.4 Functional Description

#### *6.13.4.1 SPIM Timing Diagram*

The timing diagram of SPI transaction is shown as follows:

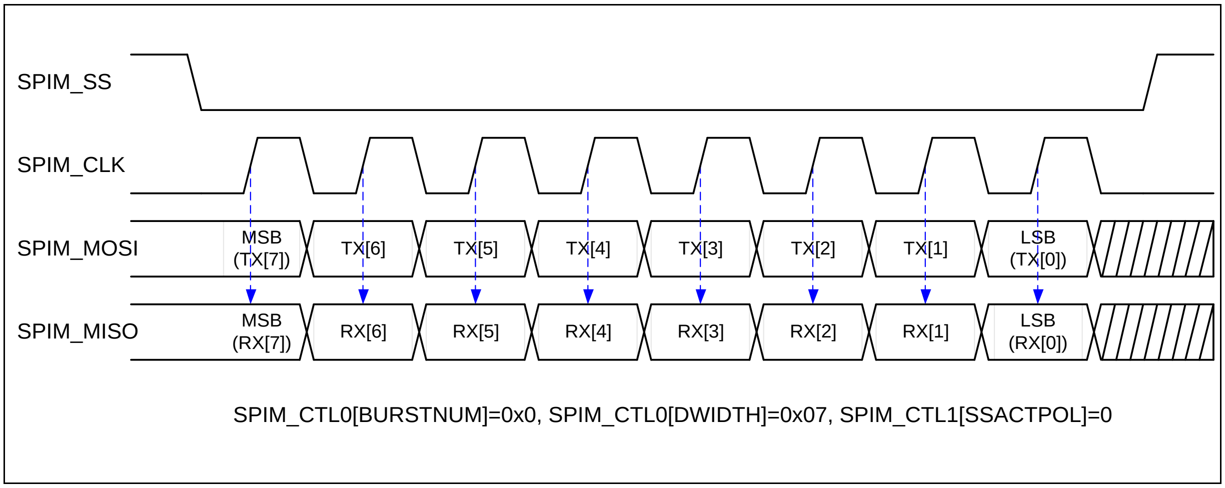


Figure 6.13-2 SPIM Timing Diagram

#### *6.13.4.2 SPIM Programming Example without DMA (I/O Mode)*

To access a device with the following requirements:

- Data is transferred with the MSB first.
- Only one byte is transmitted/received in a transfer.
- Chip select signal is active low.

You should do the following actions basically (refer to the specification of the device for the detailed steps):

1. Write a divisor into DIVIDER (SPIM_CTL1[31:16]) to determine the frequency of serial clock.
2. Set SSACTPOL (SPIM_CTL1[5]) to 0 and IFSEL (SPIM_CTL1[7:6]) to activate the device you want to access.
3. When transmitting (writing) data to device:
   - QDIODIR (SPIM_CTL0[15]) = 1
   - BURSTNUM (SPIM_CTL0[14:13]) = 0x0
   - DWIDTH (SPIM_CTL0[12:8]) = 0x07
   - Write the data you want to transmit into SPIM_TX0[7:0]
4. When receiving (reading) data from device:
   - QDIODIR (SPIM_CTL0[15]) = 0
   - BURSTNUM (SPIM_CTL0[14:13]) = 0x0
   - DWIDTH (SPIM_CTL0[12:8]) = 0x07
5. Set SPIMEN (SPIM_CTL1[0]) to 1 to start the transfer.
6. Wait for interrupt or poll the SPIMEN (SPIM_CTL1[0]) until it turns to 0.
7. When receiving (reading) data from device:
   - Read out the received data from the SPIM_RX0[7:0] register.

#### *6.13.4.3 SPIM Programming Example with DMA*

If users want to access a device with DMA function, three additional registers need to be configured, including SPIM_DMATBCNT, SPIM_SRAMADDR and SPIM_FADDR. The DMA function can be used to support loading boot code (reading data from the peripheral into system memory) or reading data from system memory and storing the data in the peripheral. Users must define the source, length and destination, and then hardware will automatically move the desired length of code to the specific target address.

#### *6.13.4.4 Code Boot Process (DMA Read Mode)*

Step 1: Check device ID by using I/O mode to confirm the device has connected.

Step 2:

1. Set the target memory address into the SPIM_SRAMADDR register.
2. Set the boot code length into the SPIM_DMATBCNT register.
3. Set the SPI Flash start address in the SPIM_FADDR register.
4. Set OPMODE (SPIM_CTL0[23:22]) with the DMA Read mode.
5. Set CMDCODE (SPIM_CTL0[31:24]) with the 0x03 command code.
6. Set SPIMEN (SPIM_CTL1[0]) to start. Then SPIM will move the code block from SPIM_FADDR to SPIM_SRAMADDR with the transfer length set in SPIM_DMATBCNT.
7. Wait for interrupt or poll the SPIMEN (SPIM_CTL1[0]) bit until it turns to 0.

If the used SPI Flash supports other associated read commands, users can also use the following read command code.

1. Fast read (0Bh), set CMDCODE (SPIM_CTL0[31:24])with 0x0B.
2. Fast dual read (3Bh), set CMDCODE (SPIM_CTL0[31:24]) with 0x3B.
3. Quad read (EBh), set CMDCODE (SPIM_CTL0[31:24]) with 0xEB.

#### *6.13.4.5 Move Data from System Memory to Peripheral (DMA Write Mode)*

Step 1: Erase the SPI Flash (using I/O mode) before programming it.

Step 2:

1. Send Write Enable command to SPI Flash (using I/O mode).
2. Set the source memory address in the SPIM_SRAMADDR register.
3. Set the transfer count into the SPIM_DMATBCNT register.
4. Set the SPI Flash start address in the SPIM_FADDR register.
5. Set OPMODE (SPIM_CTL0[23:22]) with the DMA Write mode.
6. Set CMDCODE (SPIM_CTL0[31:24]) with the 0x02 command code.
7. Set SPIMEN (SPIM_CTL1[0]) to start.
8. Wait for interrupt or poll the SPIMEN (SPIM_CTL1[0]) until it turns to 0.

If the used SPI Flash supports other associated write commands, users can also use the following command code. (User needs to refer to SPI Flash SPEC to select suitable Quad write mode command.)

1. Quad write, set CMDCODE (SPIM_CTL0[31:24]) to 0x32 for TYPE_1 program command

(refer to Figure 6.13-3).

2. Quad write, set CMDCODE (SPIM_CTL0[31:24]) to 0x38 for TYPE_2 program command (refer to Figure 6.13-4).
3. Quad write, set CMDCODE (SPIM_CTL0[31:24]) to 0x40 for TYPE_3 program command (refer to Figure 6.13-5).

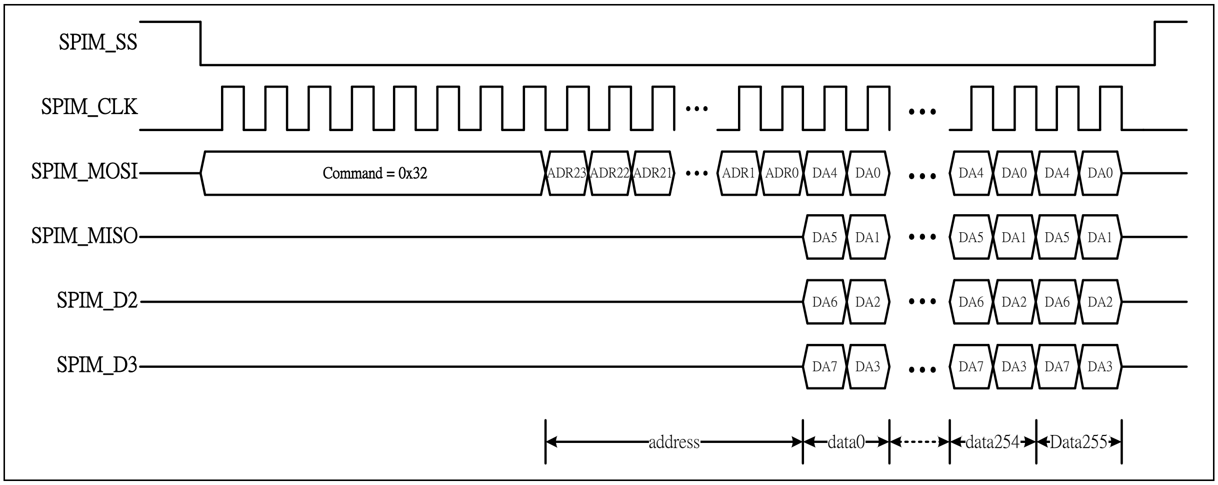


Figure 6.13-3 TYPE_1 Program Command Flow with Quad Mode Write.

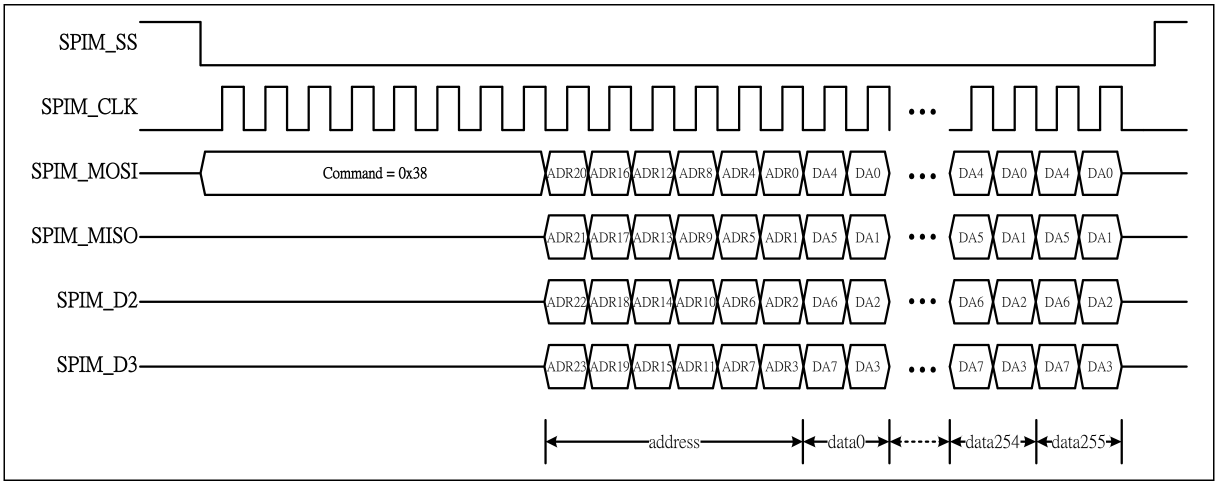


Figure 6.13-4 TYPE_2 Program Command Flow with Quad Mode Write.

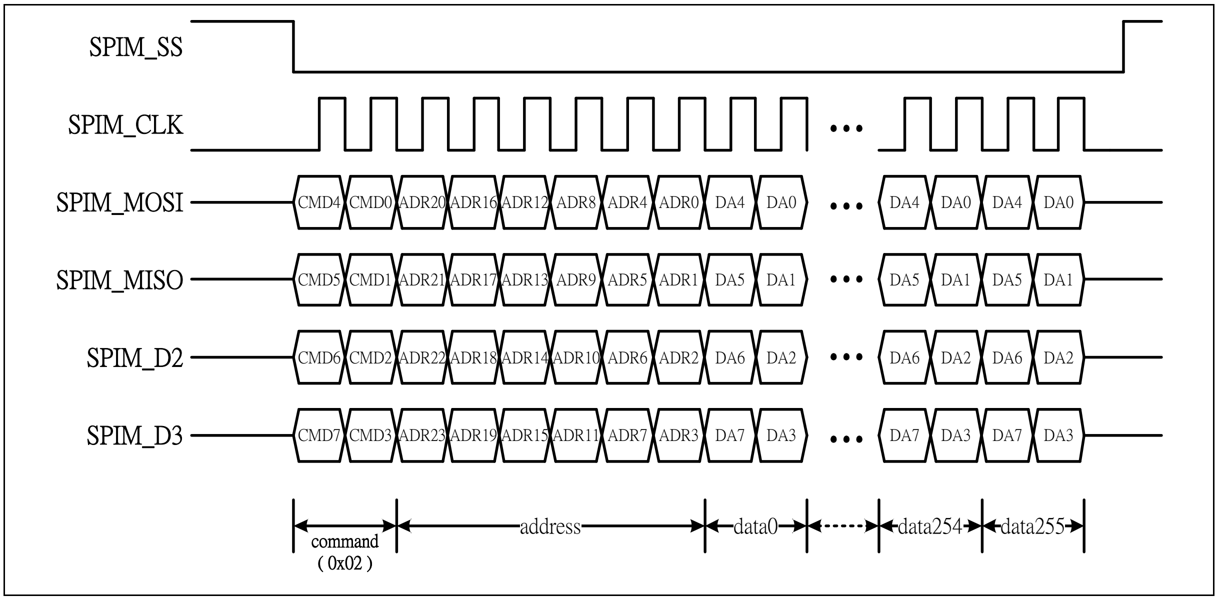

SPIM_SS
SPIM_CLK
SPIM_MOSI
CMD4 CMD0 ADR20 ADR16 ADR12 ADR8 ADR4 ADR0 DA4 DA0 ••• DA4 DA0 DA4 DA0
SPIM_MISO
CMD5 CMD1 ADR21 ADR17 ADR13 ADR9 ADR5 ADR1 DA5 DA1 ••• DA5 DA1 DA5 DA1
SPIM_D2
CMD6 CMD2 ADR22 ADR18 ADR14 ADR10 ADR6 ADR2 DA6 DA2 ••• DA6 DA2 DA6 DA2
SPIM_D3
CMD7 CMD3 ADR23 ADR19 ADR15 ADR11 ADR7 ADR3 DA7 DA3 ••• DA7 DA3 DA7 DA3
command
( 0x02 )
address
data0
data254
data255

Figure 6.13-5 TYPE_3 Program Command Flow with Quad Mode Write.

### *6.13.4.6 Direct Memory Mapping Mode*

SPI Flash can be regarded as a ROM module in Direct Memory Mapping mode. The controller will convert the AHB cycle into SPI Flash without CPU setting related SPI Flash command. The only setting CPU needs to do is to set CMDCODE (SPIM_CTL0[31:24]) with the 0x03, 0x0B, 0x3B, or 0xEB command code. Then users can access SPI Flash as a ROM module.

In Direct Memory Mapping mode, it will pre-fetch 4-word Flash data after a direct memory mapping access. If users want to modify SPIM hardware registers after the direct memory mapping access, wait for at least 250 peripheral cycles (SPIM bus cycles).

### 6.13.5 Register Map

**R**: read only, **W**: write only, **R/W**: both read and write, **C**: Only value 0 can be written

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPIM Base Address:**<br>**SPIM_BA = 0x4000_7000** | | | | |
| **SPIM_CTL0** | SPIM_BA+0x00 | R/W | Control and Status Register 0 | 0x00C0_0002 |
| **SPIM_CTL1** | SPIM_BA+0x04 | R/W | Control Register 1 | 0x0000_0010 |
| **SPIM_VALIDCTL** | SPIM_BA+0x08 | R/W | Validation Check Register | 0x0000_0000 |
| **SPIM_RXCLKDLY** | SPIM_BA+0x0C | R/W | Rx Clock Delay Control Register | 0x0000_0000 |
| **SPIM_RX0** | SPIM_BA+0x10 | R | Data Receive Register 0 | 0x0000_0000 |
| **SPIM_RX1** | SPIM_BA+0x14 | R | Data Receive Register 1 | 0x0000_0000 |
| **SPIM_RX2** | SPIM_BA+0x18 | R | Data Receive Register 2 | 0x0000_0000 |
| **SPIM_RX3** | SPIM_BA+0x1C | R | Data Receive Register 3 | 0x0000_0000 |
| **SPIM_TX0** | SPIM_BA+0x20 | R/W | Data Transmit Register 0 | 0x0000_0000 |
| **SPIM_TX1** | SPIM_BA+0x24 | R/W | Data Transmit Register 1 | 0x0000_0000 |
| **SPIM_TX2** | SPIM_BA+0x28 | R/W | Data Transmit Register 2 | 0x0000_0000 |
| **SPIM_TX3** | SPIM_BA+0x2C | R/W | Data Transmit Register 3 | 0x0000_0000 |
| **SPIM_SRAMADDR** | SPIM_BA+0x30 | R/W | SRAM Memory Address Register | 0x0000_0000 |
| **SPIM_DMATBCNT** | SPIM_BA+0x34 | R/W | DMA Transfer Byte Count Register | 0x0000_0000 |
| **SPIM_FADDR** | SPIM_BA+0x38 | R/W | SPI Flash Address Register | 0x0000_0000 |

**Note:** If software wants to write to any register of the peripheral, the SPIMEN bit of the SPIM_CTL1 register should be the status low.

### 6.13.6 Register Description

**Control and Status Register 0 (SPIM_CTL0)**

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPIM_CTL0** | SPIM_BA+0x00 | R/W | Control and Status Register 0 | 0x00C0_0002 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| CMDCODE | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| OPMODE | | BITMODE | | SUSPITV | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| QDIODIR | BURSTNUM | | DWIDTH | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| IF | IEN | B4ADDREN | Reserved | | BALEN | Reserved | CIPHOFF |

| Bits | Description | |
|---|---|---|
| [31:24] | **CMDCODE** | **Page Program Command Code**<br>0x02 = Page program (Used for DMA Write mode).<br>0x32 = Quad page program with TYPE_1 program flow (Used for DMA Write mode).<br>0x38 = Quad page program with TYPE_2 program flow (Used for DMA Write mode).<br>0x40 = Quad page program with TYPE_3 program flow (Used for DMA Write mode).<br>The Others = Reserved.<br>**Read Command Code**<br>0x03 = Standard read (Used for DMA Read/DMM mode).<br>0x0B = Fast read (Used for DMA Read/DMM mode).<br>0x3B = Fast dual read (Used for DMA Read/DMM mode).<br>0xEB = Quad read (Used for DMA Read/DMM mode).<br>The Others = Reserved.<br>**Note1:** Quad mode of SPI Flash must be enabled first by I/O mode before using quad page program/quad read commands.<br>**Note2:** See support list for SPI Flash which support these command codes.<br>**Note3:** For TYPE_1, TYPE_2, and TYPE_3 program flows, refer to Figure 6.13-3, Figure 6.13-4, and Figure 6.13-5. |
| [23:22] | **OPMODE** | **SPI Function Operation Mode**<br>00 = I/O mode.<br>01 = DMA Write mode.<br>10 = DMA Read mode.<br>11 = Direct Memory Map (DMM) mode (Default).<br>**Note:** In DMA Write mode, hardware will send just one page program command per operation. Users must take care of cross-page cases. |
| [21:20] | **BITMODE** | **SPI Interface Bit Mode**<br>00 = Standard mode. |

| | | |
|---|---|---|
| | | 01 = Dual mode.<br>10 = Quad mode.<br>11 = Reserved.<br>**Note:** Only used for I/O mode. |
| [19:16] | **SUSPITV** | **Suspend Interval**<br>These four bits provide the configuration of suspend interval between two successive transmit/receive transactions in a transfer. The default value is 0x00. When BURSTNUM = 00, setting this field has no effect on transfer. The desired interval is obtained according to the following equation (from the last falling edge of current sclk to the first rising edge of next sclk):<br>(SUSPITV+2)*period of SCLK<br>0x0 = 2 SCLK clock cycles.<br>0x1 = 3 SCLK clock cycles.<br>……<br>0xE = 16 SCLK clock cycles.<br>0xF = 17 SCLK clock cycles.<br>**Note:** Only used for I/O mode. |
| [15] | **QDIODIR** | **SPI Interface Direction Select for Quad/Dual Mode**<br>0 = Interface signals are input.<br>1 = Interface signals are output.<br>**Note:** Only used for I/O mode. |
| [14:13] | **BURSTNUM** | **Transmit/Receive Burst Number**<br>This field specifies how many transmit/receive transactions should be executed continuously in one transfer.<br>00 = Only one transmit/receive transaction will be executed in one transfer.<br>01 = Two successive transmit/receive transactions will be executed in one transfer.<br>10 = Three successive transmit/receive transactions will be executed in one transfer.<br>11 = Four successive transmit/receive transactions will be executed in one transfer.<br>**Note:** Only used for I/O Mode. |
| [12:8] | **DWIDTH** | **Transmit/Receive Bit Length**<br>This field specifies how many bits are transmitted/received in one transmit/receive transaction.<br>0x7 = 8 bits.<br>0xF = 16 bits.<br>0x17 = 24 bits.<br>0x1F = 32 bits.<br>Others = Incorrect transfer result.<br>**Note1:** Only used for I/O mode.<br>**Note2:** Only 8-, 16-, 24-, and 32-bit are allowed. Other bit length will result in incorrect transfer. |
| [7] | **IF** | **Interrupt Flag**<br>Write Operation:<br>0 = No effect.<br>1 = Write 1 to clear.<br>Read Operation:<br>0 = The transfer has not finished yet.<br>1 = The transfer has done. |

NUC505 SERIES TECHNICAL REFERENCE MANUAL

| | | |
|---|---|---|
| [6] | **IEN** | **Interrupt Enable Control**<br>0 = SPIM Interrupt Disabled.<br>1 = SPIM Interrupt Enabled. |
| [5] | **B4ADDREN** | **4-byte Address Mode Enable Control**<br>0 = 4-byte address mode Disabled.<br>1 = 4-byte address mode Enabled.<br>**Note:** Used for DMA Write/DMA Read/DMM mode. |
| [4:3] | **Reserved** | Reserved. |
| [2] | **BALEN** | **Balance the AHB Control Time Between Cipher Enable and Disable Control**<br>When cipher is enabled, the AHB control signal will delay some time caused by the encoding or decoding calculation. Therefore, if set BALEN to 1, it will make the AHB signal processing time with cipher disabled be equal to that with cipher enabled.<br>**Note:** Only useful when cipher is disabled. |
| [1] | **Reserved** | Reserved. |
| [0] | **CIPHOFF** | **Cipher Disable Control**<br>0 = Cipher function Enabled.<br>1 = Cipher function Disabled.<br>**Note1:** Cipher function only can be disabled. If user wants to enable cipher function after disabled processing, the chip must be reset again.<br>**Note2:** If there is not any key in the chip, the cipher will be disabled automatically.<br>**Note3:** In ICE mode, the cipher will be disabled automatically. |

### Control Register 1 (SPIM_CTL1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPIM_CTL1** | SPIM_BA+0x04 | R/W | Control Register 1 | 0x0000_0010 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **DIVIDER** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **DIVIDER** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | **IDLECNT** | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **IFSEL** | | **SSACTPOL** | **SS** | **Reserved** | | | **SPIMEN** |

| Bits | Description | |
|---|---|---|
| [31:16] | **DIVIDER** | **Clock Divider Register**<br>The value in this field is the frequency divider of the system clock to generate the serial clock on the output SPIM_CLK pin. The desired frequency is obtained according to the following equation:<br>$f_{SPIM\_CLK} = \frac{f_{SYS\_CLK}}{(DIVIDER)*2}$<br>**Note:** When set DIVIDER to zero, the frequency of SPIM_CLK will be equal to the frequency of SYS_CLK. |
| [15:12] | **Reserved** | Reserved. |
| [11:8] | **IDLECNT** | **Idle Interval**<br>In DMM mode, IDLECNT is set to control the minimum idle time between two SPI Flash accesses.<br>**Note:** Only used for DMM mode. |
| [7:6] | **IFSEL** | **Device/Slave Interface Select**<br>00 = SPI Interface from GPIO.<br>01 = SPI Interface from MCP.<br>10 = SPI Interface from MCP64.<br>11 = Reserved.<br>**Note:** MCP and MCP64 only can be referenced by MCP SPI Flash pad location. |
| [5] | **SSACTPOL** | **Slave Select Active Level**<br>It defines the active level of device/slave select signal (SPIM_SS).<br>0 = The SPIM_SS slave select signal is Active Low.<br>1 = The SPIM_SS slave select signal is Active High. |
| [4] | **SS** | **Slave Select Active Enable Control**<br>0 = SPIM_SS is in active level. |

| | | |
|---|---|---|
| | | 1 = SPIM_SS is in inactive level.<br>**Note:** This interface can only drive one device/slave at a given time. Therefore, the slave selects of the selected device must be set to its active level before starting any read or write transfer. |
| [3:1] | **Reserved** | Reserved. |
| [0] | **SPIMEN** | **Go and Busy Status**<br>Write Operation:<br>0 = No effect.<br>1 = Start the transfer. This bit remains set during the transfer and is automatically cleared after transfer finished.<br>Read Operation:<br>0 = The transfer has done.<br>1 = The transfer has not finished yet.<br>**Note:** All registers should be set before writing 1 to the SPIMEN bit. When a transfer is in progress, you should not write to any register of this peripheral. |

### Validation Check Register (SPIM_VALIDCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPIM_VALIDCTL** | SPIM_BA+0x08 | R/W | Validation Check Register | 0x0000_0000 |

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

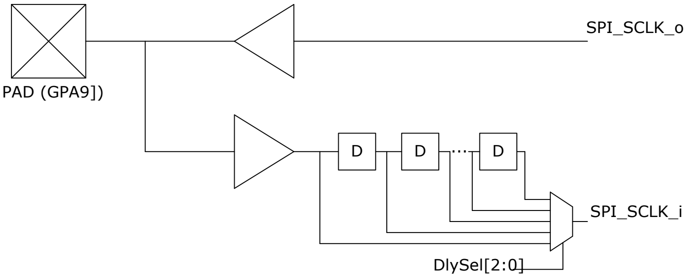
| Bits | Description | |
|---|---|---|
| [31:17] | **Reserved** | Reserved. |
| [16] | **VALIDEN** | **Validation Enable Bit**<br>Setting this bit to enable the validation function. The function can check whether the code in SPI Flash is valid or not.<br>0=>1 (Rising Edge) = Enable the validation and clear the VALIDSTS bit.<br>1=>0 (Falling Edge) = Disable the validation and update the VALIDSTS bit. |
| [15:1] | **Reserved** | Reserved. |
| [0] | **VALIDSTS** | **Validation Status Bit**<br>This bit will be updated when the VALIDEN bit changes.<br>0 = Code in SPI Flash is not valid.<br>1 = Code in SPI Flash is valid.<br>**Note:** Write 0 to clear it to 0. |

### Rx Clock Delay Control Register (SPIM_RXCLKDLY)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPIM_RXCLKDLY** | SPIM_BA+0x0C | R/W | Rx Clock Delay Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:29] | **DLYSEL** | **Rx Sample Clock Source Delay Chain Select**<br>000 = Not Delay.<br>001 = Select sample clock through 2 Delay Cell.<br>010 = Select sample clock through 4 Delay Cell.<br>011 = Select sample clock through 6 Delay Cell.<br>…<br>111 = Select sample clock through 14 Delay Cell. |
| [28:0] | **Reserved** | Reserved. |



The function diagram can be shown as above. The Rx data (from SPI Flash) may have some delay. Therefore, we use DLYSEL[2:0] bit field to adjust the sampling clock (SPI_SCLK_i) to latch the correct data.

### Data Receive Register 0~3 (SPIM_RX0~3)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPIM_RX0** | SPIM_BA+0x10 | R | Data Receive Register 0 | 0x0000_0000 |
| **SPIM_RX1** | SPIM_BA+0x14 | R | Data Receive Register 1 | 0x0000_0000 |
| **SPIM_RX2** | SPIM_BA+0x18 | R | Data Receive Register 2 | 0x0000_0000 |
| **SPIM_RX3** | SPIM_BA+0x1C | R | Data Receive Register 3 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **RX** | **Data Receive Register**<br>The Data Receive Registers hold the received data of the last executed transfer.<br>Number of valid RX registers is specified in SPIM_CTL0[BURSTNUM]. If BURSTNUM > 0, received data are held in the most significant RX register first.<br>Number of valid-bit is specified in SPIM_CTL0[DWIDTH]. If DWIDTH is 16, 24, or 32, received data are held in the least significant byte of RX register first.<br>In a byte, received data are held in the most significant bit of RX register first.<br>**Example 1:** If SPIM_CTL0[BURSTNUM] = 0x3 and SPIM_CTL1[DWIDTH] = 0x17, received data will be held in the order SPIM_RX3[23:0], SPIM_RX2[23:0], SPIM_RX1[23:0], SPIM_RX0[23:0].<br>**Example 2:** If SPIM_CTL0[BURSTNUM = 0x0 and SPIM_CTL0[DWIDTH] = 0x17, received data will be held in the order SPIM_RX0[7:0], SPIM_RX0[15:8], SPIM_RX0[23:16].<br>**Example 3:** If SPIM_CTL0[BURSTNUM = 0x0 and SPIM_CTL0[DWIDTH] = 0x07, received data will be held in the order SPIM_RX0[7], SPIM_RX0[6], …., SPIM_RX0[0]. |

### Data Transmit Register 0~3 (SPIM_TX0~3)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPIM_TX0** | SPIM_BA+0x20 | R/W | Data Transmit Register 0 | 0x0000_0000 |
| **SPIM_TX1** | SPIM_BA+0x24 | R/W | Data Transmit Register 1 | 0x0000_0000 |
| **SPIM_TX2** | SPIM_BA+0x28 | R/W | Data Transmit Register 2 | 0x0000_0000 |
| **SPIM_TX3** | SPIM_BA+0x2C | R/W | Data Transmit Register 3 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **TX** | **Data Transmit Register**<br>The Data Transmit Registers hold the data to be transmitted in next transfer.<br>Number of valid TX registers is specified in SPIM_CTL0[BURSTNUM]. If BURSTNUM > 0, data are transmitted in the most significant TX register first.<br>Number of valid-bit is specified in SPIM_CTL0[DWIDTH]. If DWIDTH is 16, 24, or 32, data are transmitted in the least significant byte of TX register first.<br>In a byte, data are transmitted in the most significant bit of TX register first.<br>**Example 1:** If SPIM_CTL0[BURSTNUM] = 0x3 and SPIM_CTL1[DWIDTH] = 0x17, data will be transmitted in the order SPIM_TX3[23:0], SPIM_TX2[23:0], SPIM_TX1[23:0], SPIM_TX0[23:0] in next transfer.<br>**Example 2:** If SPIM_CTL0[BURSTNUM] = 0x0 and SPIM_CTL0[DWIDTH] = 0x17, data will be transmitted in the order SPIM_TX0[7:0], SPIM_TX0[15:8], SPIM_TX0[23:16] in next transfer.<br>**Example 3:** If SPIM_CTL0[BURSTNUM] = 0x0 and SPIM_CTL0[DWIDTH] = 0x07, data will be transmitted in the order SPIM_TX0[7], SPIM_TX0[6], …., SPIM_TX0[0] in next transfer. |

### SRAM Memory Address Register (SPIM_SRAMADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPIM_SRAMADDR** | SPIM_BA+0x30 | R/W | SRAM Memory Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ADDR** | **SRAM Memory Address**<br>For DMA Read mode, this is the destination address for DMA transfer.<br>For DMA Write mode, this is the source address for DMA transfer.<br>**Note:** This address must be word-aligned. |

### DMA Transfer Byte Count Register (SPIM_DMATBCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPIM_DMATBCNT** | SPIM_BA+0x34 | R/W | DMA Transfer Byte Count Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **CNT** | **DMA Transfer Byte Count Register**<br>It indicates the transfer length for DMA process.<br>**Note:** The unit for counting is byte.<br>**Note2:** The number must be the multiple of 4. |

### SPI Flash Address Register (SPIM_FADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SPIM_FADDR** | SPIM_BA+0x38 | R/W | SPI Flash Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ADDR** | **SPI Flash Address Register**<br>For DMA Read mode, this is the source address for DMA transfer.<br>For DMA Write mode, this is the destination address for DMA transfer.<br>**Note:** This address must be word-aligned. |

NUC505 SERIES TECHNICAL REFERENCE MANUAL

## 6.14 $I^2S$ Controller with Internal Audio CODEC ($I^2S$)

### 6.14.1 Overview

The $I^2S$ controller consists of $I^2S$ protocol interface to internal audio CODEC and supports to use external audio CODEC. The $I^2S$ controller includes two 16 words FIFO for transfer path and receiver path respectively and is capable of handling 8, 16, 24, or 32 bits word sizes sample.

The structure of internal audio CODEC is a delta-sigma 24-bit CODEC with microphone input, audio line-in input, and headphone output.

### 6.14.2 Features

**$I^2S$ Controller**

- Supports Master mode and Slave mode
- Capable of handling 8-, 16-, 24- and 32-bit word sizes sample
- Supports Mono and Stereo audio data
- Supports $I^2S$ and most significant bit (MSB) justified data format
- Supports PCM-A and PCM-B data format
- Provides two 16 words FIFO, one for transmitting and the other for receiving
- Generates interrupt requests when FIFO levels cross a programmable boundary
- Supports TX DMA function for transmitting and RX DMA function receiving
- Supports RX Data Power Measurement
- Supports connecting to external audio CODEC

**Internal CODEC**

- Supports mono microphone input and stereo audio line-in input
- Supports stereo headphone output
- Supports stereo and mono mode
- Features of ADC
  - Total-Harmonic-Distortion with Noise (THD+N): -80 dB
  - Dynamic-Range (DR) and Signal-to-Noise ratio (SNR): 90 dB (A-Weighted)
- Features of DAC (headphone out with 32Ω loading)
  - Total-Harmonic-Distortion with Noise (THD+N): -60 dB
  - Dynamic-Range (DR) and Signal-to-Noise ratio (SNR): 93 dB (A-Weighted)
- Supports sampling rate with 8 kHz, 11.025 kHz, 12 kHz, 16 kHz, 22.05 kHz, 24 kHz, 32 kHz, 44.1 kHz, 48 kHz, and 96 kHz

### 6.14.3 Block Diagram

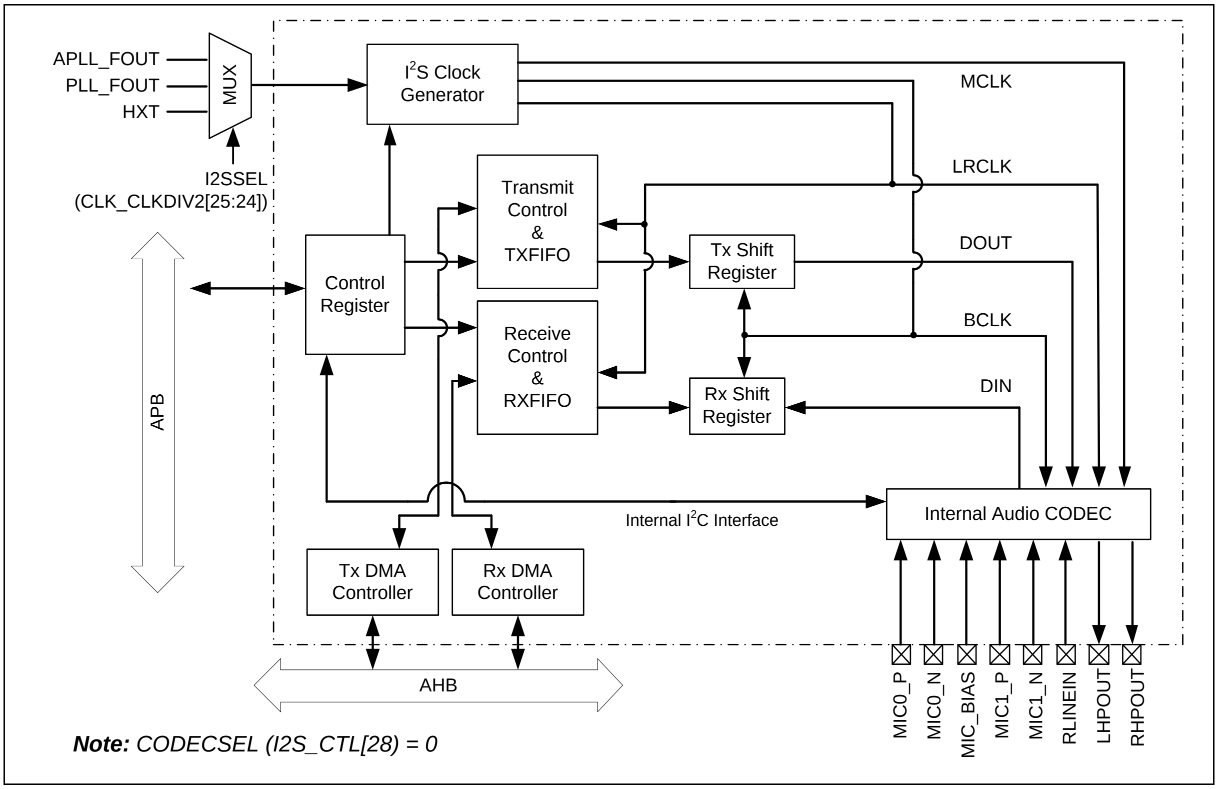


Figure 6.14-1 I²S Controller with Internal Audio CODEC Block Diagram

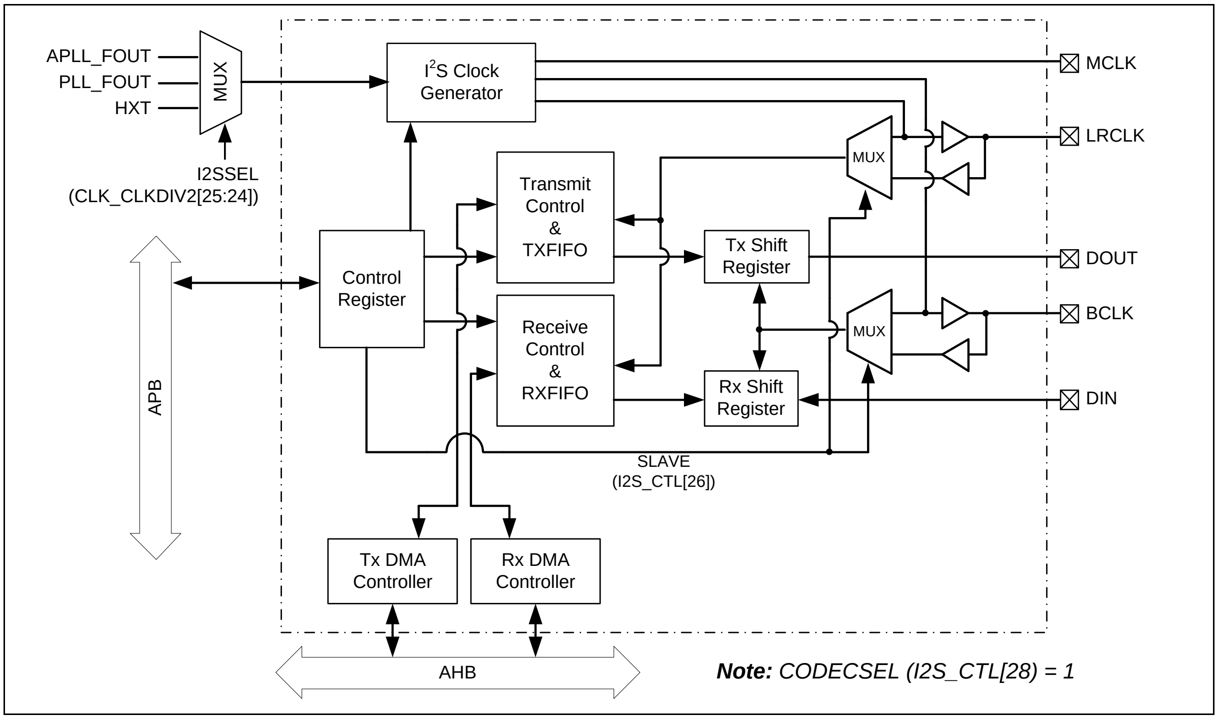


Figure 6.14-2 I²S Controller with External Audio CODEC Block Diagram

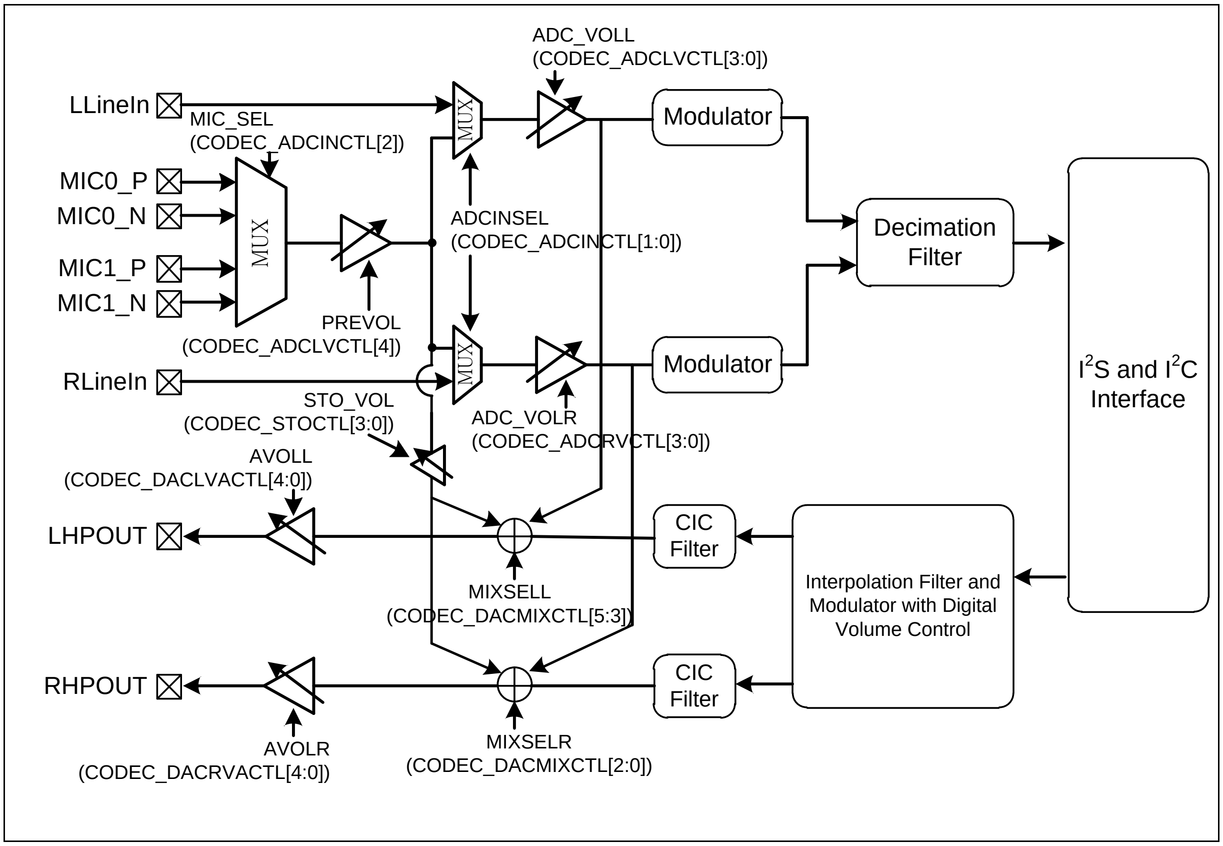


Figure 6.14-3 Internal CODEC Block Diagram

### 6.14.4 I$^2$S Operation Timing Diagram Description

The I$^2$S controller supports MSB justified and I$^2$S data format. The LRCLK signal indicates which audio channel is in transferring. The bit count of an audio channel is determined by WDWIDTH (I2S_CTL[5:4]). The transfer sequence is always first from the most significance bit, MSB. Data are read on rising clock edge and are driven on falling clock edge. In I$^2$S data format, the MSB is sent and latched on the second clock of an audio channel. In MSB justified data format, the MSB is sent and latched on the first clock of an audio channel.Figure 6.14-4 shows the timing diagram for I$^2$S data format and Figure 6.14-5 shows the I$^2$S MSB justified data format.

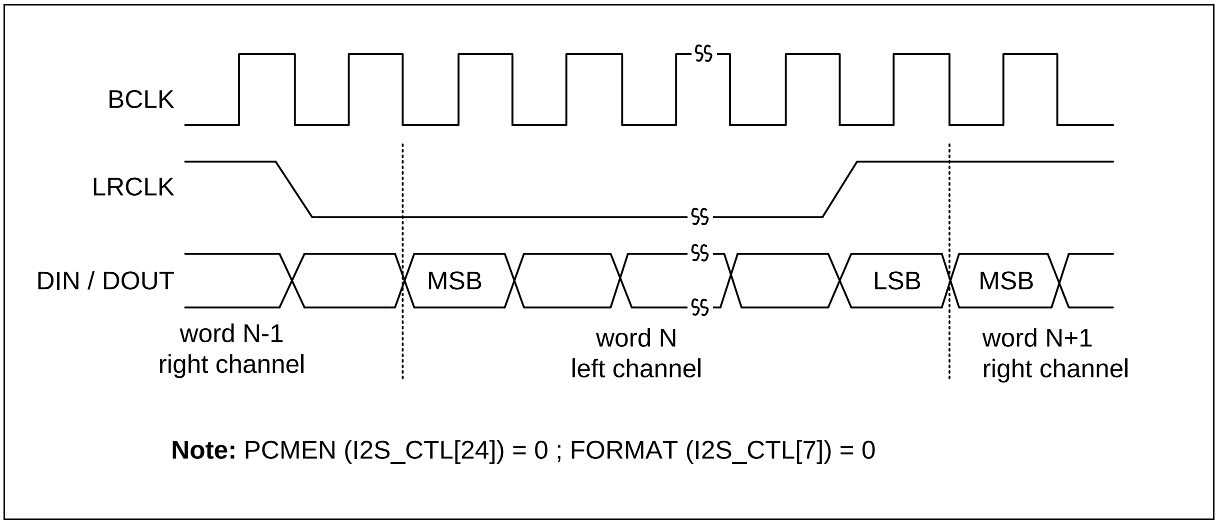


Figure 6.14-4 I²S Bus Timing Diagram

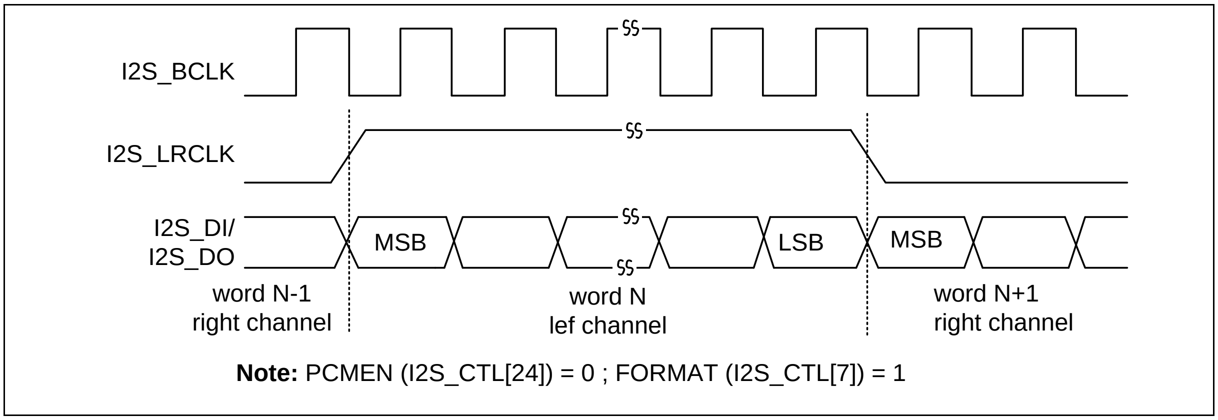
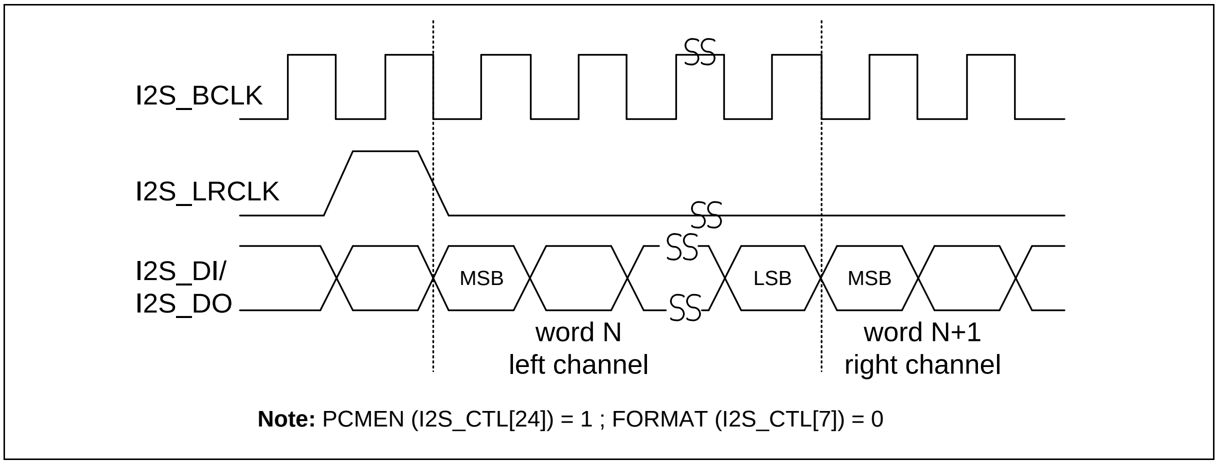


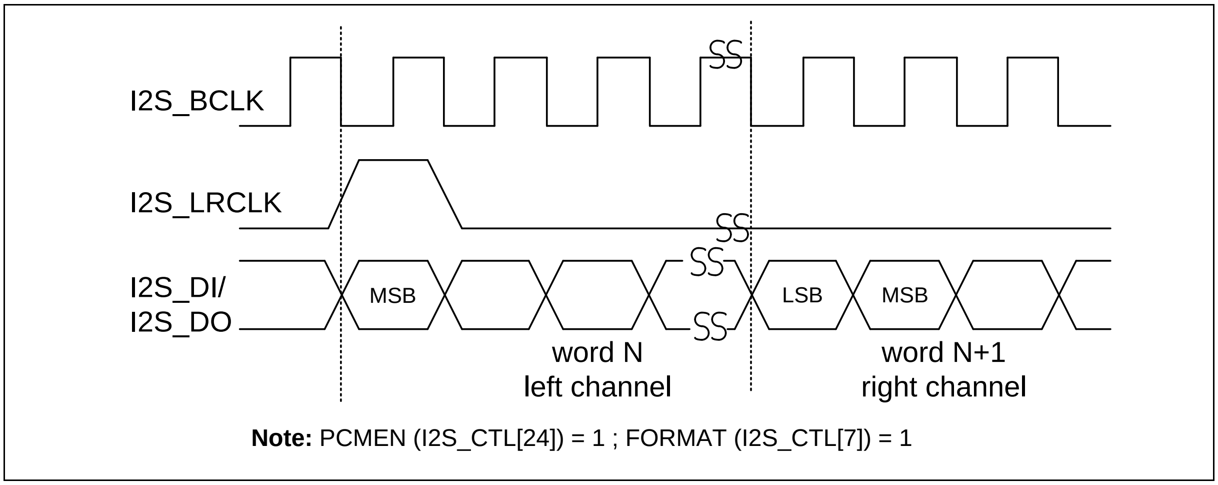
Figure 6.14-5 MSB Justified Timing Diagram

The I²S also supports PCM-A and PCM-B data format. The LRCLK signal will be 1 clock pulse to indicate the start of the new data. The left channel and right channel data follow in order. The bit count of an audio channel is determined by WDWIDTH (I2S_CTL[5:4]). In PCM-A data format, the MSB is sent and latched on the second clock of an audio channel. In PCM-B data format, the MSB is sent and latched on the first clock of an audio channel. Figure 6.14-6 and Figure 6.14-7 show the timing diagram for PCM-A and PCM-B data formats.



**Note:** PCMEN (I2S_CTL[24]) = 1 ; FORMAT (I2S_CTL[7]) = 0

Figure 6.14-6 PCM-A Audio Timing Diagram



**Note:** PCMEN (I2S_CTL[24]) = 1 ; FORMAT (I2S_CTL[7]) = 1

Figure 6.14-7 PCM-B Audio Timing Diagram

### 6.14.5 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.

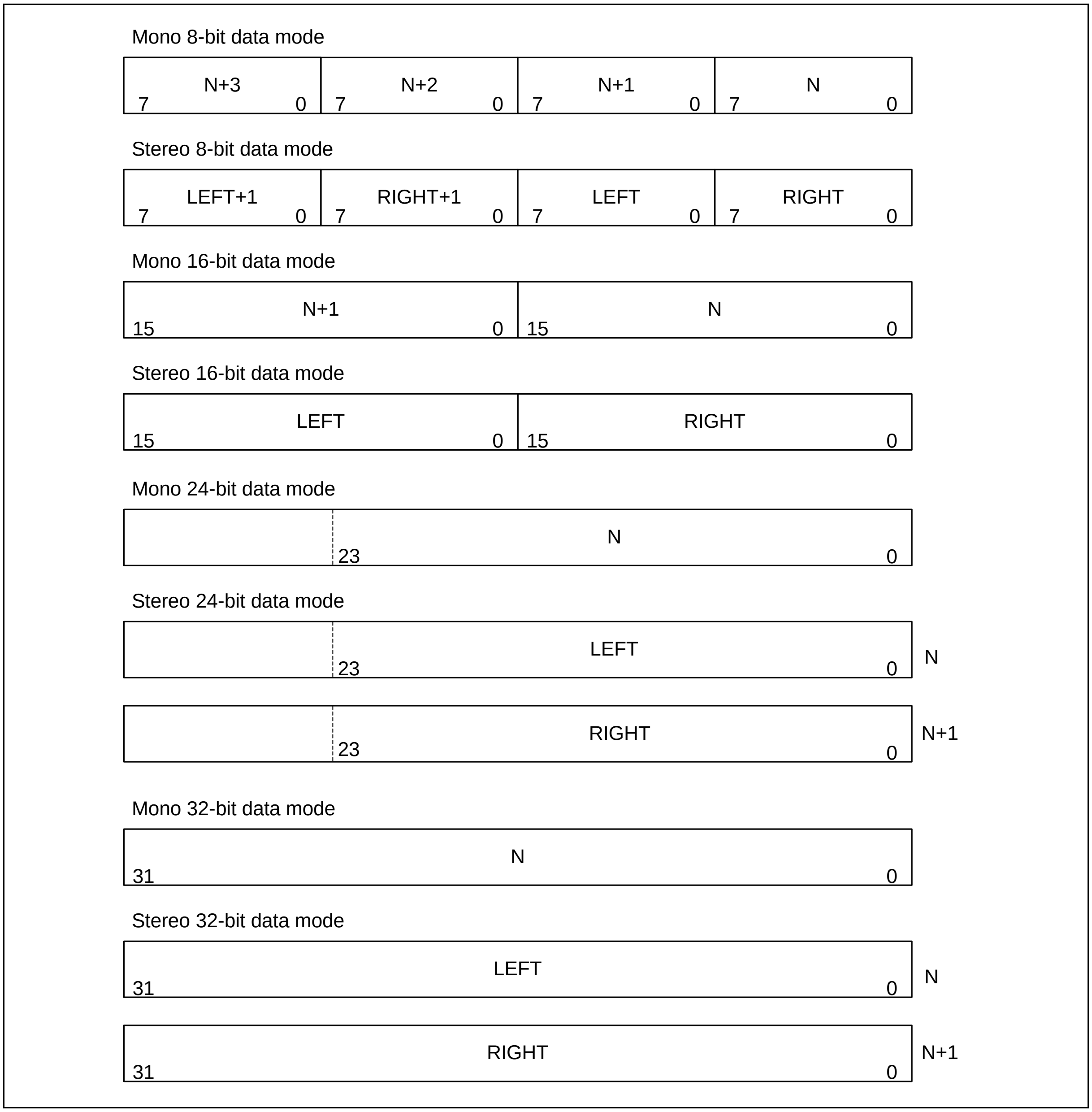


Figure 6.14-8 FIFO Contents for Various I$^2$S Modes

### 6.14.6 Functional Description

#### *6.14.6.1 Zero Crossing*

When playing the audio by I$^2$S function, the output data comes from the memory by CPU or DMA. However, it may result in some pop noise if the playing gain level is changed by user at any time.

Because if the output data is not zero, the output data cross the gain change will generate a non-continuous data. Also, the non-continuous data will cause pop noise. Therefore, the function zero crossing can help to reduce this situation. If the zero crossing function is enabled, hardware will detect if the next transfer data is zero or sign change. If the next data is zero or sign change, the zero crossing flags, LZCIF (I2S_STATUS[23]) and RZCIF (I2S_STATUS[22]), will be set to high. Then, the next transfer data will be held and output zero data automatically until the flags are cleared by user. Therefore, if user wants to modify the audio playing gain, users can enable the zero crossing interrupt function, LZCIEN (I2S_IEN[12]) and RZCIEN (I2S_IEN[11]), to indicate the zero crossing time and to change the audio gain. This will reduce the pop noise.

### *6.14.6.2 DMA Mode*

The I$^2$S function can use DMA function to access the data. If enabling TX DMA function, when transmit FIFO is not full, the I$^2$S will generate the request signal and get a data from memory automatically, until the transmit FIFO is full. However, if enabling RX DMA function, when the receive FIFO is not empty, the I$^2$S will generate the request signal and move receive data to memory automatically, until the receive FIFO is empty. Therefore, user can use TX DMA and RX DMA to share the CPU loading when recording and playing the audio.

**Example-1:** Use TX DMA to play the audio, and the audio data is at the memory address from 0x2001_0300 to 0x2001_03FC.

1. Set the correct value to MONO (I2S_CTL[6]), WDWIDTH (I2S_CTL[5:4]), FORMAT (I2S_CTL[7]) for audio format.
2. Set 0x2001_0300 to I2S_TXSTADDR (The register is word boundary).
3. Set 0x2001_03FC to I2S_TXEADDR (The register is word boundary).
4. Set TXDMAEN (I2S_CTL[20]) to 1.
5. Set TXEN (I2S_CTL[1]) and I2SEN (I2S_CTL[0]) to 1.
6. I$^2$S controller will use TX DMA to play the data from address 0x2001_0300 to 0x2001_03FC again and again.

**Example-2:** Use RX DMA to record the Audio, and move the audio to memory address from 0x2001_0300 to 0x2001_03FC.

1. Set correct value to MONO (I2S_CTL[6]), WDWIDTH (I2S_CTL[5:4]), FORMAT (I2S_CTL[7]) for audio format.
2. Set 0x2001_0300 to I2S_RXSTADDR (The register is word boundary).
3. Set 0x2001_03FC to I2S_RXEADDR (The register is word boundary).
4. Set RXDMAEN (I2S_CTL[21]) to 1.
5. Set RXEN (I2S_CTL[2]) and I2SEN (I2S_CTL[0]) to 1.
6. I$^2$S controller will use RX DMA to record the data to the memory address from 0x2001_0300 to 0x2001_03FC again and again.

#### *6.14.6.3 Master/Slave Interface*

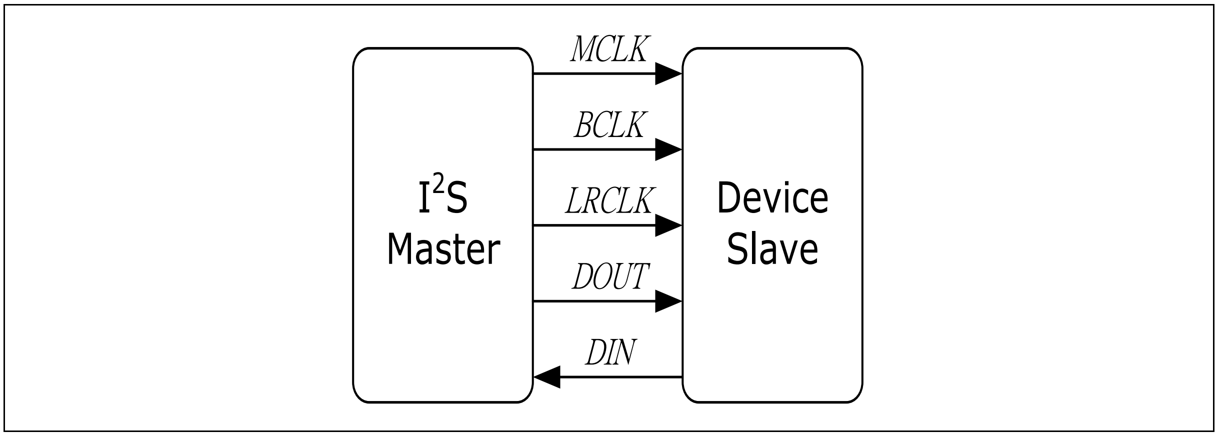


Figure 6.14-9 Master Mode Interface Block Diagram

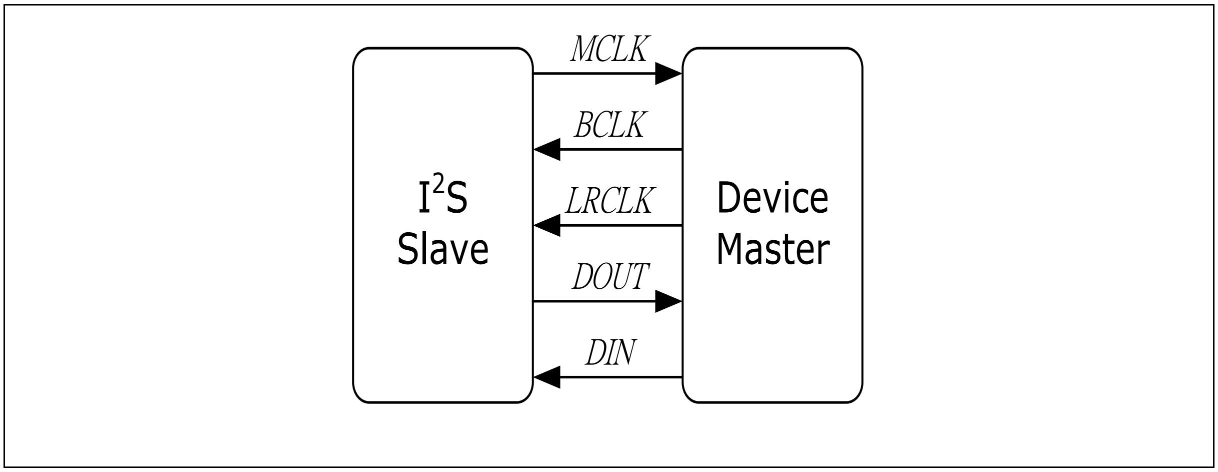


Figure 6.14-10 Slave Mode Interface Block Diagram

In Master mode, the MCLK, BCLK, LRCLK is output to device slave. However, in Slave mode, the MCLK is output to device master, and BCLK or LRCLK is input from device master.

#### *6.14.6.4 Internal Audio CODEC Interface*

The NUC505 has one internal audio CODEC. The control interface for internal audio CODEC is I$^2$C protocol and the audio interface is I$^2$S protocol (refer to Figure 6.14-1). Thus, the I$^2$S controller has a virtual I$^2$C converter block to convert the APB command to I$^2$C protocol command for easily reading or writing internal audio CODEC registers. Following are the examples to write and read internal audio CODEC registers.

**Example 1:** Write the internal audio CODEC register.

If you want to change the left channel analog gain, AVOLL (CODEC_DACLVACTL[4:0]), to mute. (For the internal audio CODEC registers, refer to section 6.15.9 and 6.15.10.)

1. Set I2SCKEN (CLK_APBCLK[14]) to 1 to open I$^2$S clock.
2. Set 0x2081_1081 to I2S_CODECCTL. This means:

   a. The frequency for SCLK of I$^2$C protocol (F_SCLK) is equal F_I2SCLK / (32*16). Which F_I2SCLK means the frequency for I$^2$S engine clock, and the value 32 is from I2CCKDIV (I2S_CODECCTL[30:24]). The value 16 is the fixed value.
   b. User wants to write internal audio CODEC register, because RW (I2S_CODECTL[16]) equal to 1.
   c. The internal audio CODEC address is 0x08 and the data is 0x1F (mute).
3. Wait BUSY (I2S_CODECCTL[31]) to 0 and the I$^2$C command is done.

**Note1:** ”F_” means the frequency of.

**Note2:** F_SCLK should be lower than F_MCLK for 32 times.

**Note3:** For the internal audio CODEC register address, refer to section 6.15.9.

**Example 2:** Read the internal audio CODEC register.

If you want to read the left channel analog gain, AVOLL (CODEC_DACLVACTL[4:0]).

1. Set I2SCKEN (CLK_APBCLK[14]) to 1 to open I$^2$S clock.
2. Set 0x2080_0800 to I2S_CODECCTL. This means :
   a. The frequency for SCLK of I$^2$C protocol (F_SCLK) is equal F_I2SCLK / (32*16). Which F_I2SCLK means the frequency for I$^2$S engine clock, and the value 32 is from I2CCKDIV (I2S_CODECCTL[30:24]). The value 16 is the fixed value.
   b. User wants to write internal audio CODEC register, because RW (I2S_CODECTL[16]) equal to 0.
   c. The internal audio CODEC address is 0x08.
3. Wait BUSY (I2S_CODECCTL[31]) to 0.
4. Read DAT (I2S_CODECCTL[7:0]), and the value is left channel analog gain.

**Note1:** ”F_” means the frequency of.

**Note2:** F_ SCLK should be lower than F_MCLK for 32 times.

**Note3:** For the internal audio CODEC register address, refer to section 6.15.9.

### *6.14.6.5 MCLK, BCLK and LRCLK Generation*

**I$^2$S in Master Mode:**

If the internal audio CODEC (CODECSEL (I2S_CTL[28]) = 0) is used, the frequency of MCLK (F_MCLK) must be 256 times bigger than the audio sample frequency (F_LRCLK). The frequency of BCLK must be 64 times bigger than F_LRCLK. The value 64 is calculated from 2*32, which 2 means stereo channel and 32 means one channel audio sample has 32 BCLK cycles.

But if the external audio CODEC (CODECSEL (I2S_CTL[28] = 1) is used, F_MCLK can be 256/128/64 times larger than F_LRCLK, which is dependent on the external audio CODEC requirement. The F_BCLK must be lower than F_MCLK and can be 2*8 / 2*16 / 2*24 / 2*32 times larger than F_LRCLK. The values 8/16/24/32 are selected by register WDWIDTH (I2S_CLK[5:4]).

*WDWIDTH = 0x0:* It means one channel audio sample has 8 BCLK cycles.
*WDWIDTH = 0x1:* It means one channel audio sample has 16 BCLK cycles.
*WDWIDTH = 0x2:* It means one channel audio sample has 24 BCLK cycles.
*WDWIDTH = 0x3:* It means one channel audio sample has 32 BCLK cycles.

Therefore, the following steps can help to set BCLKDIV (I2S_CLKDIV[16:8]) and MCLKDIV (I2S_CLKDIV[5:0]),

**Internal Audio CODEC (CODECSEL = 0):**

1. Select the audio sample frequency (F_LRCLK)
2. F_MCLK = 256 * F_LRCLK.
3. F_BCLK = 64 * F_LRCLK.
4. Set BCLKDIV (I2S_CLKDIV[16:8]) and MCLKDIV (I2S_CLKDIV[5:0]) to get the correct F_BCLK and F_MCLK.

   *F_MCLK = F_I2SCLK / (2*MCLKDIV) (if MCLKDIV ≠ 0)*  
   *= F_I2SCLK (if MCLKDIV = 0)*  
   *F_BCLK = F_I2SCLK / (2*(BCLKDIV+1))*

**External Audio CODEC (CODECSEL = 1):**

1. Select the audio sample frequency (F_LRCLK)
2. F_MCLK = 384 (or 256 or 128 or 64) * F_LRCLK. (The value depends on external audio CODEC requirement and must be bigger than F_BCLK)
3. F_BCLK = 2 * 8 * F_LRCLK (if WDWIDTH = 0x0)  
   = 2 * 16 * F_LRCLK (if WDWIDTH = 0x1)  
   = 2 * 24 * F_LRCLK (if WDWIDTH = 0x2)  
   = 2 * 32 * F_LRCLK (if WDWIDTH = 0x3)
4. Set BCLKDIV (I2S_CLKDIV[16:8]) and MCLKDIV (I2S_CLKDIV[5:0]) to get the correct F_BCLK and F_MCLK.

   *F_MCLK = F_I2SCLK / (2*MCLKDIV) (if MCLKDIV ≠ 0)*  
   *= F_I2SCLK (if MCLKDIV = 0)*  
   *F_BCLK = F_I2SCLK / (2*(BCLKDIV+1))*

| Internal Audio CODEC: CODECSEL = 0 ; I2SSEL (CLK_CLKDIV2[25:24]) = 0x3; | | | | | | | |
|---|---|---|---|---|---|---|---|
| **F_LRCLK** | **WDWIDTH** | **F_MCLK** | **F_BCLK** | **BCLKDIV** | **MCLKDIV** | **I2SDIV** | **CLK_APLLCTL** |
| **(KHz)** | **[1:0]** | **(MHz)** | **(MHz)** | **[8:0]** | **[5:0]** | **[7:0]** | **[31:0]** |
| 96 | 0x0 | 24.576 | 6.144 | 0x3 | 0x1 | 0x0 | 0x424D542 |
| | 0x1 | | | | | | |
| | 0x2 | | | | | | |
| | 0x3 | | | | | | |
| 48 | 0x0 | 12.288 | 3.072 | 0x7 | 0x2 | 0x0 | 0x424D542 |
| | 0x1 | | | | | | |
| | 0x2 | | | | | | |
| | 0x3 | | | | | | |
| 44.1 | 0x0 | 11.2896 | 2.8224 | 0x7 | 0x2 | 0x0 | 0x704D382 |
| | 0x1 | | | | | | |
| | 0x2 | | | | | | |
| | 0x3 | | | | | | |
| 32 | 0x0 | 8.192 | 2.048 | 0xB | 0x3 | 0x0 | 0x424D542 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | 0x1 | | | | | | |
| | 0x2 | | | | | | |
| | 0x3 | | | | | | |
| 16 | 0x0 | 4.096 | 1.024 | 0x17 | 0x6 | 0x0 | 0x424D542 |
| | 0x1 | | | | | | |
| | 0x2 | | | | | | |
| | 0x3 | | | | | | |
| 8 | 0x0 | 2.048 | 0.512 | 0x2F | 0xC | 0x0 | 0x424D542 |
| | 0x1 | | | | | | |
| | 0x2 | | | | | | |
| | 0x3 | | | | | | |

Table 6.14-1 Clock Reference with Internal Audio CODEC

| External Audio CODEC : CODECSEL = 1 ; I2SSEL (CLK_CLKDIV2[25:24]) = 0x3;<br>Note: Set F_MCLK = 256 * F_LRCLK | | | | | | | |
|---|---|---|---|---|---|---|---|
| F_LRCLK (KHz) | WDWIDTH [1:0] | F_MCLK (MHz) | F_BCLK (MHz) | BCLKDIV [8:0] | MCLKDIV [5:0] | I2SDIV [7:0] | CLK_APLLCTL [31:0] |
| 96 | 0x0 | 24.576 | 1.536 | 0xF | 0x1 | 0x0 | 0x424D542 |
| | 0x1 | | 3.072 | 0x7 | 0x1 | | |
| | 0x3 | | 6.144 | 0x3 | 0x1 | | |
| 48 | 0x0 | 12.288 | 0.768 | 0x1F | 0x2 | 0x0 | 0x424D542 |
| | 0x1 | | 1.536 | 0xF | 0x2 | | |
| | 0x3 | | 3.072 | 0x7 | 0x2 | | |
| 44.1 | 0x0 | 11.2896 | 0.7056 | 0x1F | 0x2 | 0x0 | 0x704D382 |
| | 0x1 | | 1.4112 | 0xF | 0x2 | | |
| | 0x3 | | 2.8224 | 0x7 | 0x2 | | |
| 32 | 0x0 | 8.192 | 0.512 | 0x2F | 0x3 | 0x0 | 0x424D542 |
| | 0x1 | | 1.024 | 0x17 | 0x3 | | |
| | 0x3 | | 2.048 | 0xB | 0x3 | | |
| 16 | 0x0 | 4.096 | 0.256 | 0x5F | 0x6 | 0x0 | 0x424D542 |
| | 0x1 | | 0.512 | 0x2F | 0x6 | | |
| | 0x3 | | 1.024 | 0x17 | 0x6 | | |
| 8 | 0x0 | 2.048 | 0.128 | 0xBF | 0xC | 0x0 | 0x424D542 |
| | 0x1 | | 0.256 | 0x5F | 0xC | | |
| | 0x3 | | 0.512 | 0x2F | 0xC | | |

Table 6.14-2 Clock Reference with External Audio CODEC

**Note:** When using external audio CODEC and WDWIDTH (I2S_CTL[5:4]) = 0x2, it would be better that I$^2$S controller operates in Slave mode.

**I$^2$S in Slave Mode:**

When the internal audio CODEC (CODECSEL = 0) is used, the I$^2$S does not support Slave mode. If the external audio CODEC (CODECSEL = 1) is used, the BCLK and LRCLK are input from external audio CODEC. Thus, F_BCLK and F_LRCLK are controlled by external audio CODEC. But F_MCLK is still an output signal, and dependent on the external audio CODEC requirement.

$$\begin{aligned} F\_MCLK &= F\_I2SCLK / (2*MCLKDIV) && \text{(if MCLKDIV} \neq 0) \\ &= F\_I2SCLK && \text{(if MCLKDIV} = 0) \end{aligned}$$

### *6.14.6.6 RX Data Power Measurement*

The I$^2$S controller supports rx data power measurement function. It can help users to easily control input audio signal gain. User can get the power average result from register I2S_RXLCHAVG and I2S_RXRCHAVG to know how big volume the audio input signal has. Then, user can control the volume gain to adjust audio signal for target level.

The power measurement method is to calculate the absolute value of the rx data between the setting window period. Therefore, the register WINSEL (I2S_RXAVGCTL[3:0]) needs to be set first.

**Example:**

If WINSEL (I2S_RXAVGCTL[3:0]) = 0x2, it means the average window is 4 samples. Thus, every 4 input audio signal will generate one average result. In other words, the result represents the average power for these 4 samples.

### *6.14.6.7 Interrupt Sources*

Figure 6.14-11 shows the interrupt sources of I$^2$S controller. Each of the interrupt sources can be enabled individually.

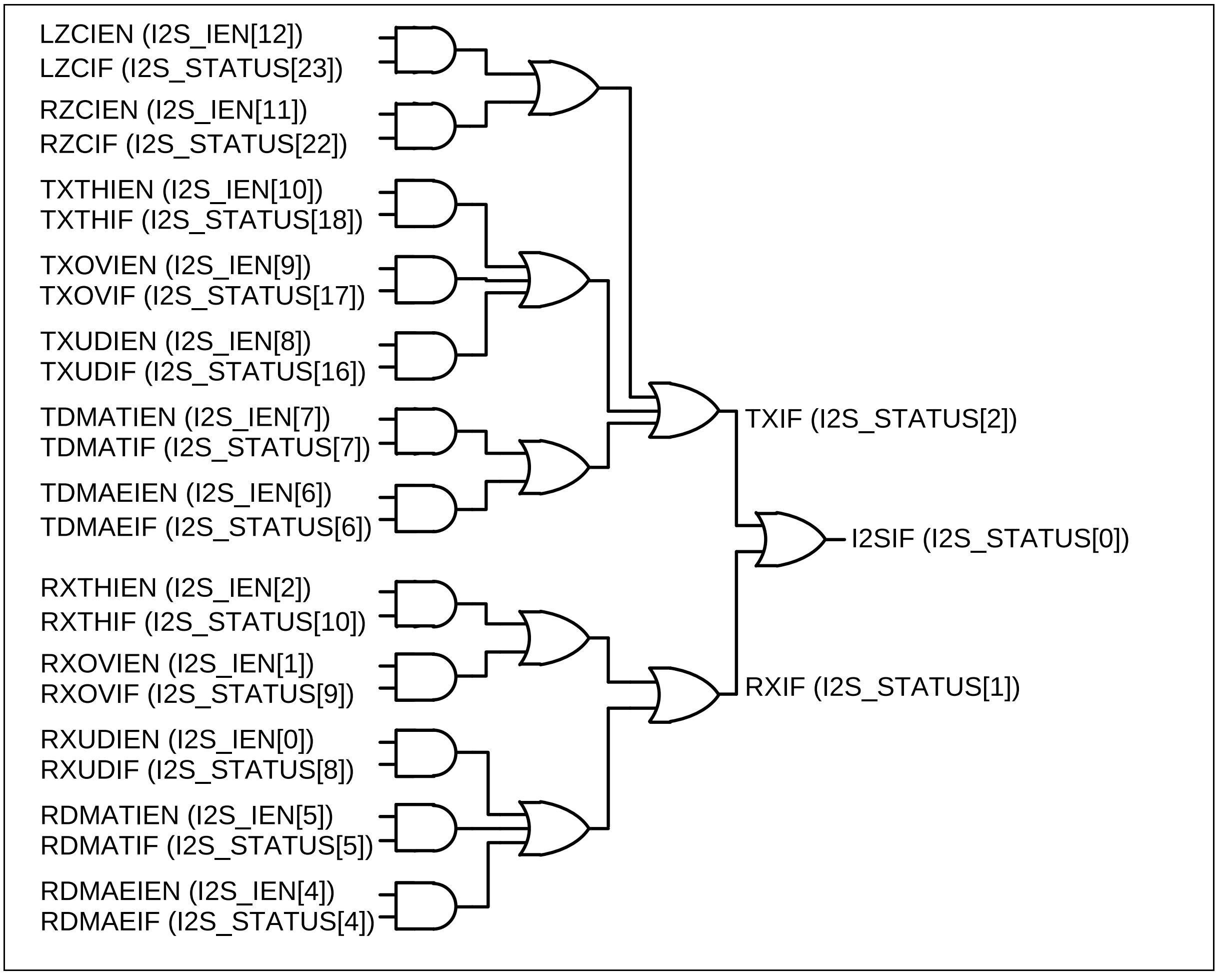


Figure 6.14-11 I$^2$S Controller Interrupts

### 6.14.7 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I$^{2}$S Base Address:**<br>**I2S_BA = 0x400E_8000** | | | | |
| **I2S_CTL** | I2S_BA+0x00 | R/W | I$^{2}$S Control Register | 0x2000_0000 |
| **I2S_CLKDIV** | I2S_BA+0x04 | R/W | I$^{2}$S Clock Divider Control Register | 0x0000_0000 |
| **I2S_IEN** | I2S_BA+0x08 | R/W | I$^{2}$S Interrupt Enable Register | 0x0000_0000 |
| **I2S_STATUS** | I2S_BA+0x0C | R/W | I$^{2}$S Status Register | 0x0014_1000 |
| **I2S_TX** | I2S_BA+0x10 | W | I$^{2}$S Transmit FIFO Register | 0x0000_0000 |
| **I2S_RX** | I2S_BA+0x14 | R | I$^{2}$S Receive FIFO Register | 0x0000_0000 |
| **I2S_CODECCTL** | I2S_BA+0x18 | R/W | I$^{2}$S Virtual I$^{2}$C Control Register | 0x0000_0000 |
| **I2S_TXSTADDR** | I2S_BA+0x20 | R/W | I$^{2}$S TX DMA Start Address Register | 0x2001_0000 |
| **I2S_TXTHADDR** | I2S_BA+0x24 | R/W | I$^{2}$S TX DMA Threshold Address Register | 0x2001_FFF0 |
| **I2S_TXEADDR** | I2S_BA+0x28 | R/W | I$^{2}$S TX DMA End Address Register | 0x2001_FFF0 |
| **I2S_TXCADDR** | I2S_BA+0x2C | R | I$^{2}$S TX DMA Current Address Register | 0x0000_0000 |
| **I2S_RXSTADDR** | I2S_BA+0x30 | R/W | I$^{2}$S RX DMA Start Address Register | 0x2001_0000 |
| **I2S_RXTHADDR** | I2S_BA+0x34 | R/W | I$^{2}$S RX DMA Threshold Address Register | 0x2001_FFF0 |
| **I2S_RXEADDR** | I2S_BA+0x38 | R/W | I$^{2}$S RX DMA End Address Register | 0x2001_FFF0 |
| **I2S_RXCADDR** | I2S_BA+0x3C | R | I$^{2}$S RX DMA Current Address Register | 0x0000_0000 |
| **I2S_RXAVGCTL** | I2S_BA+0x40 | R/W | I$^{2}$S RX Data Average Control Register | 0x0000_000A |
| **I2S_RXLCHAVG** | I2S_BA+0x44 | R | I$^{2}$S RX Left Channel Data Average | 0x0000_0000 |
| **I2S_RXRCHAVG** | I2S_BA+0x48 | R | I$^{2}$S RX Right Channel Data Average | 0x0000_0000 |

### 6.14.8 Register Description

#### I$^2$S Control Register (I2S_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_CTL** | I2S_BA+0x00 | R/W | I$^2$S Control Register | 0x2000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | CODECRST | CODECSEL | Reserved | SLAVE | MCLKEN | PCMEN |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| RXLCH | Reserved | RXDMAEN | TXDMAEN | RXCLR | TXCLR | LZCEN | RZCEN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| RXTH | | | | TXTH | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FORMAT | MONO | WDWIDTH | | MUTE | RXEN | TXEN | I2SEN |

| Bits | Description | |
|---|---|---|
| [31:30] | **Reserved** | Reserved. |
| [29] | **CODECRST** | **Internal CODEC Hardware Reset Control**<br>0 = Reset Operation.<br>1 = Normal Operation. |
| [28] | **CODECSEL** | **Internal CODEC or External CODEC Selection**<br>0 = I$^2$S interface connected to internal CODEC.<br>1 = I$^2$S interface connected to external CODEC. |
| [27] | **Reserved** | Reserved. |
| [26] | **SLAVE** | **Slave Mode**<br>I$^2$S can operate as master or slave. For Master mode, I2S_BCLK and I2S_LRCLK signals are output to CODEC. In Slave mode, I2S_BCLK and I2S_LRCLK pins are received from CODEC.<br>0 = Master mode.<br>1 = Slave mode.<br>**Note:** If using internal CODEC, the I$^2$S must be master mode. |
| [25] | **MCLKEN** | **Master Clock Enable Control**<br>0 = I$^2$S master clock output Disabled.<br>1 = I$^2$S master clock output Enabled.<br>**Note1:** I2S_MCLK is always output.<br>**Note2:** I2S_MCLK frequency is controlled by MCLKDIV[5:0]. |
| [24] | **PCMEN** | **PCM Interface Enable Control**<br>0 = I$^2$S Interface.<br>1 = PCM Interface. |

| | | |
|---|---|---|
| [23] | **RXLCH** | **Receive Left Channel Enable Control**<br>When monaural format is selected (MONO = 1), I$^2$S will receive right channel data if.<br>RXLCH is set to 0, and receive left channel data if RXLCH is set to 1.<br>0 = Receives right channel data when monaural format is selected.<br>1 = Receives left channel data when monaural format is selected. |
| [22] | **Reserved** | Reserved. |
| [21] | **RXDMAEN** | **RX DMA Enable Control (Record Path)**<br>0 = RX DMA mode Disabled.<br>1 = RX DMA mode Enabled.<br>**Note:** The I2S_RXSTADDR will be updated to new setting only when RXDMAEN is from low to high. Therefore, if you want to change I2S_RXSTADDR, you should confirm RXDMAEN is disabled. |
| [20] | **TXDMAEN** | **TX DMA Enable Control (Transmit Path)**<br>0 = TX DMA mode Disabled.<br>1 = TX DMA mode Enabled.<br>**Note:** The I2S_TXSTADDR will be updated to new setting only when TXDMAEN is from low to high. Therefore, if you want to change I2S_TXSTADDR, you should confirm TXDMAEN is disabled. |
| [19] | **RXCLR** | **Clear Receive FIFO**<br>0 = No effect.<br>1 = Receiver FIFO will be cleared.<br>**Note:** This bit will be cleared to 0 automatically.<br>**Note2:** If clearing the receiver FIFO, RXCNT (I2S_STATUS[27:24]) returns to 0x0 and receiver FIFO becomes empty. |
| [18] | **TXCLR** | **Clear Transmit FIFO**<br>0 = No effect.<br>1 = Transmit FIFO will be cleared.<br>**Note:** This bit will be cleared to 0 automatically.<br>**Note2:** If clearing the transmit FIFO, TXCNT (I2S_STATUS[31:28]) returns to 0x0 and transmit FIFO becomes empty. |
| [17] | **LZCEN** | **Left Channel Zero-cross Detect Enable Control**<br>If this bit is set to 1, when left channel data sign bit change or next shift data bits are all zero then LZCIF flag in I2S_STATUS register is set to 1.<br>0 = Left channel zero-cross detect Disabled.<br>1 = Left channel zero-cross detect Enabled. |
| [16] | **RZCEN** | **Right Channel Zero-cross Detection Enable Control**<br>If this bit is set to 1, when right channel data sign bit change or next shift data bits are all zero then RZCIF flag in I2S_STATUS register is set to 1.<br>0 = Right channel zero-cross detect Disabled.<br>1 = Right channel zero-cross detect Enabled. |

| | | |
|---|---|---|
| [15:12] | **RXTH** | **Receive FIFO Threshold Level**<br>When received data word(s) in buffer is equal to or higher than threshold level then RXTHIF flag is set.<br>0000 = 1 word data in receive FIFO.<br>0001 = 2 word data in receive FIFO.<br>0010 = 3 word data in receive FIFO.<br>0011 = 4 word data in receive FIFO.<br>0100 = 5 word data in receive FIFO.<br>0101 = 6 word data in receive FIFO.<br>0110 = 7 word data in receive FIFO.<br>0111 = 8 word data in receive FIFO.<br>1000 = 9 word data in receive FIFO.<br>1001 = 10 word data in receive FIFO.<br>1010 = 11 word data in receive FIFO.<br>1011 = 12 word data in receive FIFO.<br>1100 = 13 word data in receive FIFO.<br>1101 = 14 word data in receive FIFO.<br>1110 = 15 word data in receive FIFO.<br>1111 = 16 word data in receive FIFO. |
| [11:8] | **TXTH** | **Transmit FIFO Threshold Level**<br>If remain data word (32 bits) in transmit FIFO is the same or less than threshold level then TXTHIF flag is set.<br>0000 = 0 word data in transmit FIFO.<br>0001 = 1 word data in transmit FIFO.<br>0010 = 2 words data in transmit FIFO.<br>0011 = 3 words data in transmit FIFO.<br>0100 = 4 words data in transmit FIFO.<br>0101 = 5 words data in transmit FIFO.<br>0110 = 6 words data in transmit FIFO.<br>0111 = 7 words data in transmit FIFO.<br>1000 = 8 word data in transmit FIFO.<br>1001 = 9 word data in transmit FIFO.<br>1010 = 10 words data in transmit FIFO.<br>1011 = 11 words data in transmit FIFO.<br>1100 = 12 words data in transmit FIFO.<br>1101 = 13 words data in transmit FIFO.<br>1110 = 14 words data in transmit FIFO.<br>1111 = 15 words data in transmit FIFO. |
| [7] | **FORMAT** | **Data Format Selection**<br>If PCMEN=0,.<br>0 = I$^{2}$S data format.<br>1 = MSB justified data format.<br>If PCMEN=1,.<br>0 = PCM mode A.<br>1 = PCM mode B. |

| | | |
|---|---|---|
| [6] | **MONO** | **Monaural Data**<br>0 = Data is stereo format.<br>1 = Data is monaural format.<br>**Note:** When chip records data and MONO = 1, RXLCH (I2S_CTL[23]) will control the selection for recording right channel data or left channel data. |
| [5:4] | **WDWIDTH** | **Word Width**<br>00 = Data is 8-bit.<br>01 = Data is 16-bit.<br>10 = Data is 24-bit.<br>11 = Data is 32-bit. |
| [3] | **MUTE** | **Transmit Mute Enable Control**<br>0 = Transmit data is shifted from buffer.<br>1 = Transmit data is fixed to zero. |
| [2] | **RXEN** | **Receive Enable Control**<br>0 = Data receiving Disabled.<br>1 = Data receiving Enabled. |
| [1] | **TXEN** | **Transmit Enable Control**<br>0 = Data transmission Disabled.<br>1 = Data transmission Enabled. |
| [0] | **I2SEN** | **I$^2$S Controller Enable Control**<br>0 = Disabled.<br>1 = Enabled. |

## I2S Clock Divider Control Register (I2S_CLKDIV)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_CLKDIV** | I2S_BA+0x04 | R/W | I$^2$S Clock Divider Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:17] | **Reserved** | Reserved. |
| [16:8] | **BCLKDIV** | **Bit Clock Divider**<br>User can program these bits to generate the frequency of BCLK, when I$^2$S operates in master mode. In Slave mode, the frequency of BCLK is controlled by master device.<br>F_BCLK = F_I2SCLK / (2*(BCLKDIV+1)). |
| [7:6] | **Reserved** | Reserved. |
| [5:0] | **MCLKDIV** | **Master Clock Divider**<br>If F_I2SCLK is (2*MCLKDIV)*256*F_LRCLK then software can program these bits to generate 256*F_LRCLK clock frequency as master clock to audio CODEC. But if MCLKDIV is set to 0, MCLK is the same as I2SCLK input.<br>For example, if sampling rate is 24 kHz (F_LRCLK = 24 kHz) and F_I2SCLK is 12.288 MHz, MCLKDIV should be set to 1 to get 256*F_LRCLK frequency. (12.288 MHz= (2*1*256*24) kHz).<br>F_MCLK = F_I2SCLK / (2*MCLKDIV)   (When MCLKDIV >= 1).<br>F_MCLK = F_I2SCLK   (When MCLKDIV   = 0). |

### I²S Interrupt Enable Register (I2S_IEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_IEN** | I2S_BA+0x08 | R/W | I²S Interrupt Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | **LZCIEN** | **RZCIEN** | **TXTHIEN** | **TXOVIEN** | **TXUDIEN** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **TDMATIEN** | **TDMAEIEN** | **RDMATIEN** | **RDMAEIEN** | **Reserved** | **RXTHIEN** | **RXOVIEN** | **RXUDIEN** |

| Bits | Description | |
|---|---|---|
| [31:13] | **Reserved** | Reserved. |
| [12] | **LZCIEN** | **Left Channel Zero-cross Interrupt Enable Control**<br>Interrupt occurs if this bit is set to 1 and left channel zero-cross is detected.<br>0 = Interrupt Disabled.<br>1 = Interrupt Enabled. |
| [11] | **RZCIEN** | **Right Channel Zero-cross Interrupt Enable Control**<br>Interrupt occurs if this bit is set to 1 and right channel zero-cross is detected.<br>0 = Interrupt Disabled.<br>1 = Interrupt Enabled. |
| [10] | **TXTHIEN** | **Transmit FIFO Threshold Level Interrupt Enable Control**<br>Interrupt occurs if this bit is set to 1 and data words in transmit FIFO is less than TXTH[3:0].<br>0 = Interrupt Disabled.<br>1 = Interrupt Enabled. |
| [9] | **TXOVIEN** | **Transmit FIFO Overflow Interrupt Enable Control**<br>Interrupt occurs if this bit is set to 1 and transmit FIFO overflow flag is set to 1.<br>0 = Interrupt Disabled.<br>1 = Interrupt Enabled. |
| [8] | **TXUDIEN** | **Transmit FIFO Underflow Interrupt Enable Control**<br>Interrupt occurs if this bit is set to 1 and transmit FIFO underflow flag is set to 1.<br>0 = Interrupt Disabled.<br>1 = Interrupt Enabled. |

| | | |
|---|---|---|
| [7] | **TDMATIEN** | **TX DMA Threshold Interrupt Enable Control**<br>Interrupt occurs if this bit is set to 1 and DMA current address is equal to I2S_TXTHADDR register<br>0 = Interrupt Disabled.<br>1 = Interrupt Enabled. |
| [6] | **TDMAEIEN** | **TX DMA End Interrupt Enable Control**<br>Interrupt occurs if this bit is set to 1 and DMA current address is equal to I2S_TXEADDR register<br>0 = Interrupt Disabled.<br>1 = Interrupt Enabled. |
| [5] | **RDMATIEN** | **RX DMA Threshold Interrupt Enable Control**<br>Interrupt occurs if this bit is set to 1 and DMA current address is equal to I2S_RXTHADDR register<br>0 = Interrupt Disabled.<br>1 = Interrupt Enabled. |
| [4] | **RDMAEIEN** | **RX DMA End Interrupt Enable Control**<br>Interrupt occurs if this bit is set to 1 and DMA current address is equal to I2S_RXEADDR register<br>0 = Interrupt Disabled.<br>1 = Interrupt Enabled. |
| [3] | **Reserved** | Reserved. |
| [2] | **RXTHIEN** | **Receive FIFO Threshold Level Interrupt Enable Control**<br>When data word in receive FIFO is equal to or higher then RXTH[3:0] and the RXTHIF bit is set to 1. If RXTHIEN bit is enabled, interrupt will occur.<br>0 = Interrupt Disabled.<br>1 = Interrupt Enabled. |
| [1] | **RXOVIEN** | **Receive FIFO Overflow Interrupt Enable Control**<br>0 = Interrupt Disabled.<br>1 = Interrupt Enabled. |
| [0] | **RXUDIEN** | **Receive FIFO Underflow Interrupt Enable Control**<br>If software reads receive FIFO when it is empty the RXUDIF flag in I2S_STATUS register is set to 1.<br>0 = Interrupt Disabled.<br>1 = Interrupt Enabled. |

## I$^2$S Status Register (I2S_STATUS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_STATUS** | I2S_BA+0x0C | R/W | I$^2$S Status Register | 0x0014_1000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| TXCNT | | | | RXCNT | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| LZCIF | RZCIF | TXBUSY | TXEMPTY | TXFULL | TXTHIF | TXOVIF | TXUDIF |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | | | RXEMPTY | RXFULL | RXTHIF | RXOVIF | RXUDIF |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| TDMATIF | TDMAEIF | RDMATIF | RDMAEIF | RIGHT | TXIF | RXIF | I2SIF |

| Bits | Description | |
|---|---|---|
| [31:28] | **TXCNT** | **Transmit FIFO Level**<br>These bits indicate word number in transmit FIFO<br>0000 = No data or 16 words ( need to check the TX full flag).<br>0001 = 1 word in transmit FIFO.<br>….<br>1111 = 15 words in transmit FIFO.<br>**Note:** IF TXFULL flag is 1, and TXCNT = 0x0. It means there are 16 words in the FIFO. |
| [27:24] | **RXCNT** | **Receive FIFO Level**<br>These bits indicate word number in receive FIFO<br>0000 = No data or 16 words (need to check the RX full flag).<br>0001 = 1 word in receive FIFO.<br>….<br>1111 = 15 words in receive FIFO.<br>**Note:** IF RXFULL flag is 1, and RXCNT = 0x0. It means there are 16 words in the FIFO. |
| [23] | **LZCIF** | **Left Channel Zero-cross Flag**<br>It indicates left channel next sample data sign bit is changed or all data bits are zero.<br>0 = No zero-cross.<br>1 = Left channel zero-cross is detected.<br>**Note:** Write 1 to clear this bit to 0. |
| [22] | **RZCIF** | **Right Channel Zero-cross Flag**<br>It indicates right channel next sample data sign bit is changed or all data bits are zero.<br>0 = No zero-cross.<br>1 = Right channel zero-cross is detected.<br>**Note:** Write 1 to clear this bit to 0. |

| | | |
|---|---|---|
| [21] | **TXBUSY** | **Transmit Busy**<br>This bit is cleared to 0 when all data in transmit FIFO and shift buffer is shifted out. This bit is set to 1 when the first data is loaded to shift buffer.<br>0 = Transmit shift buffer is empty.<br>1 = Transmit shift buffer is busy.<br>**Note:** This bit is read only. |
| [20] | **TXEMPTY** | **Transmit FIFO Empty**<br>This bit reflect data word number in transmit FIFO is zero<br>0 = Not empty.<br>1 = Empty.<br>**Note:** This bit is read only. |
| [19] | **TXFULL** | **Transmit FIFO Full**<br>This bit reflect data word number in transmit FIFO is 16<br>0 = Not full.<br>1 = Full.<br>**Note:** This bit is read only. |
| [18] | **TXTHIF** | **Transmit FIFO Threshold Flag**<br>When data word(s) in transmit FIFO is equal or lower than threshold value set in TXTH[3:0] the TXTHIF bit becomes to 1. It keeps at 1 till TXCNT[3:0] is higher than TXTH[3:0] after software write I2S_TX register.<br>0 = Data word(s) in FIFO is higher than threshold level.<br>1 = Data word(s) in FIFO is equal or lower than threshold level.<br>**Note:** This bit is read only. |
| [17] | **TXOVIF** | **Transmit FIFO Overflow Flag**<br>Write data to transmit FIFO when it is full and this bit set to 1<br>0 = No overflow.<br>1 = Overflow.<br>**Note:** Write 1 to clear this bit to 0. |
| [16] | **TXUDIF** | **Transmit FIFO Underflow Flag**<br>When transmit FIFO is empty and shift logic hardware read data from data FIFO causes this set to 1.<br>0 = No underflow.<br>1 = Underflow.<br>**Note:** Write 1 to clear this bit to 0. |
| [15:13] | **Reserved** | Reserved. |
| [12] | **RXEMPTY** | **Receive FIFO Empty**<br>This bit reflects data words number in receive FIFO is zero<br>0 = FIFO not empty.<br>1 = FIFO empty.<br>**Note:** This bit is read only. |
| [11] | **RXFULL** | **Receive FIFO Full**<br>This bit reflect data words number in receive FIFO is 16<br>0 = FIFO not full.<br>1 = FIFO full.<br>**Note:** This bit is read only. |

| | | |
|---|---|---|
| [10] | **RXTHIF** | **Receive FIFO Threshold Flag**<br>When data word(s) in receive FIFO is equal to or higher than the threshold value set in RXTH[3:0] the RXTHIF bit becomes to 1. It keeps at 1 till RXCNT[3:0] is less than RXTH[3:0] after software reads the I2S_RX register.<br>0 = Data word(s) in FIFO is lower than threshold level.<br>1 = Data word(s) in FIFO is equal or higher than threshold level.<br>**Note:** This bit is read only. |
| [9] | **RXOVIF** | **Receive FIFO Overflow Flag**<br>When receive FIFO is full and receive hardware attempt write to data into receive FIFO this bit is set to 1, and data in 1st buffer is overwritten.<br>0 = No overflow occurred.<br>1 = Overflow occurred.<br>**Note:** Write 1 to clear this bit to 0. |
| [8] | **RXUDIF** | **Receive FIFO Underflow Flag**<br>Read receive FIFO when it is empty. Setting this bit to 1 indicates underflow occurred.<br>0 = No underflow occurred.<br>1 = Underflow occurred.<br>**Note:** Write 1 to clear this bit to zero |
| [7] | **TDMATIF** | **TX DMA Equal Threshold Address Interrupt Flag**<br>If TX DMA current address is equal to I2S_TXTHADDR register, this interrupt flag will be set. If the TDMATIEN is set, an interrupt to NVIC will occur.<br>0 = No TX Threshold Interrupt.<br>1 = TX Threshold Interrupt.<br>**Note:** Write 1 to clear this bit to zero |
| [6] | **TDMAEIF** | **TX DMA Equal End Address Interrupt Flag**<br>If TX DMA current address is equal to I2S_TXEADDR register, this interrupt flag will be set. If the TDMAEIEN is set, an interrupt to NVIC will occur.<br>0 = No TX End Interrupt.<br>1 = TX End Interrupt.<br>**Note:** Write 1 to clear this bit to zero |
| [5] | **RDMATIF** | **RX DMA Equal Threshold Address Interrupt Flag**<br>If RX DMA current address is equal to I2S_RXTHADDR register, this interrupt flag will be set. If the RDMATIEN is set, an interrupt to NVIC will occur.<br>0 = No RX Threshold Interrupt.<br>1 = RX Threshold Interrupt.<br>**Note:** Write 1 to clear this bit to zero |
| [4] | **RDMAEIF** | **RX DMA Equal End Address Interrupt Flag**<br>If RX DMA current address is equal to I2S_RXEADDR register, this interrupt flag will be set. If the RDMAEIEN is set, an interrupt to NVIC will occur.<br>0 = No RX End Interrupt.<br>1 = RX End Interrupt.<br>**Note:** Write 1 to clear this bit to zero |
| [3] | **RIGHT** | **Right Channel**<br>Indicates that the current transmit data belongs to right channel<br>0 = Left channel.<br>1 = Right channel.<br>**Note:** This bit is read only, |

| | | |
|---|---|---|
| [2] | **TXIF** | **I$^2$S Transmit Interrupt**<br>0 = No transmit interrupt.<br>1 = Transmit interrupt.<br>**Note1:** This flag is triggered if any of LZCIF, RZCIF, TXTHIF, TXOIF, TXUDIF, TDMATIF, and TDMAEIF occurs.<br>**Note2:** This bit is read only. |
| [1] | **RXIF** | **I$^2$S Receive Interrupt**<br>0 = No receive interrupt.<br>1 = Receive interrupt.<br>**Note1:** This flag is triggered if any of RXTHIF, RXOVIF, RXUDIF, RDMATIF, and RDMAEIF occurs.<br>**Note2:** This bit is read only. |
| [0] | **I2SIF** | **I$^2$S Interrupt Flag**<br>0 = No I$^2$S interrupt.<br>1 = I$^2$S interrupt.<br>**Note1:** This flag is triggered if any of TXIF and RXIF bits are enabled.<br>**Note2:** This bit is read only. |

### I$^2$S Transmit FIFO Register (I2S_TX)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_TX** | I2S_BA+0x10 | W | I$^2$S Transmit FIFO Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **TX** | **Transmit FIFO Register**<br>I$^2$S contains 16 words (16x32 bit) data FIFO for data transmssion. Write data to this register to prepare data for transmission. The remaining word number is indicated by TXCNT[3:0] in I2S_STATUS. |

### I$^2$S Receive FIFO Register (I2S_RX)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_RX** | I2S_BA+0x14 | R | I$^2$S Receive FIFO Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **RX** | **Receive FIFO Register**<br>I$^2$S contains 16 words (16x32 bit) data FIFO for data receiving. Read this register to get data in FIFO. The remaining data word number is indicated by RXCNT[3:0] in I2S_STATUS register. |

### I$^2$S Vitrual I$^2$C Control Register (I2S_CODECCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_CODECCTL** | I2S_BA+0x18 | R/W | I$^2$S Virtual I$^2$C Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **BUSY** | **I2CCKDIV** | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **DEVID** | | | | | | | **RW** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **ADDR** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **DAT** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31] | **BUSY** | **Busy Flag**<br>If the register 'I2S_CODECCTL' has been written, the HW would change the command to the I$^2$C format because the internal audio CODEC interface is I$^2$C. However, the speed of the I$^2$C is slow. Thus, this bit is used to indicate the end of the I$^2$C command.<br>0 = I$^2$C command is finished.<br>1 = I$^2$C command is not finished. |
| [30:24] | **I2CCKDIV** | **SCK Clock Divider**<br>Control the SCK Timing Parameter.<br>The SCK frequency is (F_I2SCLK / (I2CCKDIV * 16)).<br>**Note:** Cannot be zero.<br>**Note2:** F_SCK must be lower than or equal to F_MCLK / 16. |
| [23:17] | **DEVID** | **Internal Audio CODEC Device ID**<br>This parameter should be set to 40H. |
| [16] | **RW** | **Read or Write Command**<br>Control this command to read data from the internal audio CODEC or write data to.<br>0 = Read from the internal audio CODEC.<br>1 = Write to the internal audio CODEC. |
| [15:8] | **ADDR** | **Address Information**<br>This parameter is used to read from the internal audio CODEC or write to the internal audio CODEC. |
| [7:0] | **DAT** | **Data Information**<br>This parameter is used to read from the internal audio CODEC or write to the internal audio CODEC. |

### I$^2$S TX DMA Start Address Register (I2S_TXSTADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_TXSTADDR** | I2S_BA+0x20 | R/W | I$^2$S TX DMA Start Address Register | 0x2001_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ADDR** | **TX DMA Start Address Register**<br>**Note1:** The address is word boundary.<br>**Note2:** The address can’t be set smaller than 0x2000_0000. |

### I$^2$S TX DMA Threshold Address Register (I2S_TXTHADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_TXTHADDR** | I2S_BA+0x24 | R/W | I$^2$S TX DMA Threshold Address Register | 0x2001_FFF0 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ADDR** | **TX DMA Threshold Address Register**<br>**Note1:** The address is word boundary.<br>**Note2:** The address can’t be set smaller than 0x2000_0000. |

### I$^2$S TX DMA End Address Register (I2S_TXEADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_TXEADDR** | I2S_BA+0x28 | R/W | I$^2$S TX DMA End Address Register | 0x2001_FFF0 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ADDR** | **TX DMA End Address Register**<br>**Note1:** The address is word boundary.<br>**Note2:** If WDWIDTH[1:0] is equal to 0x2 or 0x3, user must set the correct end address to avoid the swap between right channel and left channel in stereo mode.<br>**Note2:** The address can’t be set smaller than 0x2000_0000. |

### I$^2$S TX DMA Current Address Register (I2S_TXCADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_TXCADDR** | I2S_BA+0x2C | R | I$^2$S TX DMA Current Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ADDR** | **TX DMA Current Address Register** |

### I$^2$S RX DMA Start Address Register (I2S_RXSTADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_RXSTADDR** | I2S_BA+0x30 | R/W | I$^2$S RX DMA Start Address Register | 0x2001_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ADDR** | **RX DMA Start Address Register**<br>**Note1:** The address is word boundary.<br>**Note2:** The address can't be set smaller than 0x2000_0000. |

### I$^2$S RX DMA Threshold Address Register (I2S_RXTHADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_RXTHADDR** | I2S_BA+0x34 | R/W | I$^2$S RX DMA Threshold Address Register | 0x2001_FFF0 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ADDR** | **RX DMA Threshold Address Register**<br>**Note1:** The address is word boundary.<br>**Note2:** The address can’t be set smaller than 0x2000_0000. |

### I$^2$S RX DMA End Address Register (I2S_RXEADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_RXEADDR** | I2S_BA+0x38 | R/W | I$^2$S RX DMA End Address Register | 0x2001_FFF0 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ADDR** | **RX DMA End Address Register**<br>**Note1:** The address is word boundary.<br>**Note2:** If WDWIDTH[1:0] is equal to 0x2 or 0x3, user must set the correct end address to avoid the swap between right channel and left channel in stereo mode.<br>**Note3:** The address can’t be set smaller than 0x2000_0000. |

### I$^2$S RX DMA Current Address Register (I2S_RXCADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_RXCADDR** | I2S_BA+0x3C | R | I$^2$S RX DMA Current Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ADDR** | **RX DMA Current Address Register** |

### $I^2S$ RX Data Average Control Register (I2S_RXAVGCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_RXAVGCTL** | I2S_BA+0x40 | R/W | $I^2S$ RX Data Average Control Register | 0x0000_000A |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **Reserved** | Reserved. |
| [3:0] | **WINSEL** | **RX Data Average Window Select**<br>0000 = Average window is 1 (2^0) sample.<br>0001 = Average window is 2 (2^1) samples.<br>0010 = Average window is 4 (2^2) samples.<br>…<br>1110 = Average window is 16384 (2^14) samples.<br>1111 = Average window is 32768 (2^15) samples.<br>**Note:** Every window size samples will generate one average result. |

### I$^2$S RX Left Channel Data Average (I2S_RXLCHAVG)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_RXLCHAVG** | I2S_BA+0x44 | R | I$^2$S RX Left Channel Data Average | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **RESULT** | **RX Left Channel Data Average Result**<br>The average result of left channel received data.<br>**Note:** If MONO (I2S_CTL[6]) = 1, the average result is only in this register whatever RXLCH (I2S_CTL[23]) = 1 or 0. |

### $I^2S$ RX Right Channel Data Average (I2S_RXRCHAVG)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **I2S_RXRCHAVG** | I2S_BA+0x48 | R | $I^2S$ RX Right Channel Data Average | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **RESULT** | **RX Right Channel Data Average Result**<br>The average result of left channel received data.<br>**Note:** If MONO (I2S_CTL[6]), this register will be useless. |

### 6.14.9 CODEC Register Map

This section describes the registers of internal audio CODEC. The internal audio CODEC registers cannot be accessed by CPU directly. Refer to section 6.15.6.4 for how to configure them.

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CODEC_ADCCTL** | 0x00 | RW | CODEC ADC Control Register | 0x10 |
| **CODEC_DACCTL** | 0x01 | RW | CODEC DAC Control Register | 0x00 |
| **CODEC_CTL** | 0x02 | RW | CODEC Control Register | 0x80 |
| **CODEC_DACLVCTL** | 0x03 | RW | CODEC Left Channel Volume Control Register | 0x80 |
| **CODEC_DACRVCTL** | 0x04 | RW | CODEC Right Channel Volume Control Register | 0x80 |
| **CODEC_DACLVACTL** | 0x08 | RW | CODEC Left Channel Analog Volume Control Register | 0x00 |
| **CODEC_DACRVACTL** | 0x09 | RW | CODEC Right Channel Analog Volume Control Register | 0x00 |
| **CODEC_DACMIXCTL** | 0x0A | RW | CODEC DAC Output Channel Analog Mix Control Register | 0x09 |
| **CODEC_DACPDCTL** | 0x0B | RW | CODEC DAC Power Down Control Register of Analog Blocks | 0xFF |
| **CODEC_PWRCTL** | 0x0D | RW | CODEC Power Control Register | 0x3A |
| **CODEC_ADCINCTL** | 0x0E | RW | CODEC ADC Input Control Register | 0x00 |
| **CODEC_ADCPDCTL** | 0x0F | RW | CODEC ADC Power Down Control Register | 0xFF |
| **CODEC_ADCLVCTL** | 0x10 | RW | CODEC ADC Left Channel Gain Control Register | 0x00 |
| **CODEC_ADCRVCTL** | 0x11 | RW | CODEC ADC Right Channel Gain Control Register | 0x00 |
| **CODEC_STOCTL** | 0x12 | RW | CODEC ADC Sidetone Volume Control Register | 0x00 |

### 6.14.10 CODEC Register Description

## CODEC ADC Control Register (CODEC_ADCCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CODEC_ADCCTL | 0x00 | RW | CODEC ADC Control Register | 0x10 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| ADCEN | HPFEN | Reserved | STEREO | Reserved | | | |

| Bits | Description | |
|---|---|---|
| [7] | **ADCEN** | **CODEC ADC Enable Control**<br>0 = Digital ADC function Disabled.<br>1 = Digital ADC function Enabled. |
| [6] | **HPFEN** | **High Pass Filter Enable Control**<br>0 = High Pass Filter Disabled.<br>1 = High Pass Filter Enabled. |
| [5] | **Reserved** | Reserved. |
| [4] | **STEREO** | **CODEC ADC Output Channel Mode**<br>0 = Left mono mode (Right channel outputs the same signal as left channel).<br>1 = Stereo mode. |
| [3:0] | **Reserved** | Reserved. |

### CODEC DAC Control Register (CODEC_DACCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CODEC_DACCTL | 0x01 | RW | CODEC DAC Control Register | 0x00 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| DACEN | Reserved | DAC_MODE | | DAC_DMUTEL | DAC_DMUTER | Reserved | |

| Bits | Description | |
|---|---|---|
| [7] | **DACEN** | **CODEC DAC Enable Control**<br>0 = Digital DAC function Disabled.<br>1 = Digital DAC function Enabled. |
| [6] | **Reserved** | Reserved. |
| [5:4] | **DAC_MODE** | **CODEC DAC Output Stereo or Mono Signal Control**<br>00 = Output stereo signal mode.<br>01 = Output left mono signal mode. (Right channel signal is the same as left channel).<br>10 = Output right mono signal mode. (Left channel signal is the same as right channel).<br>11 = Output (left signal + right signal)/2 mono mode. |
| [3] | **DAC_DMUTEL** | **Left Channel Digital Mute Enable Control**<br>0 = Channel Digital Output Mute Disabled.<br>1 = Channel Digital Output Mute Enabled. |
| [2] | **DAC_DMUTER** | **Right Channel Digital Mute Enable Control**<br>0 = Channel Digital Output Mute Disabled.<br>1 = Channel Digital Output Mute Enabled. |
| [1:0] | **Reserved** | Reserved. |

### CODEC Control Register (CODEC_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CODEC_CTL** | 0x02 | RW | CODEC Control Register | 0x80 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| SRESET | SLAVE | Reserved | | | | | |

| Bits | Description | |
|---|---|---|
| [7] | **SRESET** | **CODEC Soft Reset Control Bit**<br>0 = Reset AD/DA filter and I$^2$S parts, except I$^2$C block.<br>1 = Normal operation. |
| [6] | **SLAVE** | **I$^2$S Interface Mode Selection**<br>0 = I$^2$S interface is in Master mode.<br>1 = I$^2$S interface is in Slave mode. |
| [5:0] | **Reserved** | Reserved. |

### CODEC DAC Left Channel Volume Control Register (CODEC_DACLVCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CODEC_DACLVCTL** | 0x03 | RW | CODEC DAC Left Channel Volume Control Register | 0x80 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **DACVOLL** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [7:0] | **DACVOLL** | **DAC Left Channel Volume Control Register**<br>The value can be programmed from 0x00 to 0x80, and the step is 1/128 * (Voltage Full Scale).<br>**Note:** Default setting is 0x80 (Max Volume). |

### CODEC DAC Right Channel Volume Control Register (CODEC_DACRVCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CODEC_DACRVCTL** | 0x04 | RW | CODEC DAC Right Channel Volume Control Register | 0x80 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **DACVOLR** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [7:0] | **DACVOLR** | **DAC Right Channel Volume Control Register**<br>The value can be programmed from 0x00 to 0x80, and the step is 1/128 * (Voltage Full Scale).<br>**Note:** Default setting is 0x80 (Max Volume). |

### CODEC DAC Left Channel Analog Volume Control Register (CODEC_DACLVACTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CODEC_DACLVACTL** | 0x08 | RW | CODEC DAC Left Channel Analog Volume Control Register | 0x00 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | **AVOLL** | | | | |

| Bits | Description | |
|---|---|---|
| [7:5] | **Reserved** | Reserved. |
| [4:0] | **AVOLL** | **DAC Left Channel Analog Volume Control Register**<br>The value can be programmed from 0x00 to 0x1F, and the step is -2dB.<br>0x00 =  0 dB.<br>0x01 = -2 dB.<br>0x02 = -4 dB.<br>…<br>0x1E = -60 dB.<br>0x1F = Analog Mute. |

## CODEC DAC Right Channel Analog Volume Control Register (CODEC_DACRVACTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CODEC_DACRVACTL** | 0x09 | RW | CODEC DAC Right Channel Analog Volume Control Register | 0x00 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | **AVOLR** | | | | |

| Bits | Description | |
|---|---|---|
| [7:5] | **Reserved** | Reserved. |
| [4:0] | **AVOLR** | **DAC Right Channel Analog Volume Control Register**<br>The value can be programmed from 0x00 to 0x1F, and the step is -2dB.<br>0x00 =  0 dB.<br>0x01 = -2 dB.<br>0x02 = -4 dB.<br>…<br>0x1E = -60 dB.<br>0x1F = Analog Mute. |

## CODEC DAC Output Channel Analog Mix Control Register (CODEC_DACMIXCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CODEC_DACMIXCTL | 0x0A | RW | CODEC DAC Output Channel Analog Mix Control Register | 0x09 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | MIXSELL | | | MIXSELR | | |

| Bits | Description | |
|---|---|---|
| [7:6] | **Reserved** | Reserved. |
| [5:3] | **MIXSELL** | **DAC Output Left Channel Analog Mix Control**<br>000 = No signal selected.<br>001 = Left DAC line-out signal selected.<br>010 = Left analog input signal selected.<br>100 = ADC MIC signal selected.<br>011 = Left DAC line-out and left analog input signal mixer.<br>110 = Left analog input and ADC MIC signal mixer.<br>101 = Left DAC line-out and ADC MIC signal mixer.<br>111 = Left DAC line-out, analog input, and ADC MIC signal mixer.<br>**Note:** Left analog input signal can be set from LLineIn or MIC. |
| [2:0] | **MIXSELR** | **DAC Output Right Channel Analog Mix Control**<br>000 = No signal selected.<br>001 = Right DAC line-out signal selected.<br>010 = Right analog input signal selected.<br>100 = ADC MIC signal selected.<br>011 = Right DAC line-out and right analog input signal mixer.<br>110 = Right analog input and ADC MIC signal mixer.<br>101 = Right DAC line-out and ADC MIC signal mixer.<br>111 = Right DAC line-out, right analog input, and ADC MIC signal mixer.<br>**Note:** Right analog input signal can be set from RLineIn or MIC. |

### CODEC DAC Power Down Control Register of Analog Blocks (CODEC_DACPDCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CODEC_DACPDCTL | 0x0B | RW | CODEC DAC Power Down Control Register. | 0xFF |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | HPROI | PDCHG | PWDNL | PWDNR | PDPAL | PDPAR |

| Bits | Description | |
|---|---|---|
| [7:6] | **Reserved** | Reserved. |
| [5] | **HPROI** | **Headphone Output Resistance Control Register**<br>0 = The impedance of headphone out is 1 KΩ.<br>1 = The impedance of headphone out is 40 KΩ.<br>**Note:** The bit is useful when headphone analog block is Power-down. |
| [4] | **PDCHG** | **VMID Pre-charge Capacitance Control**<br>0 = VMID Pre-Charge capacitance Disabled.<br>1 = VMID Pre-Charge capacitance Enabled. |
| [3] | **PWDNL** | **DAC Left Channel Analog Block Power Control**<br>0 = Channel power Enabled.<br>1 = Channel power Disabled. |
| [2] | **PWDNR** | **DAC Right Channel Analog Block Power Control**<br>0 = Channel power Enabled.<br>1 = Channel power Disabled. |
| [1] | **PDPAL** | **DAC Left Headphone Out Analog Block Power Control**<br>0 = Headphone analog block power Enabled.<br>1 = Headphone analog block power Disabled. |
| [0] | **PDPAR** | **DAC Right Headphone Out Analog Block Power Control**<br>0 = Headphone analog block power Enabled.<br>1 = Headphone analog block power Disabled. |

### CODEC Power Control Register (CODEC_PWRCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CODEC_PWRCTL** | 0x0D | RW | CODEC Power Control Register | 0x3A |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | **PDBIAS** | **RESADJ** | | |

| Bits | Description | |
|---|---|---|
| [7:4] | **Reserved** | Reserved. |
| [3] | **PDBIAS** | **CODEC Reference (MIC_BIAS) Power Control**<br>0 = Reference power Enabled.<br>1 = Reference power Disabled. |
| [2:0] | **RESADJ** | **Current Biasing Resistor Selection**<br>000 = Smallest biasing resistor.<br>001 = Biggest biasing resistor.<br>010 = Medium big biasing resistor.<br>100 = Medium small biasing resistor.<br>Other = Reserved. |

### CODEC ADC Input Control Register (CODEC_ADCINCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CODEC_ADCINCTL** | 0x0E | RW | CODEC ADC Input Control Register | 0x00 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | **MIC_SEL** | **ADCINSEL** | |

| Bits | Description | |
|---|---|---|
| [7:3] | **Reserved** | Reserved. |
| [2] | **MIC_SEL** | **Input Microphone Source Selection**<br>0 = Input signal is from MIC0.<br>1 = Input signal is from MIC1. |
| [1:0] | **ADCINSEL** | **Analog Input Signal Source Selection**<br>00 = Analog input signal is from line input.<br>10 = Analog input signal is from microphone input.<br>Other = Reserved. |

### CODEC ADC Power Down Control Register (CODEC_ADCPDCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| CODEC_ADCPDCTL | 0x0F | RW | CODEC ADC Power Down Control Register | 0xFF |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Reserved | | PDMIC | PDSTO | PDPGAL | PDPGAR | PDL | PDR |

| Bits | Description | |
|---|---|---|
| [7:6] | **Reserved** | Reserved. |
| [5] | **PDMIC** | **Bias Current of Microphone Power Control and Right Line-in Channel Selection**<br>0 = MIC-BIAS pin is for mic-bias function, and bias current power Enabled.<br>1 = MIC-BIAS pin is for right line-in function, and bias current power Disabled. |
| [4] | **PDSTO** | **Sidetone Analog Gain Block Power Control**<br>0 = Gain block power Enabled.<br>1 = Gain block power Disabled. |
| [3] | **PDPGAL** | **Left Channel Analog Gain Block Power Control Register**<br>0 = Gain block power Enabled.<br>1 = Gain block power Disabled. |
| [2] | **PDPGAR** | **Right Channel Analog Gain Block Power Control Register**<br>0 = Gain block power Enabled.<br>1 = Gain block power Disabled. |
| [1] | **PDL** | **Left Channel Sigma-delta Module Power Control**<br>0 = Sigma-Delta Module power Enabled.<br>1 = Sigma-Delta Module power Disabled. |
| [0] | **PDR** | **Right Channel Sigma-delta Module Power Control**<br>0 = Sigma-Delta Module power Enabled.<br>1 = Sigma-Delta Module power Disabled. |

## CODEC ADC Left Channel Gain Control Register (CODEC_ADCLVCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CODEC_ADCLVCTL** | 0x10 | RW | CODEC ADC Left Channel Gain Control Register | 0x00 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | **PREVOL** | **ADC_VOLL** | | | |

| Bits | Description | |
|---|---|---|
| [7:5] | **Reserved** | Reserved. |
| [4] | **PREVOL** | **Microphone Input Signal Pre-gain Boost Control Register**<br>0 = Pre-Gain is 0 dB.<br>1 = Pre-Gain is 20 dB. |
| [3:0] | **ADC_VOLL** | **ADC Left Channel Gain Control**<br>The value can be programmed from 0x0 to 0xF, and the step is 1.6dB.<br>0x0 = 0 dB.<br>0x1 = 1.6 dB.<br>0x2 = 3.2 dB.<br>…<br>0xE = 22.4 dB.<br>0xF = Volume Mute. |

### CODEC ADC Right Channel Gain Control Register (CODEC_ADCRVCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CODEC_ADCRVCTL** | 0x11 | RW | CODEC ADC Right Channel Gain Control Register | 0x00 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | **ADC_VOLR** | | | |

| Bits | Description | |
|---|---|---|
| [7:4] | **Reserved** | Reserved. |
| [3:0] | **ADC_VOLR** | **ADC Right Channel Gain Control**<br>The value can be programmed from 0x0 to 0xF, and the step is 1.6dB.<br>0x0 = 0 dB.<br>0x1 = 1.6 dB.<br>0x2 = 3.2 dB.<br>…<br>0xE = 22.4 dB.<br>0xF = Volume Mute. |

### CODEC ADC Sidetone Volume Control Register (CODEC_STOCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **CODEC_STOCTL** | 0x12 | RW | CODEC ADC Sidetone Volume Control Register | 0x00 |

| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | **STO_VOL** | | | |

| Bits | Description | |
|---|---|---|
| [7:4] | **Reserved** | Reserved. |
| [3:0] | **STO_VOL** | **ADC Sidetone Volume Control**<br>The value can be programmed from 0x0 to 0xF, and the step is -1.6dB.<br>0x0 = 0 dB.<br>0x1 = -1.6 dB.<br>0x2 = -3.2 dB.<br>…<br>0xE = -22.4 dB.<br>0xF = Volume Mute. |

## 6.15 USB 2.0 Device Controller (USBD)

### 6.15.1 Overview

The USB device controller interfaces the AHB bus and the UTMI bus. The USB controller contains both the AHB master interface and AHB slave interface. CPU programs the USB controller registers through the AHB slave interface. For IN or OUT transfer, the USB device controller needs to write data to memory or read data from memory through the AHB master interface. The USB device controller is complaint with USB 2.0 specification and it contains 12 configurable endpoints in addition to control endpoint. These endpoints could be configured to BULK, INTERRUPT or ISO. The USB device controller has a built-in DMA to relieve the load of CPU.

### 6.15.2 Features

- USB Specification reversion 2.0 compliant
- Supports 12 configurable endpoints in addition to Control Endpoint
- Each of the endpoints can be Isochronous, Bulk or Interrupt and either IN or OUT direction
- Three different operation modes of an in-endpoint － Auto Validation mode, Manual Validation mode, Fly mode
- Supports DMA operation
- 2048 Bytes Configurable RAM used as endpoint buffer
- Supports Endpoint Maximum Packet Size up to 1024 bytes

### 6.15.3 Block Diagram

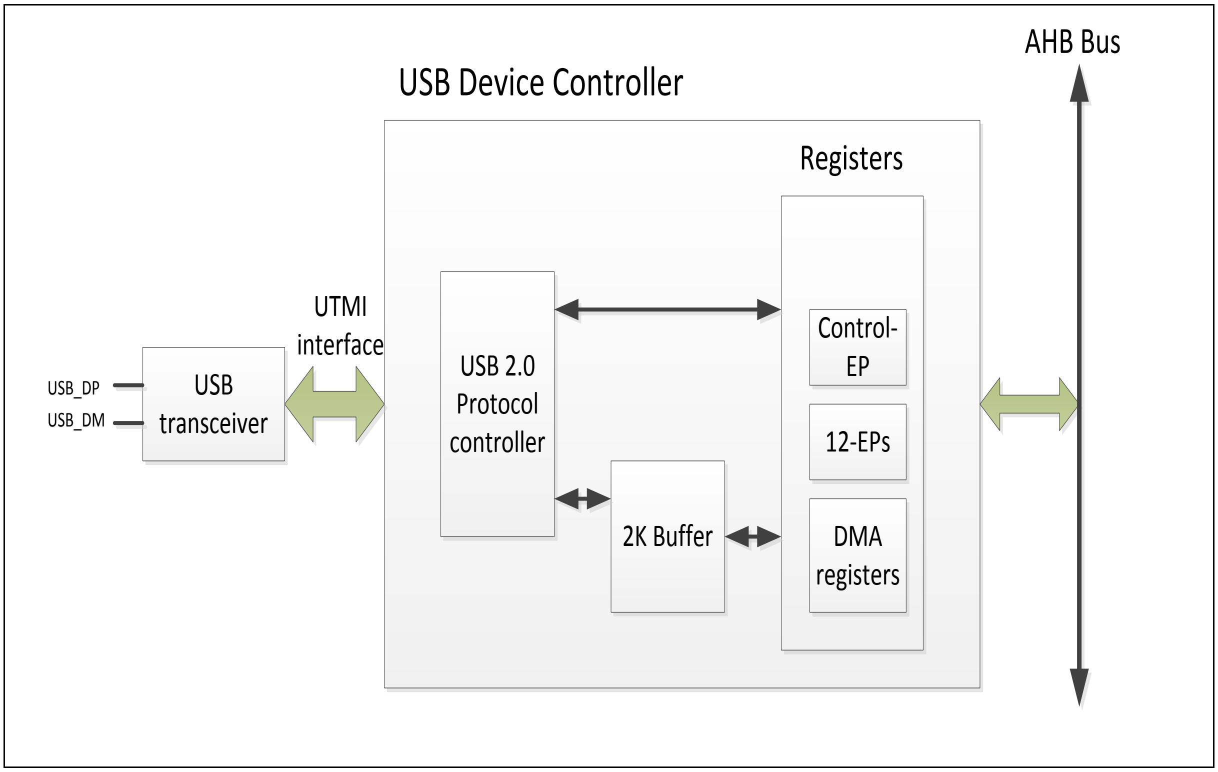


Figure 6.15-1 USB Device Controller Block Diagram

### 6.15.4 Functional Description

#### *6.15.4.1 Operation of different In-transfer modes*

The data for any in-transfer is written into the internal buffer when in turn is sent to the host on receipt of an in-token. There are three different modes by which the data sent to the host is validated by CPU.

- Auto-Validation Mode
- Manual-Validation Mode
- Fly Mode

#### *6.15.4.2 Auto-Validation Mode*

If an endpoint is selected to be operating in auto-validation mode, the endpoint responds only with data payload to be equal to EPMPS register. The endpoint controller wait until the amount of data is equal to EPMPS value and then validates the data. If CPU needs to send a short-packet at the end of a transfer, the SHORTTXEN bit of USBD_EPxRSPCTL[6] should be set. When this bit set, any remaining data in the buffer is validated and is sent to the host, for the forthcoming in-token.

This mode requires least intervention of CPU, as most of the work is done by the USB device controller. The mode can be selected, when the data payload sent to host is always equal to MPS size.

| SHORTTXEN | Data Availability In Buffer | Data Sent/NAK Sent |
|---|---|---|
| 0 | < Max. Packet Size | NAK sent |
| 0 | >= Max. Packet Size | Data payload of max. packet size |
| 1 | < Max. Packet Size | Available data of < max. packet size |
| 1 | >= Max. Packet Size | Data payload of max. packet size sent |

#### *6.15.4.3 Manual-Validation Mode*

If the endpoint is selected to be operating in manual-validation mode, the endpoint responds only when the data in the buffer is validated by CPU every time. The CPU has to write data into the buffer and then write the count of the data into EPxTXCNT register. Once the data is validating by writing a count into the EPxTXCNT register, the data is sent to the host on receipt of an in-token.

This mode requires intervention of CPU for each transfer. But this would be useful, if the data-count to be sent each time is not fixed, and it is being decided by CPU.

| EPxTXCNT Written | Data Availability In Buffer | Data Sent/NAK Sent |
|---|---|---|
| NO | - | NAK |
| YES | EPxTXCNT | Data payload of EPxTXCNT sent |

#### *6.15.4.4 Fly Mode*

The fly mode is simplest mode of operation, where there is no validation procedure. The buffer is being filled by CPU. If an in-token is sent from the host, the data in the buffer is automatically validated and sent to the host. If the data in the buffer spans more than one packet of maximum packet size, the controller automatically packs to equal to MPS and send it to the host.

This mode requires the least intervention by CPU. This mode is best suited for isochronous data transfer, where the speed of data transfer is more important than the packet size.

| Data Availability In Buffer | Data Sent |
|---|---|
| < Max. Packet Size | Data available sent |
| >= Max. Packet Size | |

#### *6.15.4.5 Scatter-Gather function*

When the scatter gather DMA function is enabled, SGEN is set high and USBD_DMACNT is set to 8 bytes, and DMA will be enabled to fetch the descriptor which describes the real memory address and length. The descriptor will be an 8-byte format, like the following:

| | Format | | | |
|---|---|---|---|---|
| | [31] | [30] | [29:0] | |
| Word0 | MEM_ADDR[31:0] | | | |
| Word1 | EOT | RD | Reserved | Count[19:0] |

**MEM_ADDR**: It specifies the memory address (AHB address).

**EOT**: The end of transfer. When this bit is set to high, it means this is the last descriptor.

**RD**: “1” means read from memory into buffer. “0” means read from buffer into memory.

### 6.15.5 Registers Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD Base Address:**<br>**USBD_BA = 0x4001_9000** | | | | |
| **USBD_GINTSTS** | USBD_BA+0x000 | R | Global Interrupt Status Register | 0x0000_0000 |
| **USBD_GINTEN** | USBD_BA+0x008 | R/W | Global Interrupt Enable Register | 0x0000_0001 |
| **USBD_BUSINTSTS** | USBD_BA+0x010 | R/W | USB Bus Interrupt Status Register | 0x0000_0000 |
| **USBD_BUSINTEN** | USBD_BA+0x014 | R/W | USB Bus Interrupt Enable Register | 0x0000_0040 |
| **USBD_OPER** | USBD_BA+0x018 | R/W | USB Operational Register | 0x0000_0002 |
| **USBD_FRAMECNT** | USBD_BA+0x01C | R | USB Frame Count Register | 0x0000_0000 |
| **USBD_FADDR** | USBD_BA+0x020 | R/W | USB Function Address Register | 0x0000_0000 |
| **USBD_TEST** | USBD_BA+0x024 | R/W | USB Test Mode Register | 0x0000_0000 |
| **USBD_CEPDAT** | USBD_BA+0x028 | R/W | Control-Endpoint Data Buffer | 0x0000_0000 |
| **USBD_CEPCTL** | USBD_BA+0x02C | R/W | Control-Endpoint Control Register | 0x0000_0000 |
| **USBD_CEPINTEN** | USBD_BA+0x030 | R/W | Control-Endpoint Interrupt Enable Control Register | 0x0000_0000 |
| **USBD_CEPINTSTS** | USBD_BA+0x034 | R/W | Control-Endpoint Interrupt Status | 0x0000_1800 |
| **USBD_CEPTXCNT** | USBD_BA+0x038 | R/W | Control-Endpoint In-transfer Data Count | 0x0000_0000 |
| **USBD_CEPRXCNT** | USBD_BA+0x03C | R | Control-Endpoint Out-transfer Data Count | 0x0000_0000 |
| **USBD_CEPDATCNT** | USBD_BA+0x040 | R | Control-Endpoint Data Count | 0x0000_0000 |
| **USBD_SETUP1_0** | USBD_BA+0x044 | R | Setup1 & Setup0 bytes | 0x0000_0000 |
| **USBD_SETUP3_2** | USBD_BA+0x048 | R | Setup3 & Setup2 Bytes | 0x0000_0000 |
| **USBD_SETUP5_4** | USBD_BA+0x04C | R | Setup5 & Setup4 Bytes | 0x0000_0000 |
| **USBD_SETUP7_6** | USBD_BA+0x050 | R | Setup7 & Setup6 Bytes | 0x0000_0000 |
| **USBD_CEPBUFSTART** | USBD_BA+0x054 | R/W | Control Endpoint RAM Start Address Register | 0x0000_0000 |
| **USBD_CEPBUFEND** | USBD_BA+0x058 | R/W | Control Endpoint RAM End Address Register | 0x0000_0000 |
| **USBD_DMACTL** | USBD_BA+0x05C | R/W | DMA Control Status Register | 0x0000_0000 |
| **USBD_DMACNT** | USBD_BA+0x060 | R/W | DMA Count Register | 0x0000_0000 |
| **USBD_EPADAT** | USBD_BA+0x064 | R/W | Endpoint A Data Register | 0x0000_0000 |
| **USBD_EPAINTSTS** | USBD_BA+0x068 | R/W | Endpoint A Interrupt Status Register | 0x0000_0003 |
| **USBD_EPAINTEN** | USBD_BA+0x06C | R/W | Endpoint A Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPADATCNT** | USBD_BA+0x070 | R | Endpoint A Data Available Count Register | 0x0000_0000 |
| **USBD_EPARSPCTL** | USBD_BA+0x074 | R/W | Endpoint A Response Control Register | 0x0000_0000 |
| **USBD_EPAMPS** | USBD_BA+0x078 | R/W | Endpoint A Maximum Packet Size Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **USBD_EPATXCNT** | USBD_BA+0x07C | R/W | Endpoint A Transfer Count Register | 0x0000_0000 |
| **USBD_EPACFG** | USBD_BA+0x080 | R/W | Endpoint A Configuration Register | 0x0000_0012 |
| **USBD_EPABUFSTART** | USBD_BA+0x084 | R/W | Endpoint A RAM Start Address Register | 0x0000_0000 |
| **USBD_EPABUFEND** | USBD_BA+0x088 | R/W | Endpoint A RAM End Address Register | 0x0000_0000 |
| **USBD_EPBDAT** | USBD_BA+0x08C | R/W | Endpoint B Data Register | 0x0000_0000 |
| **USBD_EPBINTSTS** | USBD_BA+0x090 | R/W | Endpoint B Interrupt Status Register | 0x0000_0003 |
| **USBD_EPBINTEN** | USBD_BA+0x094 | R/W | Endpoint B Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPBDATCNT** | USBD_BA+0x098 | R | Endpoint B Data Available Count Register | 0x0000_0000 |
| **USBD_EPBRSPCTL** | USBD_BA+0x09C | R/W | Endpoint B Response Control Register | 0x0000_0000 |
| **USBD_EPBMPS** | USBD_BA+0x0A0 | R/W | Endpoint B Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPBTXCNT** | USBD_BA+0x0A4 | R/W | Endpoint B Transfer Count Register | 0x0000_0000 |
| **USBD_EPBCFG** | USBD_BA+0x0A8 | R/W | Endpoint B Configuration Register | 0x0000_0022 |
| **USBD_EPBBUFSTART** | USBD_BA+0x0AC | R/W | Endpoint B RAM Start Address Register | 0x0000_0000 |
| **USBD_EPBBUFEND** | USBD_BA+0x0B0 | R/W | Endpoint B RAM End Address Register | 0x0000_0000 |
| **USBD_EPCDAT** | USBD_BA+0x0B4 | R/W | Endpoint C Data Register | 0x0000_0000 |
| **USBD_EPCINTSTS** | USBD_BA+0x0B8 | R/W | Endpoint C Interrupt Status Register | 0x0000_0003 |
| **USBD_EPCINTEN** | USBD_BA+0x0BC | R/W | Endpoint C Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPCDATCNT** | USBD_BA+0x0C0 | R | Endpoint C Data Available Count Register | 0x0000_0000 |
| **USBD_EPCRSPCTL** | USBD_BA+0x0C4 | R/W | Endpoint C Response Control Register | 0x0000_0000 |
| **USBD_EPCMPS** | USBD_BA+0x0C8 | R/W | Endpoint C Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPCTXCNT** | USBD_BA+0x0CC | R/W | Endpoint C Transfer Count Register | 0x0000_0000 |
| **USBD_EPCCFG** | USBD_BA+0x0D0 | R/W | Endpoint C Configuration Register | 0x0000_0032 |
| **USBD_EPCBUFSTART** | USBD_BA+0x0D4 | R/W | Endpoint C RAM Start Address Register | 0x0000_0000 |
| **USBD_EPCBUFEND** | USBD_BA+0x0D8 | R/W | Endpoint C RAM End Address Register | 0x0000_0000 |
| **USBD_EPDDAT** | USBD_BA+0x0DC | R/W | Endpoint D Data Register | 0x0000_0000 |
| **USBD_EPDINTSTS** | USBD_BA+0x0E0 | R/W | Endpoint D Interrupt Status Register | 0x0000_0003 |
| **USBD_EPDINTEN** | USBD_BA+0x0E4 | R/W | Endpoint D Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPDDATCNT** | USBD_BA+0x0E8 | R | Endpoint D Data Available Count Register | 0x0000_0000 |
| **USBD_EPDRSPCTL** | USBD_BA+0x0EC | R/W | Endpoint D Response Control Register | 0x0000_0000 |
| **USBD_EPDMPS** | USBD_BA+0x0F0 | R/W | Endpoint D Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPDTXCNT** | USBD_BA+0x0F4 | R/W | Endpoint D Transfer Count Register | 0x0000_0000 |
| **USBD_EPDCFG** | USBD_BA+0x0F8 | R/W | Endpoint D Configuration Register | 0x0000_0042 |
| **USBD_EPDBUFSTART** | USBD_BA+0x0FC | R/W | Endpoint D RAM Start Address Register | 0x0000_0000 |
| **USBD_EPDBUFEND** | USBD_BA+0x100 | R/W | Endpoint D RAM End Address Register | 0x0000_0000 |

NUC505 SERIES TECHNICAL REFERENCE MANUAL

| | | | | |
|---|---|---|---|---|
| **USBD_EPEDAT** | USBD_BA+0x104 | R/W | Endpoint E Data Register | 0x0000_0000 |
| **USBD_EPEINTSTS** | USBD_BA+0x108 | R/W | Endpoint E Interrupt Status Register | 0x0000_0003 |
| **USBD_EPEINTEN** | USBD_BA+0x10C | R/W | Endpoint E Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPEDATCNT** | USBD_BA+0x110 | R | Endpoint E Data Available Count Register | 0x0000_0000 |
| **USBD_EPERSPCTL** | USBD_BA+0x114 | R/W | Endpoint E Response Control Register | 0x0000_0000 |
| **USBD_EPEMPS** | USBD_BA+0x118 | R/W | Endpoint E Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPETXCNT** | USBD_BA+0x11C | R/W | Endpoint E Transfer Count Register | 0x0000_0000 |
| **USBD_EPECFG** | USBD_BA+0x120 | R/W | Endpoint E Configuration Register | 0x0000_0052 |
| **USBD_EPEBUFSTART** | USBD_BA+0x124 | R/W | Endpoint E RAM Start Address Register | 0x0000_0000 |
| **USBD_EPEBUFEND** | USBD_BA+0x128 | R/W | Endpoint E RAM End Address Register | 0x0000_0000 |
| **USBD_EPFDAT** | USBD_BA+0x12C | R/W | Endpoint F Data Register | 0x0000_0000 |
| **USBD_EPFINTSTS** | USBD_BA+0x130 | R/W | Endpoint F Interrupt Status Register | 0x0000_0003 |
| **USBD_EPFINTEN** | USBD_BA+0x134 | R/W | Endpoint F Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPFDATCNT** | USBD_BA+0x138 | R | Endpoint F Data Available Count Register | 0x0000_0000 |
| **USBD_EPFRSPCTL** | USBD_BA+0x13C | R/W | Endpoint F Response Control Register | 0x0000_0000 |
| **USBD_EPFMPS** | USBD_BA+0x140 | R/W | Endpoint F Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPFTXCNT** | USBD_BA+0x144 | R/W | Endpoint F Transfer Count Register | 0x0000_0000 |
| **USBD_EPFCFG** | USBD_BA+0x148 | R/W | Endpoint F Configuration Register | 0x0000_0062 |
| **USBD_EPFBUFSTART** | USBD_BA+0x14C | R/W | Endpoint F RAM Start Address Register | 0x0000_0000 |
| **USBD_EPFBUFEND** | USBD_BA+0x150 | R/W | Endpoint F RAM End Address Register | 0x0000_0000 |
| **USBD_EPGDAT** | USBD_BA+0x154 | R/W | Endpoint G Data Register | 0x0000_0000 |
| **USBD_EPGINTSTS** | USBD_BA+0x158 | R/W | Endpoint G Interrupt Status Register | 0x0000_0003 |
| **USBD_EPGINTEN** | USBD_BA+0x15C | R/W | Endpoint G Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPGDATCNT** | USBD_BA+0x160 | R | Endpoint G Data Available Count Register | 0x0000_0000 |
| **USBD_EPGRSPCTL** | USBD_BA+0x164 | R/W | Endpoint G Response Control Register | 0x0000_0000 |
| **USBD_EPGMPS** | USBD_BA+0x168 | R/W | Endpoint G Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPGTXCNT** | USBD_BA+0x16C | R/W | Endpoint G Transfer Count Register | 0x0000_0000 |
| **USBD_EPGCFG** | USBD_BA+0x170 | R/W | Endpoint G Configuration Register | 0x0000_0072 |
| **USBD_EPGBUFSTART** | USBD_BA+0x174 | R/W | Endpoint G RAM Start Address Register | 0x0000_0000 |
| **USBD_EPGBUFEND** | USBD_BA+0x178 | R/W | Endpoint G RAM End Address Register | 0x0000_0000 |
| **USBD_EPHDAT** | USBD_BA+0x17C | R/W | Endpoint H Data Register | 0x0000_0000 |
| **USBD_EPHINTSTS** | USBD_BA+0x180 | R/W | Endpoint H Interrupt Status Register | 0x0000_0003 |
| **USBD_EPHINTEN** | USBD_BA+0x184 | R/W | Endpoint H Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPHDATCNT** | USBD_BA+0x188 | R | Endpoint H Data Available Count Register | 0x0000_0000 |

| | | | | |
|---|---|---|---|---|
| **USBD_EPHRSPCTL** | USBD_BA+0x18C | R/W | Endpoint H Response Control Register | 0x0000_0000 |
| **USBD_EPHMPS** | USBD_BA+0x190 | R/W | Endpoint H Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPHTXCNT** | USBD_BA+0x194 | R/W | Endpoint H Transfer Count Register | 0x0000_0000 |
| **USBD_EPHCFG** | USBD_BA+0x198 | R/W | Endpoint H Configuration Register | 0x0000_0082 |
| **USBD_EPHBUFSTART** | USBD_BA+0x19C | R/W | Endpoint H RAM Start Address Register | 0x0000_0000 |
| **USBD_EPHBUFEND** | USBD_BA+0x1A0 | R/W | Endpoint H RAM End Address Register | 0x0000_0000 |
| **USBD_EPIDAT** | USBD_BA+0x1A4 | R/W | Endpoint I Data Register | 0x0000_0000 |
| **USBD_EPIINTSTS** | USBD_BA+0x1A8 | R/W | Endpoint I Interrupt Status Register | 0x0000_0003 |
| **USBD_EPIINTEN** | USBD_BA+0x1AC | R/W | Endpoint I Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPIDATCNT** | USBD_BA+0x1B0 | R | Endpoint I Data Available Count Register | 0x0000_0000 |
| **USBD_EPIRSPCTL** | USBD_BA+0x1B4 | R/W | Endpoint I Response Control Register | 0x0000_0000 |
| **USBD_EPIMPS** | USBD_BA+0x1B8 | R/W | Endpoint I Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPITXCNT** | USBD_BA+0x1BC | R/W | Endpoint I Transfer Count Register | 0x0000_0000 |
| **USBD_EPICFG** | USBD_BA+0x1C0 | R/W | Endpoint I Configuration Register | 0x0000_0092 |
| **USBD_EPIBUFSTART** | USBD_BA+0x1C4 | R/W | Endpoint I RAM Start Address Register | 0x0000_0000 |
| **USBD_EPIBUFEND** | USBD_BA+0x1C8 | R/W | Endpoint I RAM End Address Register | 0x0000_0000 |
| **USBD_EPJDAT** | USBD_BA+0x1CC | R/W | Endpoint J Data Register | 0x0000_0000 |
| **USBD_EPJINTSTS** | USBD_BA+0x1D0 | R/W | Endpoint J Interrupt Status Register | 0x0000_0003 |
| **USBD_EPJINTEN** | USBD_BA+0x1D4 | R/W | Endpoint J Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPJDATCNT** | USBD_BA+0x1D8 | R | Endpoint J Data Available Count Register | 0x0000_0000 |
| **USBD_EPJRSPCTL** | USBD_BA+0x1DC | R/W | Endpoint J Response Control Register | 0x0000_0000 |
| **USBD_EPJMPS** | USBD_BA+0x1E0 | R/W | Endpoint J Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPJTXCNT** | USBD_BA+0x1E4 | R/W | Endpoint J Transfer Count Register | 0x0000_0000 |
| **USBD_EPJCFG** | USBD_BA+0x1E8 | R/W | Endpoint J Configuration Register | 0x0000_00A2 |
| **USBD_EPJBUFSTART** | USBD_BA+0x1EC | R/W | Endpoint J RAM Start Address Register | 0x0000_0000 |
| **USBD_EPJBUFEND** | USBD_BA+0x1F0 | R/W | Endpoint J RAM End Address Register | 0x0000_0000 |
| **USBD_EPKDAT** | USBD_BA+0x1F4 | R/W | Endpoint K Data Register | 0x0000_0000 |
| **USBD_EPKINTSTS** | USBD_BA+0x1F8 | R/W | Endpoint K Interrupt Status Register | 0x0000_0003 |
| **USBD_EPKINTEN** | USBD_BA+0x1FC | R/W | Endpoint K Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPKDATCNT** | USBD_BA+0x200 | R | Endpoint K Data Available Count Register | 0x0000_0000 |
| **USBD_EPKRSPCTL** | USBD_BA+0x204 | R/W | Endpoint K Response Control Register | 0x0000_0000 |
| **USBD_EPKMPS** | USBD_BA+0x208 | R/W | Endpoint K Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPKTXCNT** | USBD_BA+0x20C | R/W | Endpoint K Transfer Count Register | 0x0000_0000 |
| **USBD_EPKCFG** | USBD_BA+0x210 | R/W | Endpoint K Configuration Register | 0x0000_00B2 |

| | | | | |
|---|---|---|---|---|
| **USBD_EPKBUFSTART** | USBD_BA+0x214 | R/W | Endpoint K RAM Start Address Register | 0x0000_0000 |
| **USBD_EPKBUFEND** | USBD_BA+0x218 | R/W | Endpoint K RAM End Address Register | 0x0000_0000 |
| **USBD_EPLDAT** | USBD_BA+0x21C | R/W | Endpoint L Data Register | 0x0000_0000 |
| **USBD_EPLINTSTS** | USBD_BA+0x220 | R/W | Endpoint L Interrupt Status Register | 0x0000_0003 |
| **USBD_EPLINTEN** | USBD_BA+0x224 | R/W | Endpoint L Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPLDATCNT** | USBD_BA+0x228 | R | Endpoint L Data Available Count Register | 0x0000_0000 |
| **USBD_EPLRSPCTL** | USBD_BA+0x22C | R/W | Endpoint L Response Control Register | 0x0000_0000 |
| **USBD_EPLMPS** | USBD_BA+0x230 | R/W | Endpoint L Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPLTXCNT** | USBD_BA+0x234 | R/W | Endpoint L Transfer Count Register | 0x0000_0000 |
| **USBD_EPLCFG** | USBD_BA+0x238 | R/W | Endpoint L Configuration Register | 0x0000_00C2 |
| **USBD_EPLBUFSTART** | USBD_BA+0x23C | R/W | Endpoint L RAM Start Address Register | 0x0000_0000 |
| **USBD_EPLBUFEND** | USBD_BA+0x240 | R/W | Endpoint L RAM End Address Register | 0x0000_0000 |
| **USBD_UVCHDAT0** | USBD_BA+0x248 | R/W | USB Header Word0 | 0x0000_0000 |
| **USBD_UVCHDAT1** | USBD_BA+0x24C | R/W | USB Header Word1 | 0x0000_0000 |
| **USBD_UVCHDAT2** | USBD_BA+0x250 | R/W | USB Header Word2 | 0x0000_0000 |
| **USBD_UVCEPAHCNT** | USBD_BA+0x254 | R/W | Endpoint A Header Count | 0x0000_0000 |
| **USBD_UVCEPBHCNT** | USBD_BA+0x258 | R/W | Endpoint B Header Count | 0x0000_0000 |
| **USBD_UVCEPCHCNT** | USBD_BA+0x25C | R/W | Endpoint C Header Count | 0x0000_0000 |
| **USBD_UVCEPDHCNT** | USBD_BA+0x260 | R/W | Endpoint D Header Count | 0x0000_0000 |
| **USBD_UVCEPEHCNT** | USBD_BA+0x264 | R/W | Endpoint E Header Count | 0x0000_0000 |
| **USBD_UVCEPFHCNT** | USBD_BA+0x268 | R/W | Endpoint F Header Count | 0x0000_0000 |
| **USBD_UVCEPGHCNT** | USBD_BA+0x26C | R/W | Endpoint G Header Count | 0x0000_0000 |
| **USBD_UVCEPHHCNT** | USBD_BA+0x270 | R/W | Endpoint H Header Count | 0x0000_0000 |
| **USBD_UVCEPIHCNT** | USBD_BA+0x274 | R/W | Endpoint I Header Count | 0x0000_0000 |
| **USBD_UVCEPJHCNT** | USBD_BA+0x278 | R/W | Endpoint J Header Count | 0x0000_0000 |
| **USBD_UVCEPKHCNT** | USBD_BA+0x27C | R/W | Endpoint K Header Count | 0x0000_0000 |
| **USBD_UVCEPLHCNT** | USBD_BA+0x280 | R/W | Endpoint L Header Count | 0x0000_0000 |
| **USBD_DMAADDR** | USBD_BA+0x700 | R/W | AHB DMA Address Register | 0x0000_0000 |
| **USBD_PHYCTL** | USBD_BA+0x704 | R/W | USB PHY Control Register | 0x0000_0420 |

### 6.15.6 Register Description

#### Global Interrupt Status Register (USBD_GINTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_GINTSTS** | USBD_BA+0x000 | R | Global Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| Reserved | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | | EPLIF | EPKIF | EPJIF | EPIIF | EPHIF | EPGIF |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| EPFIF | EPEIF | EPDIF | EPCIF | EPBIF | EPAIF | CEPIF | USBIF |

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13] | **EPLIF** | **Endpoints L Interrupt**<br>When set, the corresponding Endpoint L's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [12] | **EPKIF** | **Endpoints K Interrupt**<br>When set, the corresponding Endpoint K's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [11] | **EPJIF** | **Endpoints J Interrupt**<br>When set, the corresponding Endpoint J's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [10] | **EPIIF** | **Endpoints I Interrupt**<br>When set, the corresponding Endpoint I's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |

| | | |
|---|---|---|
| [9] | **EPHIF** | **Endpoints H Interrupt**<br>When set, the corresponding Endpoint H’s interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [8] | **EPGIF** | **Endpoints G Interrupt**<br>When set, the corresponding Endpoint G's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [7] | **EPFIF** | **Endpoints F Interrupt**<br>When set, the corresponding Endpoint F's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [6] | **EPEIF** | **Endpoints E Interrupt**<br>When set, the corresponding Endpoint E's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [5] | **EPDIF** | **Endpoints D Interrupt**<br>When set, the corresponding Endpoint D's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [4] | **EPCIF** | **Endpoints C Interrupt**<br>When set, the corresponding Endpoint C's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [3] | **EPBIF** | **Endpoints B Interrupt**<br>When set, the corresponding Endpoint B's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [2] | **EPAIF** | **Endpoints a Interrupt**<br>When set, the corresponding Endpoint A's interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |
| [1] | **CEPIF** | **Control Endpoint Interrupt**<br>This bit conveys the interrupt status for control endpoint. When set, Control-ep’s interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |

| | | |
|---|---|---|
| [0] | **USBIF** | **USB Interrupt**<br>This bit conveys the interrupt status for USB specific events endpoint. When set, USB interrupt status register should be read to determine the cause of the interrupt.<br>0 = No interrupt event occurred.<br>1 = The related interrupt event is occurred. |

### Global Interrupt Enable Register (USBD_GINTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_GINTEN** | USBD_BA+0x008 | R/W | Global Interrupt Enable Register | 0x0000_0001 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | **EPLIEN** | **EPKIEN** | **EPJIEN** | **EPIIEN** | **EPHIEN** | **EPGIEN** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **EPFIEN** | **EPEIEN** | **EPDIEN** | **EPCIEN** | **EPBIEN** | **EPAIEN** | **CEPIEN** | **USBIEN** |

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13] | **EPLIEN** | **Interrupt Enable Control for Endpoint L**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint L<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [12] | **EPKIEN** | **Interrupt Enable Control for Endpoint K**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint K<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [11] | **EPJIEN** | **Interrupt Enable Control for Endpoint J**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint J<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [10] | **EPIIEN** | **Interrupt Enable Control for Endpoint I**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint I<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [9] | **EPHIEN** | **Interrupt Enable Control for Endpoint H**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint H<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |

| | | |
|---|---|---|
| [8] | **EPGIEN** | **Interrupt Enable Control for Endpoint G**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint G<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [7] | **EPFIEN** | **Interrupt Enable Control for Endpoint F**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint F<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [6] | **EPEIEN** | **Interrupt Enable Control for Endpoint E**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint E<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [5] | **EPDIEN** | **Interrupt Enable Control for Endpoint D**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint D<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [4] | **EPCIEN** | **Interrupt Enable Control for Endpoint C**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint C<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [3] | **EPBIEN** | **Interrupt Enable Control for Endpoint B**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint B<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [2] | **EPAIEN** | **Interrupt Enable Control for Endpoint a**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the endpoint A.<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [1] | **CEPIEN** | **Control Endpoint Interrupt Enable Bit**<br>When set, this bit enables a local interrupt to be generated when an interrupt is pending for the control endpoint.<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |
| [0] | **USBIEN** | **USB Interrupt Enable Bit**<br>When set, this bit enables a local interrupt to be generated when a USB event occurs on the bus.<br>0 = The related interrupt Disabled.<br>1 = The related interrupt Enabled. |

NUC505 SERIES TECHNICAL REFERENCE MANUAL

### USB Bus Interrupt Status Register (USBD_BUSINTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_BUSINTSTS** | USBD_BA+0x010 | R/W | USB Bus Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | | **VBUSDETIF** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **PHYCLKVLDIF** | **DMADONEIF** | **HISPDIF** | **SUSPENDIF** | **RESUMEIF** | **RSTIF** | **SOFIF** |

| Bits | Description | |
|---|---|---|
| [31:9] | **Reserved** | Reserved. |
| [8] | **VBUSDETIF** | **VBUS Detection Interrupt Status**<br>0 = No VBUS is plug-in.<br>1 = VBUS is plug-in.<br>Write 1 to clear this bit to 0. |
| [7] | **Reserved** | Reserved. |
| [6] | **PHYCLKVLDIF** | **Usable Clock Interrupt**<br>0 = Usable clock is not available.<br>1 = Usable clock is available from the transceiver.<br>Write 1 to clear this bit to 0. |
| [5] | **DMADONEIF** | **DMA Completion Interrupt**<br>0 = No DMA transfer over.<br>1 = DMA transfer is over.<br>Write 1 to clear this bit to 0. |
| [4] | **HISPDIF** | **High-speed Settle**<br>0 = No valid high-speed reset protocol is detected.<br>1 = Valid high-speed reset protocol is over and the device has settled in high-speed.<br>Write 1 to clear this bit to 0. |
| [3] | **SUSPENDIF** | **Suspend Request**<br>This bit is set as default and it has to be cleared by writing ‘1’ before the USB reset. This bit is also set when a USB Suspend request is detected from the host.<br>0 = No USB Suspend request is detected from the host.<br>1= USB Suspend request is detected from the host.<br>Write 1 to clear this bit to 0. |

| | | |
|---|---|---|
| [2] | **RESUMEIF** | **Resume**<br>When set, this bit indicates that a device resume has occurred.<br>0 = No device resume has occurred.<br>1 = Device resume has occurred.<br>Write 1 to clear this bit to 0. |
| [1] | **RSTIF** | **Reset Status**<br>When set, this bit indicates that either the USB root port reset is end.<br>0 = No USB root port reset is end.<br>1 = USB root port reset is end.<br>Write 1 to clear this bit to 0. |
| [0] | **SOFIF** | **SOF Receive Control**<br>This bit indicates when a start-of-frame packet has been received.<br>0 = No start-of-frame packet has been received.<br>1 = Start-of-frame packet has been received.<br>Write 1 to clear this bit to 0. |

### USB Bus Interrupt Enable Register (USBD_BUSINTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_BUSINTEN** | USBD_BA+0x014 | R/W | USB Bus Interrupt Enable Register | 0x0000_0040 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | | **VBUSDETIEN** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **PHYCLKVLDIEN** | **DMADONEIEN** | **HISPDIEN** | **SUSPENDIEN** | **RESUMEIEN** | **RSTIEN** | **SOFIEN** |

| Bits | Description | |
|---|---|---|
| [31:9] | **Reserved** | Reserved. |
| [8] | **VBUSDETIEN** | **VBUS Detection Interrupt Enable Control**<br>This bit enables the VBUS floating detection interrupt.<br>0 = VBUS floating detection interrupt Disabled.<br>1 = VBUS floating detection interrupt Enabled. |
| [7] | **Reserved** | Reserved. |
| [6] | **PHYCLKVLDIEN** | **Usable Clock Interrupt**<br>This bit enables the usable clock interrupt.<br>0 = Usable clock interrupt Disabled.<br>1 = Usable clock interrupt Enabled. |
| [5] | **DMADONEIEN** | **DMA Completion Interrupt**<br>This bit enables the DMA completion interrupt<br>0 = DMA completion interrupt Disabled.<br>1 = DMA completion interrupt Enabled. |
| [4] | **HISPDIEN** | **High-speed Settle**<br>This bit enables the high-speed settle interrupt.<br>0 = High-speed settle interrupt Disabled.<br>1 = High-speed settle interrupt Enabled. |
| [3] | **SUSPENDIEN** | **Suspend Request**<br>This bit enables the Suspend interrupt.<br>0 = Suspend interrupt Disabled.<br>1 = Suspend interrupt Enabled. |

| | | |
|---|---|---|
| [2] | **RESUMEIEN** | **Resume**<br>This bit enables the Resume interrupt.<br>0 = Resume interrupt Disabled.<br>1 = Resume interrupt Enabled. |
| [1] | **RSTIEN** | **Reset Status**<br>This bit enables the USB-Reset interrupt.<br>0 = USB-Reset interrupt Disabled.<br>1 = USB-Reset interrupt Enabled. |
| [0] | **SOFIEN** | **SOF Interrupt**<br>This bit enables the SOF interrupt.<br>0 = SOF interrupt Disabled.<br>1 = SOF interrupt Enabled. |

### USB Operational Register (USBD_OPER)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_OPER** | USBD_BA+0x018 | R/W | USB Operational Register | 0x0000_0002 |

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

| Bits | Description | |
|---|---|---|
| [31:3] | **Reserved** | Reserved. |
| [2] | **CURSPD** | **Current USB Speed**<br>0 = The device has settled in Full Speed.<br>1 = The USB device controller has settled in High Speed. |
| [1] | **HISPDEN** | **USB High-speed**<br>0 = The USB device controller suppresses the chirp-sequence during reset protocol, thereby allowing the USB device controller to settle in full-speed, even though it is connected to a USB2.0 Host.<br>1 = The USB device controller initiates a chirp-sequence during reset protocol. |
| [0] | **RESUMEEN** | **Generate Resume**<br>0 = No resume sequence to be initiated to the host.<br>1 = A resume sequence to be initiated to the host if device remote wake-up is enabled.<br>**Note:** This bit is self-cleared. |

### USB Frame Count Register (USBD_FRAMECNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_FRAMECNT** | USBD_BA+0x01C | R | USB Frame Count Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13:3] | **FRAMECNT** | **Frame Counter**<br>This field contains the frame count from the most recent start-of-frame packet. |
| [2:0] | **MFRAMECNT** | **Micro-frame Counter**<br>This field contains the micro-frame number for the frame number in the frame counter field. |

### USB Function Address Register (USBD_FADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_FADDR** | USBD_BA+0x020 | R/W | USB Function Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:7] | **Reserved** | Reserved. |
| [6:0] | **FADDR** | **USB Function Address**<br>This field contains the current USB address of the device. This field is cleared when a root port reset is detected. |

### USB Test Mode Register (USBD_TEST)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_TEST** | USBD_BA+0x024 | R/W | USB Test Mode Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:3] | **Reserved** | Reserved. |
| [2:0] | **TESTMODE** | **Test Mode Selection**<br>000 = Normal Operation.<br>001 = Test_J.<br>010 = Test_K.<br>011 = Test_SE0_NAK.<br>100 = Test_Packet.<br>101 = Test_Force_Enable.<br>110 = Reserved.<br>111 = Reserved.<br>**Note:** This field is cleared when root port reset is detected. |

### Control-Endpoint Data Buffer (USBD_CEPDAT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_CEPDAT** | USBD_BA+0x028 | R/W | Control-Endpoint Data Buffer | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **DAT** | **Control-endpoint Data Buffer**<br>Control endpoint data buffer for the buffer transaction (read or write).<br>Only word or byte access is supported. |

### Control-Endpoint Control Register (USBD_CEPCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_CEPCTL** | USBD_BA+0x02C | R/W | Control-Endpoint Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:4] | **Reserved** | Reserved. |
| [3] | **FLUSH** | **CEP-fLUSH Bit**<br>0 = No packet buffer and its corresponding USBD_CEPDATCNT register to be cleared.<br>1 = The packet buffer and its corresponding USBD_CEPDATCNT register to be cleared. This bit is self-cleared. |
| [2] | **ZEROLEN** | **Zero Packet Length**<br>This bit is valid for Auto Validation mode only.<br>0 = No zero length packet to the host during Data stage to an IN token.<br>1 = USB device controller can send a zero length packet to the host during Data stage to an IN token. This bit gets cleared once the zero length data packet is sent. Thus, the local CPU does not need to write again to clear this bit. |
| [1] | **STALLEN** | **Stall Enable Control**<br>When this stall bit is set, the control endpoint sends a stall handshake in response to any in or out token thereafter. This is typically used for response to invalid/unsupported requests. When this bit is being set the NAK clear bit has to be cleared at the same time since the NAK clear bit has highest priority than STALL. It is automatically cleared on receipt of a next setup-token. Thus, the local CPU does not need to write again to clear this bit.<br>0 = No sends a stall handshake in response to any in or out token thereafter.<br>1 = The control endpoint sends a stall handshake in response to any in or out token thereafter.<br>**Note:** This bit can be updated only when CPU writes data[1:0] 0x2 or 0x0. |

|  |  |  |
|---|---|---|
| [0] | **NAKCLR** | **No Acknowledge Control**<br>This bit plays a crucial role in any control transfer.<br>0 = The bit is being cleared by the local CPU by writing zero, the USB device controller will be responding with NAKs for the subsequent status phase. This mechanism holds the host from moving to the next request, until the local CPU is also ready to process the next request.<br>1 = This bit is set to one by the USB device controller, whenever a setup token is received. The local CPU can take its own time to finish off any house-keeping work based on the request and then clear this bit.<br>**Note:** This bit can be updated only when CPU writes data[1:0] is 0x2 or 0x0. |

### Control Endpoint Interrupt Enable Control Register (USBD_CEPINTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| USBD_CEPINTEN | USBD_BA+0x030 | R/W | Control-Endpoint Interrupt Enable Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| Reserved | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | | | BUFEMPTYIEN | BUFFULLIEN | STSDONEIEN | ERRIEN | STALLIEN |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| NAKIEN | RXPKIEN | TXPKIEN | PINGIEN | INTKIEN | OUTTKIEN | SETUPPKIEN | SETUPTKIEN |

| Bits | Description | |
|---|---|---|
| [31:13] | **Reserved** | Reserved. |
| [12] | **BUFEMPTYIEN** | **Buffer Empty Interrupt**<br>0 = The buffer empty interrupt in Control Endpoint Disabled.<br>1= The buffer empty interrupt in Control Endpoint Enabled. |
| [11] | **BUFFULLIEN** | **Buffer Full Interrupt**<br>0 = The buffer full interrupt in Control Endpoint Disabled.<br>1 = The buffer full interrupt in Control Endpoint Enabled. |
| [10] | **STSDONEIEN** | **Status Completion Interrupt**<br>0 = The Status Completion interrupt in Control Endpoint Disabled.<br>1 = The Status Completion interrupt in Control Endpoint Enabled. |
| [9] | **ERRIEN** | **USB Error Interrupt**<br>0 = The USB Error interrupt in Control Endpoint Disabled.<br>1 = The USB Error interrupt in Control Endpoint Enabled. |
| [8] | **STALLIEN** | **STALL Sent Interrupt**<br>0 = The STALL sent interrupt in Control Endpoint Disabled.<br>1 = The STALL sent interrupt in Control Endpoint Enabled. |
| [7] | **NAKIEN** | **NAK Sent Interrupt**<br>0 = The NAK sent interrupt in Control Endpoint Disabled.<br>1 = The NAK sent interrupt in Control Endpoint Enabled. |
| [6] | **RXPKIEN** | **Data Packet Received Interrupt**<br>0 = The data received interrupt in Control Endpoint Disabled.<br>1 = The data received interrupt in Control Endpoint Enabled. |

| | | |
|---|---|---|
| [5] | **TXPKIEN** | **Data Packet Transmitted Interrupt**<br>0 = The data packet transmitted interrupt in Control Endpoint Disabled.<br>1 = The data packet transmitted interrupt in Control Endpoint Enabled. |
| [4] | **PINGIEN** | **Ping Token Interrupt**<br>0 = The ping token interrupt in Control Endpoint Disabled.<br>1 = The ping token interrupt Control Endpoint Enabled. |
| [3] | **INTKIEN** | **in Token Interrupt**<br>0 = The IN token interrupt in Control Endpoint Disabled.<br>1 = The IN token interrupt in Control Endpoint Enabled. |
| [2] | **OUTTKIEN** | **Out Token Interrupt**<br>0 = The OUT token interrupt in Control Endpoint Disabled.<br>1 = The OUT token interrupt in Control Endpoint Enabled. |
| [1] | **SETUPPKIEN** | **Setup Packet Interrupt**<br>0 = The SETUP packet interrupt in Control Endpoint Disabled.<br>1 = The SETUP packet interrupt in Control Endpoint Enabled. |
| [0] | **SETUPTKIEN** | **Setup Token Interrupt Enable Bit**<br>0 = The SETUP token interrupt in Control Endpoint Disabled.<br>1 = The SETUP token interrupt in Control Endpoint Enabled. |

### Control-Endpoint Interrupt Status (USBD_CEPINTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_CEPINTSTS** | USBD_BA+0x034 | R/W | Control-Endpoint Interrupt Status | 0x0000_1800 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | **BUFEMPTYIF** | **BUFFULLIF** | **STSDONEIF** | **ERRIF** | **STALLIF** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **NAKIF** | **RXPKIF** | **TXPKIF** | **PINGIF** | **INTKIF** | **OUTTKIF** | **SETUPPKIF** | **SETUPTKIF** |

| Bits | Description | |
|---|---|---|
| [31:13] | **Reserved** | Reserved. |
| [12] | **BUFEMPTYIF** | **Buffer Empty Interrupt**<br>0 = The control-endpoint buffer is not empty.<br>1 = The control-endpoint buffer is empty.<br>**Note:** Write 1 to clear this bit to 0. |
| [11] | **BUFFULLIF** | **Buffer Full Interrupt**<br>0 = The control-endpoint buffer is not full.<br>1 = The control-endpoint buffer is full.<br>**Note:** Write 1 to clear this bit to 0. |
| [10] | **STSDONEIF** | **Status Completion Interrupt**<br>0 = No USB transaction has completed successfully.<br>1 = The status stage of a USB transaction has completed successfully.<br>**Note:** Write 1 to clear this bit to 0. |
| [9] | **ERRIF** | **USB Error Interrupt**<br>0 = No error had occurred during the transaction.<br>1 = An error had occurred during the transaction.<br>**Note:** Write 1 to clear this bit to 0. |
| [8] | **STALLIF** | **STALL Sent Interrupt**<br>0 = No stall-token is sent in response to an IN/OUT token.<br>1 = A stall-token is sent in response to an IN/OUT token.<br>**Note:** Write 1 to clear this bit to 0. |

| | | |
|---|---|---|
| [7] | **NAKIF** | **NAK Sent Interrupt**<br>0 = No NAK-token is sent in response to an IN/OUT token.<br>1 = A NAK-token is sent in response to an IN/OUT token.<br>**Note:** Write 1 to clear this bit to 0. |
| [6] | **RXPKIF** | **Data Packet Received Interrupt**<br>0 = No data packet is successfully received from the host for an OUT-token and an ACK is sent to the host.<br>1 = A data packet is successfully received from the host for an OUT-token and an ACK is sent to the host.<br>**Note:** Write 1 to clear this bit to 0. |
| [5] | **TXPKIF** | **Data Packet Transmitted Interrupt**<br>0 = No data packet is successfully transmitted to the host in response to an IN-token and an ACK-token is received for the same.<br>1 = A data packet is successfully transmitted to the host in response to an IN-token and an ACK-token is received for the same.<br>**Note:** Write 1 to clear this bit to 0. |
| [4] | **PINGIF** | **Ping Token Interrupt**<br>0 = The control-endpoint does not receive a ping token from the host.<br>1 = The control-endpoint receives a ping token from the host.<br>**Note:** Write 1 to clear this bit to 0. |
| [3] | **INTKIF** | **in Token Interrupt**<br>0 = The control-endpoint does not receive an IN token from the host.<br>1 = The control-endpoint receives an IN token from the host.<br>Write 1 to clear this bit to 0. |
| [2] | **OUTTKIF** | **Out Token Interrupt**<br>0 = The control-endpoint does not receive an OUT token from the host.<br>1 = The control-endpoint receives an OUT token from the host.<br>**Note:** Write 1 to clear this bit to 0. |
| [1] | **SETUPPKIF** | **Setup Packet Interrupt**<br>This bit must be cleared (by writing 1) before the next setup packet can be received. If the bit is not cleared, then the successive setup packets will be overwritten in the setup packet buffer.<br>0 = No setup packet has been received from the host.<br>1 = A setup packet has been received from the host.<br>**Note:** Write 1 to clear this bit to 0. |
| [0] | **SETUPTKIF** | **Setup Token Interrupt**<br>0 = No setup token is received.<br>1 = A setup token is received. Writing 1 clears this status bit<br>**Note:** Write 1 to clear this bit to 0. |

### Control-Endpoint In-transfer Data Count (USBD_CEPTXCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_CEPTXCNT** | USBD_BA+0x038 | R/W | Control-Endpoint In-transfer Data Count | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **TXCNT** | **In-transfer Data Count**<br>There is no mode selection for the control endpoint (but it operates like manual mode).The local-CPU has to fill the control-endpoint buffer with the data to be sent for an in-token and to write the count of bytes in this register. When zero is written into this field, a zero length packet is sent to the host. When the count written in the register is more than the MPS, the data sent will be of only MPS. |

### Control-Endpoint Out-transfer Data Count (USBD_CEPRXCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_CEPRXCNT** | USBD_BA+0x03C | R | Control-Endpoint Out-transfer Data Count | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **RXCNT** | **Out-transfer Data Count**<br>The USB device controller maintains the count of the data received in case of an out transfer, during the control transfer. |

### Control- Endpoint Data Count (USBD_CEPDATCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_CEPDATCNT** | USBD_BA+0x040 | R | Control-Endpoint Data Count | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **DATCNT** | **Control-endpoint Data Count**<br>The USB device controller maintains the count of the data of control-endpoint. |

### Setup1 & Setup0 Bytes (USBD_SETUP1_0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_SETUP1_0** | USBD_BA+0x044 | R | Setup1 & Setup0 bytes | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **SETUP1** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SETUP0** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:8] | **SETUP1** | **Setup Byte 1[15:8]**<br>This register provides byte 1 of the last setup packet received. For a Standard Device Request, the following bRequest Code information is returned.<br>00000000 = Get Status.<br>00000001 = Clear Feature.<br>00000010 = Reserved.<br>00000011 = Set Feature.<br>00000100 = Reserved.<br>00000101 = Set Address.<br>00000110 = Get Descriptor.<br>00000111 = Set Descriptor.<br>00001000 = Get Configuration.<br>00001001 = Set Configuration.<br>00001010 = Get Interface.<br>00001011 = Set Interface.<br>00001100 = Synch Frame. |

| | | |
|---|---|---|
| [7:0] | **SETUP0** | **Setup Byte 0[7:0]**<br>This register provides byte 0 of the last setup packet received. For a Standard Device Request, the following bmRequestType information is returned.<br>Bit 7(Direction):<br>0 = Host to device.<br>1 = Device to host.<br>Bit 6-5 (Type):<br>00 = Standard.<br>01 = Class.<br>10 = Vendor.<br>11 = Reserved.<br>Bit 4-0 (Recipient)<br>00000 = Device.<br>00001 = nterface.<br>00010 = Endpoint.<br>00011 =:Other.<br>Others =:Reserved. |

### Setup3 & Setup2 Bytes (USBD_SETUP3_2)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_SETUP3_2** | USBD_BA+0x048 | R | Setup3 & Setup2 Bytes | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **SETUP3** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SETUP2** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:8] | **SETUP3** | **Setup Byte 3 [15:8]**<br>This register provides byte 3 of the last setup packet received. For a Standard Device Request, the most significant byte of the wValue field is returned. |
| [7:0] | **SETUP2** | **Setup Byte 2 [7:0]**<br>This register provides byte 2 of the last setup packet received. For a Standard Device Request, the least significant byte of the wValue field is returned. |

### Setup5 & Setup4 Bytes (USBD_SETUP5_4)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_SETUP5_4** | USBD_BA+0x04C | R | Setup5 & Setup4 Bytes | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **SETUP5** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SETUP4** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:8] | **SETUP5** | **Setup Byte 5[15:8]**<br>This register provides byte 5 of the last setup packet received. For a Standard Device Request, the most significant byte of the wIndex field is returned. |
| [7:0] | **SETUP4** | **Setup Byte 4[7:0]**<br>This register provides byte 4 of the last setup packet received. For a Standard Device Request, the least significant byte of the wIndex is returned. |

### Setup7 & Setup6 Bytes (USBD_SETUP7_6)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_SETUP7_6** | USBD_BA+0x050 | R | Setup7 & Setup6 Bytes | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **SETUP7** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **SETUP6** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:8] | **SETUP7** | **Setup Byte 7[15:8]**<br>This register provides byte 7 of the last setup packet received. For a Standard Device Request, the most significant byte of the wLength field is returned. |
| [7:0] | **SETUP6** | **Setup Byte 6[7:0]**<br>This register provides byte 6 of the last setup packet received. For a Standard Device Request, the least significant byte of the wLength field is returned. |

### Control Endpoint RAM Start Address Register (USBD_CEPBUFSTART)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_CEPBUFSTART** | USBD_BA+0x054 | R/W | Control Endpoint RAM Start Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:0] | **SADDR** | **Control-endpoint Start Address**<br>This is the start-address of the RAM space allocated for the control-endpoint. |

### Control Endpoint RAM End Address Register (USBD_CEPBUFEND)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_CEPBUFEND** | USBD_BA+0x058 | R/W | Control Endpoint RAM End Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:0] | **EADDR** | **Control-endpoint End Address**<br>This is the end-address of the RAM space allocated for the control-endpoint. |

### DMA Control Status Register (USBD_DMACTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_DMACTL** | USBD_BA+0x05C | R/W | DMA Control Status Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7] | **DMARST** | **Reset DMA State Machine**<br>0 = No reset the DMA state machine.<br>1 = Reset the DMA state machine. |
| [6] | **SGEN** | **Scatter Gather Function Enable Bit**<br>0 = Scatter gather function Disabled.<br>1 = Scatter gather function Enabled. |
| [5] | **DMAEN** | **DMA Enable Bit**<br>0 = DMA function Disabled.<br>1 = DMA function Enabled. |
| [4] | **DMARD** | **DMA Operation**<br>0 = The operation is a DMA write (read from USB buffer). DMA will check endpoint data available count (USBD_EPxDATCNT) according to EPNM setting before to perform DMA write operation.<br>1 = The operation is a DMA read (write to USB buffer). |
| [3:0] | **EPNUM** | **DMA Endpoint Address Bits**<br>Used to define the Endpoint Address |

### DMA Count Register (USBD_DMACNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_DMACNT** | USBD_BA+0x060 | R/W | DMA Count Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:20] | **Reserved** | Reserved. |
| [19:0] | **DMACNT** | **DMA Transfer Count**<br>The transfer count of the DMA operation to be performed is written to this register. |

### Endpoint A~L Data Register (USBD_EPADAT~ USBD_EPLDAT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPADAT** | USBD_BA+0x064 | R/W | Endpoint A Data Register | 0x0000_0000 |
| **USBD_EPBDAT** | USBD_BA+0x08C | R/W | Endpoint B Data Register | 0x0000_0000 |
| **USBD_EPCDAT** | USBD_BA+0x0B4 | R/W | Endpoint C Data Register | 0x0000_0000 |
| **USBD_EPDDAT** | USBD_BA+0x0DC | R/W | Endpoint D Data Register | 0x0000_0000 |
| **USBD_EPEDAT** | USBD_BA+0x104 | R/W | Endpoint E Data Register | 0x0000_0000 |
| **USBD_EPFDAT** | USBD_BA+0x12C | R/W | Endpoint F Data Register | 0x0000_0000 |
| **USBD_EPGDAT** | USBD_BA+0x154 | R/W | Endpoint G Data Register | 0x0000_0000 |
| **USBD_EPHDAT** | USBD_BA+0x17C | R/W | Endpoint H Data Register | 0x0000_0000 |
| **USBD_EPIDAT** | USBD_BA+0x1A4 | R/W | Endpoint I Data Register | 0x0000_0000 |
| **USBD_EPJDAT** | USBD_BA+0x1CC | R/W | Endpoint J Data Register | 0x0000_0000 |
| **USBD_EPKDAT** | USBD_BA+0x1F4 | R/W | Endpoint K Data Register | 0x0000_0000 |
| **USBD_EPLDAT** | USBD_BA+0x21C | R/W | Endpoint L Data Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **EPDAT** | **Endpoint A~L Data Register**<br>Endpoint A~L data buffer for the buffer transaction (read or write).<br>Only word or byte access are supported. |

## Endpoint A~L Interrupt Status Register (USBD_EPAINTSTS~ USBD_EPLINTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPAINTSTS** | USBD_BA+0x068 | R/W | Endpoint A Interrupt Status Register | 0x0000_0003 |
| **USBD_EPBINTSTS** | USBD_BA+0x090 | R/W | Endpoint B Interrupt Status Register | 0x0000_0003 |
| **USBD_EPCINTSTS** | USBD_BA+0x0B8 | R/W | Endpoint C Interrupt Status Register | 0x0000_0003 |
| **USBD_EPDINTSTS** | USBD_BA+0x0E0 | R/W | Endpoint D Interrupt Status Register | 0x0000_0003 |
| **USBD_EPEINTSTS** | USBD_BA+0x108 | R/W | Endpoint E Interrupt Status Register | 0x0000_0003 |
| **USBD_EPFINTSTS** | USBD_BA+0x130 | R/W | Endpoint F Interrupt Status Register | 0x0000_0003 |
| **USBD_EPGINTSTS** | USBD_BA+0x158 | R/W | Endpoint G Interrupt Status Register | 0x0000_0003 |
| **USBD_EPHINTSTS** | USBD_BA+0x180 | R/W | Endpoint H Interrupt Status Register | 0x0000_0003 |
| **USBD_EPIINTSTS** | USBD_BA+0x1A8 | R/W | Endpoint I Interrupt Status Register | 0x0000_0003 |
| **USBD_EPJINTSTS** | USBD_BA+0x1D0 | R/W | Endpoint J Interrupt Status Register | 0x0000_0003 |
| **USBD_EPKINTSTS** | USBD_BA+0x1F8 | R/W | Endpoint K Interrupt Status Register | 0x0000_0003 |
| **USBD_EPLINTSTS** | USBD_BA+0x220 | R/W | Endpoint L Interrupt Status Register | 0x0000_0003 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| Reserved | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | | | SHORTRXIF | ERRIF | NYETIF | STALLIF | NAKIF |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| PINGIF | INTKIF | OUTTKIF | RXPKIF | TXPKIF | SHORTTXIF | BUFEMPTYIF | BUFFULLIF |

| Bits | Description | |
|---|---|---|
| [31:13] | **Reserved** | Reserved. |
| [12] | **SHORTRXIF** | **Bulk Out Short Packet Received**<br>0 = No bulk out short packet is received.<br>1 = Received bulk out short packet (including zero length packet).<br>Write 1 to clear this bit to 0. |
| [11] | **ERRIF** | **ERR Sent**<br>0 = No any error in the transaction.<br>1 = There occurs any error in the transaction.<br>Write 1 to clear this bit to 0. |

| | | |
|---|---|---|
| [10] | **NYETIF** | **NYET Sent**<br>0 = The space available in the RAM is sufficient to accommodate the next on coming data packet.<br>1 = The space available in the RAM is not sufficient to accommodate the next on coming data packet.<br>Write 1 to clear this bit to 0. |
| [9] | **STALLIF** | **USB STALL Sent**<br>0 = The last USB packet could be accepted or provided because the endpoint was stalled, and was acknowledged with a STALL.<br>1 = The last USB packet could not be accepted or provided because the endpoint was stalled, and was acknowledged with a STALL.<br>Write 1 to clear this bit to 0. |
| [8] | **NAKIF** | **USB NAK Sent**<br>0 = The last USB IN packet could be provided, and was acknowledged with an ACK.<br>1 = The last USB IN packet could not be provided, and was acknowledged with a NAK.<br>Write 1 to clear this bit to 0. |
| [7] | **PINGIF** | **PING Token Interrupt**<br>0 = A Data PING token has not been received from the host.<br>1 = A Data PING token has been received from the host.<br>Write 1 to clear this bit to 0. |
| [6] | **INTKIF** | **Data IN Token Interrupt**<br>0 = No Data IN token has been received from the host.<br>1 = A Data IN token has been received from the host.<br>Write 1 to clear this bit to 0. |
| [5] | **OUTTKIF** | **Data OUT Token Interrupt**<br>0 = A Data OUT token has not been received from the host.<br>1 = A Data OUT token has been received from the host. This bit also set by PING token (in high-speed only).<br>**Note:** Write 1 to clear this bit to 0. |
| [4] | **RXPKIF** | **Data Packet Received Interrupt**<br>0 = No data packet is received from the host by the endpoint.<br>1 = A data packet is received from the host by the endpoint.<br>**Note:** Write 1 to clear this bit to 0. |
| [3] | **TXPKIF** | **Data Packet Transmitted Interrupt**<br>0 = No data packet is transmitted from the endpoint to the host.<br>1 = A data packet is transmitted from the endpoint to the host.<br>**Note:** Write 1 to clear this bit to 0. |
| [2] | **SHORTTXIF** | **Short Packet Transferred Interrupt**<br>0 = The length of the last packet was not less than the Maximum Packet Size (EPMPS).<br>1 = The length of the last packet was less than the Maximum Packet Size (EPMPS).<br>**Note:** Write 1 to clear this bit to 0. |

|  |  |  |
|---|---|---|
| [1] | **BUFEMPTYIF** | **Buffer Empty**<br>For an IN endpoint, a buffer is available to the local side for writing up to FIFO full of bytes.<br>0 = The endpoint buffer is not empty.<br>1 = The endpoint buffer is empty.<br>For an OUT endpoint:<br>0 = The currently selected buffer does not have a count of 0.<br>1 = The currently selected buffer has a count of 0, or no buffer is available on the local side (nothing to read).<br>**Note:** This bit is read-only. |
| [0] | **BUFFULLIF** | **Buffer Full**<br>For an IN endpoint, the currently selected buffer is full, or no buffer is available to the local side for writing (no space to write). For an OUT endpoint, there is a buffer available on the local side, and there are FIFO full of bytes available to be read (entire packet is available for reading).<br>0 = The endpoint packet buffer is not full.<br>1 = The endpoint packet buffer is full.<br>**Note:** This bit is read-only. |

## Endpoint A~L Interrupt Enable Control Register (USBD_EPAINTEN~ USBD_EPLINTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPAINTEN** | USBD_BA+0x06C | R/W | Endpoint A Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPBINTEN** | USBD_BA+0x094 | R/W | Endpoint B Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPCINTEN** | USBD_BA+0x0BC | R/W | Endpoint C Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPDINTEN** | USBD_BA+0x0E4 | R/W | Endpoint D Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPEINTEN** | USBD_BA+0x10C | R/W | Endpoint E Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPFINTEN** | USBD_BA+0x134 | R/W | Endpoint F Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPGINTEN** | USBD_BA+0x15C | R/W | Endpoint G Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPHINTEN** | USBD_BA+0x184 | R/W | Endpoint H Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPIINTEN** | USBD_BA+0x1AC | R/W | Endpoint I Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPJINTEN** | USBD_BA+0x1D4 | R/W | Endpoint J Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPKINTEN** | USBD_BA+0x1FC | R/W | Endpoint K Interrupt Enable Register | 0x0000_0000 |
| **USBD_EPLINTEN** | USBD_BA+0x224 | R/W | Endpoint L Interrupt Enable Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| Reserved | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | | | SHORTRXIEN | ERRIEN | NYETIEN | STALLIEN | NAKIEN |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| PINGIEN | INTKIEN | OUTTKIEN | RXPKIEN | TXPKIEN | SHORTTXIEN | BUFEMPTYIEN | BUFFULLIEN |

| Bits | Description | |
|---|---|---|
| [31:13] | **Reserved** | Reserved. |
| [12] | **SHORTRXIEN** | **Bulk Out Short Packet Interrupt Enable Bit**<br>When set, this bit enables a local interrupt to be set whenever bulk out short packet occurs on the bus for this endpoint.<br>0 = Bulk out interrupt Disabled.<br>1 = Bulk out interrupt Enabled. |

| | | |
|---|---|---|
| [11] | **ERRIEN** | **ERR Interrupt Enable Bit**<br>When set, this bit enables a local interrupt to be set whenever ERR condition occurs on the bus for this endpoint.<br>0 = Error event interrupt Disabled.<br>1 = Error event interrupt Enabled. |
| [10] | **NYETIEN** | **NYET Interrupt Enable Bit**<br>When set, this bit enables a local interrupt to be set whenever NYET condition occurs on the bus for this endpoint.<br>0 = NYET condition interrupt Disabled.<br>1 = NYET condition interrupt Enabled. |
| [9] | **STALLIEN** | **USB STALL Sent Interrupt Enable Bit**<br>When set, this bit enables a local interrupt to be set when a stall token is sent to the host.<br>0 = STALL token interrupt Disabled.<br>1 = STALL token interrupt Enabled. |
| [8] | **NAKIEN** | **USB NAK Sent Interrupt Enable Bit**<br>When set, this bit enables a local interrupt to be set when a NAK token is sent to the host.<br>0 = NAK token interrupt Disabled.<br>1 = NAK token interrupt Enabled. |
| [7] | **PINGIEN** | **PING Token Interrupt Enable Bit**<br>When set, this bit enables a local interrupt to be set when a PING token has been received from the host.<br>0 = PING token interrupt Disabled.<br>1 = PING token interrupt Enabled. |
| [6] | **INTKIEN** | **Data IN Token Interrupt Enable Bit**<br>When set, this bit enables a local interrupt to be set when a Data IN token has been received from the host.<br>0 = Data IN token interrupt Disabled.<br>1 = Data IN token interrupt Enabled. |
| [5] | **OUTTKIEN** | **Data OUT Token Interrupt Enable Bit**<br>When set, this bit enables a local interrupt to be set when a Data OUT token has been received from the host.<br>0 = Data OUT token interrupt Disabled.<br>1 = Data OUT token interrupt Enabled. |
| [4] | **RXPKIEN** | **Data Packet Received Interrupt Enable Bit**<br>When set, this bit enables a local interrupt to be set when a data packet has been transmitted to the host.<br>0 = Data packet has been transmitted to the host interrupt Disabled.<br>1 = Data packet has been transmitted to the host interrupt Enabled. |
| [3] | **TXPKIEN** | **Data Packet Transmitted Interrupt Enable Bit**<br>When set, this bit enables a local interrupt to be set when a data packet has been received from the host.<br>0 = Data packet has been received from the host interrupt Disabled.<br>1 = Data packet has been received from the host interrupt Enabled. |

| | | |
|---|---|---|
| [2] | **SHORTTXIEN** | **Short Packet Transferred Interrupt Enable Bit**<br>When set, this bit enables a local interrupt to be set when a short data packet has been transferred to/from the host.<br>0 = Short data packet interrupt Disabled.<br>1 = Short data packet interrupt Enabled. |
| [1] | **BUFEMPTYIEN** | **Buffer Empty Interrupt**<br>When set, this bit enables a local interrupt to be set when a buffer empty condition is detected on the bus.<br>0 = Buffer empty interrupt Disabled.<br>1 = Buffer empty interrupt Enabled. |
| [0] | **BUFFULLIEN** | **Buffer Full Interrupt**<br>When set, this bit enables a local interrupt to be set when a buffer full condition is detected on the bus.<br>0 = Buffer full interrupt Disabled.<br>1 = Buffer full interrupt Enabled. |

## Endpoint A~L Data Available Count Register (USBD_EPADATCNT~ USBD_EPLDATCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPADATCNT** | USBD_BA+0x070 | R | Endpoint A Data Available Count Register | 0x0000_0000 |
| **USBD_EPBDATCNT** | USBD_BA+0x098 | R | Endpoint B Data Available Count Register | 0x0000_0000 |
| **USBD_EPCDATCNT** | USBD_BA+0x0C0 | R | Endpoint C Data Available Count Register | 0x0000_0000 |
| **USBD_EPDDATCNT** | USBD_BA+0x0E8 | R | Endpoint D Data Available Count Register | 0x0000_0000 |
| **USBD_EPEDATCNT** | USBD_BA+0x110 | R | Endpoint E Data Available Count Register | 0x0000_0000 |
| **USBD_EPFDATCNT** | USBD_BA+0x138 | R | Endpoint F Data Available Count Register | 0x0000_0000 |
| **USBD_EPGDATCNT** | USBD_BA+0x160 | R | Endpoint G Data Available Count Register | 0x0000_0000 |
| **USBD_EPHDATCNT** | USBD_BA+0x188 | R | Endpoint H Data Available Count Register | 0x0000_0000 |
| **USBD_EPIDATCNT** | USBD_BA+0x1B0 | R | Endpoint I Data Available Count Register | 0x0000_0000 |
| **USBD_EPJDATCNT** | USBD_BA+0x1D8 | R | Endpoint J Data Available Count Register | 0x0000_0000 |
| **USBD_EPKDATCNT** | USBD_BA+0x200 | R | Endpoint K Data Available Count Register | 0x0000_0000 |
| **USBD_EPLDATCNT** | USBD_BA+0x228 | R | Endpoint L Data Available Count Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30:16] | **DMALOOP** | **DMA Loop**<br>This register is the remaining DMA loop to complete. Each loop means 32-byte transfer. |
| [15:0] | **DATCNT** | **Data Count**<br>For an IN endpoint (EPDIR(USBD_EPxCFG[3] is high.), this register returns the number of valid bytes in the IN endpoint packet buffer.<br>For an OUT endpoint (EPDIR(USBD_EPxCFG[3] is low.), this register returns the number of received valid bytes in the Host OUT transfer. |

### Endpoint A~L Response Control Register (USBD_EPARSPCTL~ USBD_EPLRSPCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPARSPCTL** | USBD_BA+0x074 | R/W | Endpoint A Response Control Register | 0x0000_0000 |
| **USBD_EPBRSPCTL** | USBD_BA+0x09C | R/W | Endpoint B Response Control Register | 0x0000_0000 |
| **USBD_EPCRSPCTL** | USBD_BA+0x0C4 | R/W | Endpoint C Response Control Register | 0x0000_0000 |
| **USBD_EPDRSPCTL** | USBD_BA+0x0EC | R/W | Endpoint D Response Control Register | 0x0000_0000 |
| **USBD_EPERSPCTL** | USBD_BA+0x114 | R/W | Endpoint E Response Control Register | 0x0000_0000 |
| **USBD_EPFRSPCTL** | USBD_BA+0x13C | R/W | Endpoint F Response Control Register | 0x0000_0000 |
| **USBD_EPGRSPCTL** | USBD_BA+0x164 | R/W | Endpoint G Response Control Register | 0x0000_0000 |
| **USBD_EPHRSPCTL** | USBD_BA+0x18C | R/W | Endpoint H Response Control Register | 0x0000_0000 |
| **USBD_EPIRSPCTL** | USBD_BA+0x1B4 | R/W | Endpoint I Response Control Register | 0x0000_0000 |
| **USBD_EPJRSPCTL** | USBD_BA+0x1DC | R/W | Endpoint J Response Control Register | 0x0000_0000 |
| **USBD_EPKRSPCTL** | USBD_BA+0x204 | R/W | Endpoint K Response Control Register | 0x0000_0000 |
| **USBD_EPLRSPCTL** | USBD_BA+0x22C | R/W | Endpoint L Response Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **DISBUF** | **SHORTTXEN** | **ZEROLEN** | **HALT** | **TOGGLE** | **MODE** | | **FLUSH** |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7] | **DISBUF** | **Buffer Disable Bit**<br>This bit is used to receive unknown size OUT short packet. The received packet size is reference USBD_EPxDATCNT register.<br>0 = Buffer Not Disabled when Bulk-OUT short packet is received.<br>1 = Buffer Disabled when Bulk-OUT short packet is received. |

|  |  |  |
|---|---|---|
| [6] | **SHORTTXEN** | **Short Packet Transfer Enable Bit**<br>This bit is applicable only in case of Auto-Validate Method. This bit is set to validate any remaining data in the buffer which is not equal to the MPS of the endpoint, and happens to be the last transfer. This bit gets cleared once the data packet is sent.<br>0 = Not validate any remaining data in the buffer which is not equal to the MPS of the endpoint.<br>1 = Validate any remaining data in the buffer which is not equal to the MPS of the endpoint. |
| [5] | **ZEROLEN** | **Zero Length**<br>This bit is used to send a zero-length packet response to an IN-token. When this bit is set, a zero packet is sent to the host on reception of an IN-token. This bit gets cleared once the zero length data packet is sent.<br>0 = A zero packet is not sent to the host on reception of an IN-token.<br>1 = A zero packet is sent to the host on reception of an IN-token. |
| [4] | **HALT** | **Endpoint Halt**<br>This bit is used to send a STALL handshake as response to the token from the host. When an Endpoint Set Feature (ep_halt) is detected by the local CPU, it must write a ‘1’ to this bit.<br>0 = Do not send a STALL handshake as response to the token from the host.<br>1 = Send a STALL handshake as response to the token from the host. |
| [3] | **TOGGLE** | **Endpoint Toggle**<br>This bit is used to clear the endpoint data toggle bit. Reading this bit returns the current state of the endpoint data toggle bit.<br>The local CPU may use this bit to initialize the end-point’s toggle in case of reception of a Set Interface request or a Clear Feature (ep_halt) request from the host. Only when toggle bit is “1”, this bit can be written into the inversed write data bit[3].<br>0 = Do not clear the endpoint data toggle bit.<br>1 = Clear the endpoint data toggle bit. |
| [2:1] | **MODE** | **Mode Control**<br>The two bits determine the operation mode of the in-endpoint.<br>00 = Auto-Validate Mode.<br>01 = Manual-Validate Mode.<br>10 = Fly Mode.<br>11 = Reserved.<br>**Note:** These bits are not valid for an out-endpoint. The auto validate mode will be activated when the reserved mode is selected. |
| [0] | **FLUSH** | **Buffer Flush**<br>Writing 1 to this bit causes the packet buffer to be flushed and the corresponding EP_AVAIL register to be cleared. This bit is self-cleared. This bit should always be written after a configuration event.<br>0 = The packet buffer is not flushed.<br>1 = The packet buffer is flushed by user. |

### Endpoint A~L Maximum Packet Size Register (USBD_EPAMPS~ USBD_EPLMPS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPAMPS** | USBD_BA+0x078 | R/W | Endpoint A Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPBMPS** | USBD_BA+0x0A0 | R/W | Endpoint B Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPCMPS** | USBD_BA+0x0C8 | R/W | Endpoint C Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPDMPS** | USBD_BA+0x0F0 | R/W | Endpoint D Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPEMPS** | USBD_BA+0x118 | R/W | Endpoint E Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPFMPS** | USBD_BA+0x140 | R/W | Endpoint F Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPGMPS** | USBD_BA+0x168 | R/W | Endpoint G Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPHMPS** | USBD_BA+0x190 | R/W | Endpoint H Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPIMPS** | USBD_BA+0x1B8 | R/W | Endpoint I Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPJMPS** | USBD_BA+0x1E0 | R/W | Endpoint J Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPKMPS** | USBD_BA+0x208 | R/W | Endpoint K Maximum Packet Size Register | 0x0000_0000 |
| **USBD_EPLMPS** | USBD_BA+0x230 | R/W | Endpoint L Maximum Packet Size Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:11] | **Reserved** | Reserved. |
| [10:0] | **EPMPS** | **Endpoint Maximum Packet Size**<br>This field determines the Maximum Packet Size of the Endpoint. |

### Endpoint A~L Transfer Count Register (USBD_EPATXCNT~ USBD_EPLTXCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPATXCNT** | USBD_BA+0x07C | R/W | Endpoint A Transfer Count Register | 0x0000_0000 |
| **USBD_EPBTXCNT** | USBD_BA+0x0A4 | R/W | Endpoint B Transfer Count Register | 0x0000_0000 |
| **USBD_EPCTXCNT** | USBD_BA+0x0CC | R/W | Endpoint C Transfer Count Register | 0x0000_0000 |
| **USBD_EPDTXCNT** | USBD_BA+0x0F4 | R/W | Endpoint D Transfer Count Register | 0x0000_0000 |
| **USBD_EPETXCNT** | USBD_BA+0x11C | R/W | Endpoint E Transfer Count Register | 0x0000_0000 |
| **USBD_EPFTXCNT** | USBD_BA+0x144 | R/W | Endpoint F Transfer Count Register | 0x0000_0000 |
| **USBD_EPGTXCNT** | USBD_BA+0x16C | R/W | Endpoint G Transfer Count Register | 0x0000_0000 |
| **USBD_EPHTXCNT** | USBD_BA+0x194 | R/W | Endpoint H Transfer Count Register | 0x0000_0000 |
| **USBD_EPITXCNT** | USBD_BA+0x1BC | R/W | Endpoint I Transfer Count Register | 0x0000_0000 |
| **USBD_EPJTXCNT** | USBD_BA+0x1E4 | R/W | Endpoint J Transfer Count Register | 0x0000_0000 |
| **USBD_EPKTXCNT** | USBD_BA+0x20C | R/W | Endpoint K Transfer Count Register | 0x0000_0000 |
| **USBD_EPLTXCNT** | USBD_BA+0x234 | R/W | Endpoint L Transfer Count Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:11] | **Reserved** | Reserved. |
| [10:0] | **TXCNT** | **Endpoint Transfer Count**<br>For IN endpoints, this field determines the total number of bytes to be sent to the host in case of manual validation method.<br>For OUT endpoints, this field has no effect. |

## Endpoint A~L Configuration Register (USBD_EPACFG~ USBD_EPLCFG)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPACFG** | USBD_BA+0x080 | R/W | Endpoint A Configuration Register | 0x0000_0012 |
| **USBD_EPBCFG** | USBD_BA+0x0A8 | R/W | Endpoint B Configuration Register | 0x0000_0022 |
| **USBD_EPCCFG** | USBD_BA+0x0D0 | R/W | Endpoint C Configuration Register | 0x0000_0032 |
| **USBD_EPDCFG** | USBD_BA+0x0F8 | R/W | Endpoint D Configuration Register | 0x0000_0042 |
| **USBD_EPECFG** | USBD_BA+0x120 | R/W | Endpoint E Configuration Register | 0x0000_0052 |
| **USBD_EPFCFG** | USBD_BA+0x148 | R/W | Endpoint F Configuration Register | 0x0000_0062 |
| **USBD_EPGCFG** | USBD_BA+0x170 | R/W | Endpoint G Configuration Register | 0x0000_0072 |
| **USBD_EPHCFG** | USBD_BA+0x198 | R/W | Endpoint H Configuration Register | 0x0000_0082 |
| **USBD_EPICFG** | USBD_BA+0x1C0 | R/W | Endpoint I Configuration Register | 0x0000_0092 |
| **USBD_EPJCFG** | USBD_BA+0x1E8 | R/W | Endpoint J Configuration Register | 0x0000_00A2 |
| **USBD_EPKCFG** | USBD_BA+0x210 | R/W | Endpoint K Configuration Register | 0x0000_00B2 |
| **USBD_EPLCFG** | USBD_BA+0x238 | R/W | Endpoint L Configuration Register | 0x0000_00C2 |

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

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:4] | **EPNUM** | **Endpoint Number**<br>This field selects the number of the endpoint. Valid numbers 1 to 15.<br>Do not support two endpoints have same endpoint number. |
| [3] | **EPDIR** | **Endpoint Direction**<br>0 = out-endpoint (Host OUT to Device).<br>1 = in-endpoint (Host IN to Device).<br>A maximum of one OUT and IN endpoint is allowed for each endpoint number. |

| | | |
|---|---|---|
| [2:1] | **EPTYPE** | **Endpoint Type**<br>This field selects the type of this endpoint. Endpoint 0 is forced to a Control type.<br>00 = Reserved.<br>01 = Bulk.<br>10 = Interrupt.<br>11 = Isochronous. |
| [0] | **EPEN** | **Endpoint Valid**<br>When set, this bit enables this endpoint. This bit has no effect on Endpoint 0, which is always enabled.<br>0 = The endpoint Disabled.<br>1 = The endpoint Enabled. |

### Endpoint A~L RAM Start Address Register (USBD_EPABUFSTART~ USBD_EPLBUFSTART)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPABUFSTART** | USBD_BA+0x084 | R/W | Endpoint A RAM Start Address Register | 0x0000_0000 |
| **USBD_EPBBUFSTART** | USBD_BA+0x0AC | R/W | Endpoint B RAM Start Address Register | 0x0000_0000 |
| **USBD_EPCBUFSTART** | USBD_BA+0x0D4 | R/W | Endpoint C RAM Start Address Register | 0x0000_0000 |
| **USBD_EPDBUFSTART** | USBD_BA+0x0FC | R/W | Endpoint D RAM Start Address Register | 0x0000_0000 |
| **USBD_EPEBUFSTART** | USBD_BA+0x124 | R/W | Endpoint E RAM Start Address Register | 0x0000_0000 |
| **USBD_EPFBUFSTART** | USBD_BA+0x14C | R/W | Endpoint F RAM Start Address Register | 0x0000_0000 |
| **USBD_EPGBUFSTART** | USBD_BA+0x174 | R/W | Endpoint G RAM Start Address Register | 0x0000_0000 |
| **USBD_EPHBUFSTART** | USBD_BA+0x19C | R/W | Endpoint H RAM Start Address Register | 0x0000_0000 |
| **USBD_EPIBUFSTART** | USBD_BA+0x1C4 | R/W | Endpoint I RAM Start Address Register | 0x0000_0000 |
| **USBD_EPJBUFSTART** | USBD_BA+0x1EC | R/W | Endpoint J RAM Start Address Register | 0x0000_0000 |
| **USBD_EPKBUFSTART** | USBD_BA+0x214 | R/W | Endpoint K RAM Start Address Register | 0x0000_0000 |
| **USBD_EPLBUFSTART** | USBD_BA+0x23C | R/W | Endpoint L RAM Start Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:0] | **SADDR** | **Endpoint Start Address**<br>This is the start-address of the RAM space allocated for the endpoint A~L. |

## Endpoint A~L RAM End Address Register (USBD_EPABUFEND~ USBD_EPLBUFEND)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_EPABUFEND** | USBD_BA+0x088 | R/W | Endpoint A RAM End Address Register | 0x0000_0000 |
| **USBD_EPBBUFEND** | USBD_BA+0x0B0 | R/W | Endpoint B RAM End Address Register | 0x0000_0000 |
| **USBD_EPCBUFEND** | USBD_BA+0x0D8 | R/W | Endpoint C RAM End Address Register | 0x0000_0000 |
| **USBD_EPDBUFEND** | USBD_BA+0x100 | R/W | Endpoint D RAM End Address Register | 0x0000_0000 |
| **USBD_EPEBUFEND** | USBD_BA+0x128 | R/W | Endpoint E RAM End Address Register | 0x0000_0000 |
| **USBD_EPFBUFEND** | USBD_BA+0x150 | R/W | Endpoint F RAM End Address Register | 0x0000_0000 |
| **USBD_EPGBUFEND** | USBD_BA+0x178 | R/W | Endpoint G RAM End Address Register | 0x0000_0000 |
| **USBD_EPHBUFEND** | USBD_BA+0x1A0 | R/W | Endpoint H RAM End Address Register | 0x0000_0000 |
| **USBD_EPIBUFEND** | USBD_BA+0x1C8 | R/W | Endpoint I RAM End Address Register | 0x0000_0000 |
| **USBD_EPJBUFEND** | USBD_BA+0x1F0 | R/W | Endpoint J RAM End Address Register | 0x0000_0000 |
| **USBD_EPKBUFEND** | USBD_BA+0x218 | R/W | Endpoint K RAM End Address Register | 0x0000_0000 |
| **USBD_EPLBUFEND** | USBD_BA+0x240 | R/W | Endpoint L RAM End Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:0] | **EADDR** | **Endpoint End Address**<br>This is the end-address of the RAM space allocated for the endpoint A~L. |

## USB Head word0 (USBD_UVCHDAT0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_UVCHDAT0** | USBD_BA+0x248 | R/W | USB Header Word0 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **DAT** | The first head data(byte 0 was sent first) |

### USB Head word1 (USBD_UVCHDAT1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_UVCHDAT1** | USBD_BA+0x24C | R/W | USB Header Word1 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **DAT** | The second head data(byte 0 was sent first) |

### USB Head word2 (USBD_UVCHDAT2)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_UVCHDAT2** | USBD_BA+0x250 | R/W | USB Header Word2 | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **DAT** | The third head data(byte 0 was sent first) |

### Endpoint A~L Header Count Register (USBD_UVCEPAHCNT~ USBD_UVCEPLCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_UVCEPAHCNT** | USBD_BA+0x254 | R/W | Endpoint A Header Count | 0x0000_0000 |
| **USBD_UVCEPBHCNT** | USBD_BA+0x258 | R/W | Endpoint B Header Count | 0x0000_0000 |
| **USBD_UVCEPCHCNT** | USBD_BA+0x25C | R/W | Endpoint C Header Count | 0x0000_0000 |
| **USBD_UVCEPDHCNT** | USBD_BA+0x260 | R/W | Endpoint D Header Count | 0x0000_0000 |
| **USBD_UVCEPEHCNT** | USBD_BA+0x264 | R/W | Endpoint E Header Count | 0x0000_0000 |
| **USBD_UVCEPFHCNT** | USBD_BA+0x268 | R/W | Endpoint F Header Count | 0x0000_0000 |
| **USBD_UVCEPGHCNT** | USBD_BA+0x26C | R/W | Endpoint G Header Count | 0x0000_0000 |
| **USBD_UVCEPHHCNT** | USBD_BA+0x270 | R/W | Endpoint H Header Count | 0x0000_0000 |
| **USBD_UVCEPIHCNT** | USBD_BA+0x274 | R/W | Endpoint I Header Count | 0x0000_0000 |
| **USBD_UVCEPJHCNT** | USBD_BA+0x278 | R/W | Endpoint J Header Count | 0x0000_0000 |
| **USBD_UVCEPKHCNT** | USBD_BA+0x27C | R/W | Endpoint K Header Count | 0x0000_0000 |
| **USBD_UVCEPLHCNT** | USBD_BA+0x280 | R/W | Endpoint L Header Count | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:4] | **Reserved** | Reserved. |
| [3:0] | **CNT** | **This is the header count for the endpoint A~L** The header count must be EVEN. |

### AHB Address Register (USBD_DMAADDR)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_DMAADDR** | USBD_BA+0x700 | R/W | AHB DMA Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **DMAADDR** | **DMAADDR**<br>The register specifies the address from which the DMA has to read / write. The address must WORD (32-bit) aligned. |

### USB PHY Control Register (USBD_PHYCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBD_PHYCTL** | USBD_BA+0x704 | R/W | USB PHY Control Register | 0x0000_0420 |

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

| Bits | Description | |
|---|---|---|
| [31] | **VBUSDET** | **VBUS Status**<br>0 = The VBUS is not detected yet.<br>1 = The VBUS is detected. |
| [30:25] | **Reserved** | Reserved. |
| [24] | **WKEN** | **Wake-up Enable Bit**<br>0 = The wake-up function Disabled.<br>1 = The wake-up function Enabled. |
| [23:10] | **Reserved** | Reserved. |
| [9] | **PHYEN** | **PHY Suspend Enable Bit**<br>0 = The USB PHY is suspend.<br>1 = The USB PHY is not suspend. |
| [8] | **DPPUEN** | **DP Pull-up**<br>0 = Pull-up resistor on D+ Disabled.<br>1 = Pull-up resistor on D+ Enabled. |
| [7:4] | **Reserved** | Reserved |
| [3] | **LOWPWREN** | **PHY Low Power Mode Enable Bit (Low Active)**<br>USB PHY HS low power mode configuration control bit.<br>When this bit is 0, USB PHY is in Low Power Mode and USB PHY TX driver is enable only when USB Device Controller (USBD) transmit the data out to USB bus. In this mode, HS signal quality is not guaranteed.<br>0 = USB PHY Low Power Mode Enabled.<br>1 = USB PHY Low Power Mode Disabled. |
| [2:0] | **Reserved** | Reserved. |

## 6.16 USB 1.1 Host Controller (USBH)

### 6.16.1 Overview

The NUC505 series is equipped with one USB 1.1 Host Controller (USBH) that supports Open Host Controller Interface (OpenHCI, OHCI) Specification and register-level description of a host controller to manage the devices and data transfer of Universal Serial Bus (USB).

The USBH supports an integrated Root Hub with a USB port, a DMA for real-time data transfer between system memory and USB bus, port power control and port overcurrent detection.

The USBH is responsible for detecting the connect and disconnect of USB devices, managing data transfer, collecting status and activity of USB bus, providing power control and detecting overcurrent of attached USB devices.

### 6.16.2 Features

- Supports Universal Serial Bus (USB) Specification Revision 1.1.
- Supports Open Host Controller Interface (OpenHCI) Specification Revision 1.0.
- Supports both full-speed (12Mbps) and low-speed (1.5Mbps) USB devices.
- Supports Control, Bulk, Interrupt and Isochronous transfers.
- Supports an integrated Root Hub.
- Supports one USB host port in LQFP48 or LQFP64 and two USB host ports in QFN88
- Supports port power control and port overcurrent detection.
- Supports DMA for real-time data transfer.

### 6.16.3 Block Diagram

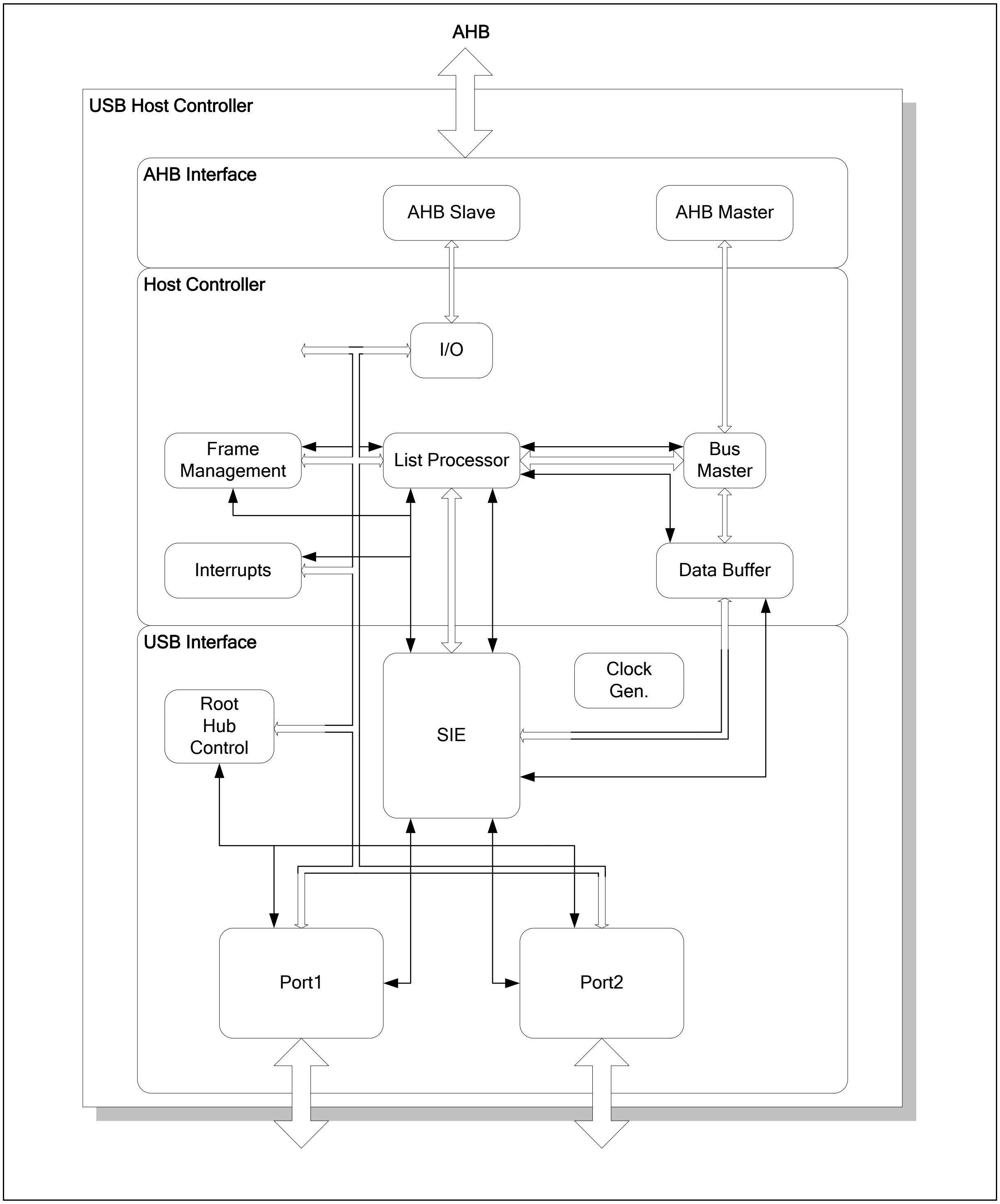


Figure 6.16-1 USB 1.1 Host Controller Block Diagram

### 6.16.4 Basic Configuration

The USBH clock source is derived from PLL. User has to set the PLL related configurations before USB host controller is enabled. Set the USBHCKEN (CLK_AHBCLK[4]) bit to enable USBH clock and 4-bit pre-scaler USBDIV (CLK_CLKDIV0[7:4]) to generate the proper USBH clock rate. The proper USBH clock rate is 48 MHz.

### 6.16.5 Functional Description

#### *6.16.5.1 AHB Interface*

The OpenHCI Host Controller is connected to the system by the AHB bus. The design requires both master and slave bus operations. As a master, the Host Controller is responsible for running cycles on the AHB bus to access EDs and TDs as well as transferring data between memory and the local data buffer. As a slave, the Host Controller monitors the cycles on the AHB bus and determines when to respond to these cycles. Configuration and non-real-time control access to the Host Controller operational registers are through the AHB bus slave interface.

#### *6.16.5.2 Host Controller*

The host controller includes 5 functional blocks, including List Processing, Frame Management, Interrupt Processing, Host Controller Bus Master and Data Buffer.

The List Processor manages the data structures from the Host Controller Driver and coordinates all activity within the Host Controller.

The Frame Management is responsible for managing the frame specific tasks required by the USB specification and the OpenHCI specification. These tasks are:

1) Management of the OpenHCI frame specific Operational Registers

2) Operation of the Largest Data Packet Counter.

3) Performing frame qualifications on USB Transaction requests to the SIE.

4) Generate SOF token requests to the SIE.

Interrupts are the communication method for HC-initiated communication with the Host Controller Driver. There are several events that may trigger an interrupt from the Host Controller. Each specific event sets a specific bit in the HcInterruptStatus register.

The Host Controller Bus Master is the central block in the data path. The Host Controller Bus Master coordinates all access to the AHB Interface. There are two sources of bus mastering within Host Controller: the List Processor and the Data Buffer Engine.

The Data Buffer serves as the data interface between the Bus Master and the SIE. It is a combination of a 64-byte latched based bi-directional asynchronous FIFO and a single Dword AHB Holding Register.

#### *6.16.5.3 USB Interface*

The USB interface includes the integrated Root Hub with two external ports, Port 1 and Port 2 as well as the Serial Interface Engine (SIE) and USB clock generator. The interface combines responsibility for executing bus transactions requested by the HC as well as the hub and port management specified by USB.

The SIE is responsible for managing all transactions to the USB. It controls the bus protocol, packet generation/extraction, data parallel-to-serial conversion, CRC coding, bit stuffing, and NRZI encoding. All transactions on the USB are requested from the List Processor and Frame Manager.

The Root Hub is a collection of ports that are individually controlled and a hub that maintains control/status over functions common to all ports.

### 6.16.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **USBH Base Address:**<br>**USBH_BA = 0x4000_b000** | | | | |
| **HCREVISION** | USBH_BA+0x000 | R | Host Controller Revision Register | 0x0000_0110 |
| **HCCONTROL** | USBH_BA+0x004 | R/W | Host Controller Control Register | 0x0000_0000 |
| **HCCOMMANDSTATUS** | USBH_BA+0x008 | R/W | Host Controller Command Status Register | 0x0000_0000 |
| **HCINTERRUPTSTATUS** | USBH_BA+0x00C | R/W | Host Controller Interrupt Status Register | 0x0000_0000 |
| **HCINTERRUPTENABLE** | USBH_BA+0x010 | R/W | Host Controller Interrupt Enable Control Register | 0x0000_0000 |
| **HCINTERRUPTDISABLE** | USBH_BA+0x014 | R/W | Host Controller Interrupt Disable Control Register | 0x0000_0000 |
| **HCHCCA** | USBH_BA+0x018 | R/W | Host Controller Communication Area Register | 0x0000_0000 |
| **HCPERIODCURRENTED** | USBH_BA+0x01C | R/W | Host Controller Period Current ED Register | 0x0000_0000 |
| **HCCONTROLHEADED** | USBH_BA+0x020 | R/W | Host Controller Control Head ED Register | 0x0000_0000 |
| **HCCONTROLCURRENTED** | USBH_BA+0x024 | R/W | Host Controller Control Current ED Register | 0x0000_0000 |
| **HCBULKHEADED** | USBH_BA+0x028 | R/W | Host Controller Bulk Head ED Register | 0x0000_0000 |
| **HCBULKCURRENTED** | USBH_BA+0x02C | R/W | Host Controller Bulk Current ED Register | 0x0000_0000 |
| **HCDONEHEAD** | USBH_BA+0x030 | R/W | Host Controller Done Head Register | 0x0000_0000 |
| **HCFMINTERVAL** | USBH_BA+0x034 | R/W | Host Controller Frame Interval Register | 0x0000_2EDF |
| **HCFMREMAINING** | USBH_BA+0x038 | R | Host Controller Frame Remaining Register | 0x0000_0000 |
| **HCFMNUMBER** | USBH_BA+0x03C | R | Host Controller Frame Number Register | 0x0000_0000 |
| **HCPERIODICSTART** | USBH_BA+0x040 | R/W | Host Controller Periodic Start Register | 0x0000_0000 |
| **HCLSTHRESHOLD** | USBH_BA+0x044 | R/W | Host Controller Low-speed Threshold Register | 0x0000_0628 |
| **HCRHDESCRIPTORA** | USBH_BA+0x048 | R/W | Host Controller Root Hub Descriptor A Register | 0x0000_0902 |
| **HCRHDESCRIPTORB** | USBH_BA+0x04C | R/W | Host Controller Root Hub Descriptor B Register | 0x0000_0000 |
| **HCRHSTATUS** | USBH_BA+0x050 | R/W | Host Controller Root Hub Status Register | 0x0000_0000 |
| **HCRHPORTSTATUS1** | USBH_BA+0x054 | R/W | Host Controller Root Hub Port Status [1] | 0x0000_0000 |
| **HCRHPORTSTATUS2** | USBH_BA+0x058 | R/W | Host Controller Root Hub Port Status [2] | 0x0000_0000 |
| **HCPHYCONTROL** | USBH_BA+0x200 | R/W | Host Controller PHY Control Register | 0x0000_0000 |
| **HCMISCCONTROL** | USBH_BA+0x204 | R/W | Host Controller Miscellaneous Control Register | 0x0000_0000 |

## 6.16.7 Register Description

### Host Controller Revision Register (HcRevision)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HCREVISION** | USBH_BA+0x000 | R | Host Controller Revision Register | 0x0000_0110 |

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

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **REV** | **Revision**<br>Indicates the Open HCI Specification revision number implemented by the Hardware. Host Controller supports 1.1 specification.<br>(X.Y = XYh). |

### Host Controller Control Register (HcControl)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HCCONTROL | USBH_BA+0x004 | R/W | Host Controller Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HCFS | | BLE | CLE | IE | PLE | CBSR | |

| Bits | Description | |
|---|---|---|
| [31:9] | Reserved | Reserved. |
| [8] | Reserved | Reserved. |
| [7:6] | HCFS | **Host Controller Functional State**<br>This field sets the Host Controller state. The Controller may force a state change from USBSUSPEND to USBRESUME after detecting resume signaling from a downstream port. States are:<br>00 = USBRESET.<br>01 = USBRESUME.<br>10 = USBOPERATIONAL.<br>11 = USBSUSPEND. |
| [5] | BLE | **Bulk List Enable Bit**<br>0 = Processing of the Bulk list after next SOF (Start-Of-Frame) Disabled.<br>1 = Processing of the Bulk list in the next frame Enabled. |
| [4] | CLE | **Control List Enable Bit**<br>0 = Processing of the Control list after next SOF (Start-Of-Frame) Disabled.<br>1 = Processing of the Control list in the next frame Enabled. |
| [3] | IE | **Isochronous Enable Bit**<br>Both ISOEn and PLE (HcControl[2]) high enables Host Controller to process the Isochronous list. Either ISOEn or PLE (HcControl[2]) is low disables Host Controller to process the Isochronous list.<br>0 = Processing of the Isochronous list after next SOF (Start-Of-Frame) Disabled.<br>1 = Processing of the Isochronous list in the next frame Enabled, if the PLE (HcControl[2]) is high, too. |

| | | |
|---|---|---|
| [2] | **PLE** | **Periodic List Enable Bit**<br>When set, this bit enables processing of the Periodic (interrupt and isochronous) list. The Host Controller checks this bit prior to attempting any periodic transfers in a frame.<br>0 = Processing of the Periodic (Interrupt and Isochronous) list after next SOF (Start-Of-Frame) Disabled.<br>1 = Processing of the Periodic (Interrupt and Isochronous) list in the next frame Enabled.<br>To enable the processing of the Isochronous list, user has to set both PLE and IE (HcControl[3]) high. |
| [1:0] | **CBSR** | **Control Bulk Service Ratio**<br>This specifies the service ratio between Control and Bulk EDs. Before processing any of the non-periodic lists, HC must compare the ratio specified with its internal count on how many nonempty Control EDs have been processed, in determining whether to continue serving another Control ED or switching to Bulk EDs. The internal count will be retained when crossing the frame boundary. In case of reset, HCD is responsible for restoring this<br>Value.<br>00 = Number of Control EDs over Bulk EDs served is 1:1.<br>01 = Number of Control EDs over Bulk EDs served is 2:1.<br>10 = Number of Control EDs over Bulk EDs served is 3:1.<br>11 = Number of Control EDs over Bulk EDs served is 4:1. |

### Host Controller Command Status Register (HcCommandStatus)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HCCOMMANDSTATUS | USBH_BA+0x008 | R/W | Host Controller Command Status Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:18] | **Reserved** | Reserved. |
| [17:16] | **SOC** | **Scheduling Overrun Count**<br>These bits are incremented on each scheduling overrun error. It is initialized to 00b and wraps around at 11b. This will be incremented when a scheduling overrun is detected even if SO (HcInterruptStatus[0]) has already been set. |
| [15:3] | **Reserved** | Reserved. |
| [2] | **BLF** | **Bulk List Filled**<br>Set high to indicate there is an active TD on the Bulk list. This bit may be set by either software or the Host Controller and cleared by the Host Controller each time it begins processing the head of the Bulk list.<br>0 = No active TD found or Host Controller begins to process the head of the Bulk list.<br>1 = An active TD added or found on the Bulk list. |
| [1] | **CLF** | **Control List Filled**<br>Set high to indicate there is an active TD on the Control List. It may be set by either software or the Host Controller and cleared by the Host Controller each time it begins processing the head of the Control List.<br>0 = No active TD found or Host Controller begins to process the head of the Control list.<br>1 = An active TD added or found on the Control list. |
| [0] | **HCR** | **Host Controller Reset**<br>This bit is set to initiate the software reset of Host Controller. This bit is cleared by the Host Controller, upon completed of the reset operation.<br>This bit, when set, didn’t reset the Root Hub and no subsequent reset signaling be asserted to its downstream ports.<br>0 = Host Controller is not in software reset state.<br>1 = Host Controller is in software reset state. |

## Host Controller Interrupt Status Register (HcInterruptStatus)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HCINTERRUPTSTATUS | USBH_BA+0x00C | R/W | Host Controller Interrupt Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| Reserved | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Reserved | RHSC | FNO | Reserved | RD | SF | WDH | SO |

| Bits | Description | |
|---|---|---|
| [31:7] | **Reserved** | Reserved. |
| [6] | **RHSC** | **Root Hub Status Change**<br>This bit is set when the content of HcRhStatus or the content of HcRhPortStatus1 register has changed.<br>0 = The content of HcRhStatus and the content of HcRhPortStatus1 register didn't change.<br>1 = The content of HcRhStatus or the content of HcRhPortStatus1 register has changed. |
| [5] | **FNO** | **Frame Number Overflow**<br>This bit is set when bit 15 of Frame Number changes from 1 to 0 or from 0 to 1.<br>0 = The bit 15 of Frame Number didn't change.<br>1 = The bit 15 of Frame Number changes from 1 to 0 or from 0 to 1. |
| [4] | **Reserved** | Reserved. |
| [3] | **RD** | **Resume Detected**<br>Set when Host Controller detects resume signaling on a downstream port.<br>0 = No resume signaling detected on a downstream port.<br>1 = Resume signaling detected on a downstream port. |
| [2] | **SF** | **Start of Frame**<br>Set when the Frame Management functional block signals a 'Start of Frame' event. Host Control generates a SOF token at the same time.<br>0 =.Not the start of a frame.<br>1 =.Indicate the start of a frame and Host Controller generates a SOF token. |

|  |  |  |
|---|---|---|
| [1] | **WDH** | **Write Back Done Head**<br>Set after the Host Controller has written HcDoneHead to HccaDoneHead. Further updates of the HccaDoneHead will not occur until this bit has been cleared.<br>0 =.Host Controller didn’t update HccaDoneHead.<br>1 =.Host Controller has written HcDoneHead to HccaDoneHead. |
| [0] | **SO** | **Scheduling Overrun**<br>Set when the List Processor determines a Schedule Overrun has occurred.<br>0 **=** Schedule Overrun didn’t occur.<br>1 = Schedule Overrun has occurred. |

### Host Controller Interrupt Enable Control Register (HcInterruptEnable)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HCINTERRUPTENABLE** | USBH_BA+0x010 | R/W | Host Controller Interrupt Enable Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **MIE** | **Reserved** | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | **RHSC** | **FNO** | **Reserved** | **RD** | **SF** | **WDH** | **SO** |

| Bits | Description | |
|---|---|---|
| [31] | **MIE** | **Master Interrupt Enable Bit**<br>This bit is a global interrupt enable. A write of '1' allows interrupts to be enabled via the specific enable bits listed above.<br>Write Operation:<br>0 = No effect.<br>1 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Enabled if the corresponding bit in HcInterruptEnable is high.<br>Read Operation:<br>0 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Disabled even if the corresponding bit in HcInterruptEnable is high.<br>1 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Enabled if the corresponding bit in HcInterruptEnable is high. |
| [30:7] | **Reserved** | Reserved. |
| [6] | **RHSC** | **Root Hub Status Change Interrupt Enable Bit**<br>Write Operation:<br>0 = No effect.<br>1 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Enabled.<br>Read Operation:<br>0 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Disabled.<br>1 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Enabled. |

|  |  |  |
|---|---|---|
| [5] | **FNO** | **Frame Number Overflow Interrupt Enable Bit**<br>Write Operation:<br>0 = No effect.<br>1 = Interrupt generation due to FNO (HcInterruptStatus[5]) Enabled.<br>Read Operation:<br>0 = Interrupt generation due to FNO (HcInterruptStatus[5]) Disabled.<br>1 = Interrupt generation due to FNO (HcInterruptStatus[5]) Enabled. |
| [4] | **Reserved** | Reserved. |
| [3] | **RD** | **Resume Detected Interrupt Enable Bit**<br>Write Operation:<br>0 = No effect.<br>1 = Interrupt generation due to RD (HcInterruptStatus[3]) Enabled.<br>Read Operation:<br>0 = Interrupt generation due to RD (HcInterruptStatus[3]) Disabled.<br>1 = Interrupt generation due to RD (HcInterruptStatus[3]) Enabled. |
| [2] | **SF** | **Start of Frame Interrupt Enable Bit**<br>Write Operation:<br>0 = No effect.<br>1 = Interrupt generation due to SF (HcInterruptStatus[2]) Enabled.<br>Read Operation:<br>0 = Interrupt generation due to SF (HcInterruptStatus[2]) Disabled.<br>1 = Interrupt generation due to SF (HcInterruptStatus[2]) Enabled. |
| [1] | **WDH** | **Write Back Done Head Interrupt Enable Bit**<br>Write Operation:<br>0 = No effect.<br>1 = Interrupt generation due to WDH (HcInterruptStatus[1]) Enabled.<br>Read Operation:<br>0 = Interrupt generation due to WDH (HcInterruptStatus[1]) Disabled.<br>1 = Interrupt generation due to WDH (HcInterruptStatus[1]) Enabled. |
| [0] | **SO** | **Scheduling Overrun Interrupt Enable Bit**<br>Write Operation:<br>0 = No effect.<br>1 = Interrupt generation due to SO (HcInterruptStatus[0]) Enabled.<br>Read Operation:<br>0 = Interrupt generation due to SO (HcInterruptStatus[0]) Disabled.<br>1 = Interrupt generation due to SO (HcInterruptStatus[0]) Enabled. |

## Host Controller Interrupt Disable Control Register (HcInterruptDisable)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HCINTERRUPTDISABLE | USBH_BA+0x014 | R/W | Host Controller Interrupt Disable Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| MIE | Reserved | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | RHSC | FNO | Reserved | RD | SF | WDH | SO |

| Bits | Description | |
|---|---|---|
| [31] | MIE | **Master Interrupt Disable Bit**<br>Global interrupt disable. Writing '1' to disable all interrupts.<br>Write Operation:<br>0 = No effect.<br>1 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Disabled if the corresponding bit in HcInterruptEnable is high.<br>Read Operation:<br>0 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Disabled even if the corresponding bit in HcInterruptEnable is high.<br>1 = Interrupt generation due to RHSC (HcInterruptStatus[6]), FNO (HcInterruptStatus[5]), RD (HcInterruptStatus[3]), SF (HcInterruptStatus[2]), WDH (HcInterruptStatus[1]) or SO (HcInterruptStatus[0]) Enabled if the corresponding bit in HcInterruptEnable is high. |
| [30:7] | Reserved | Reserved. |
| [6] | RHSC | **Root Hub Status Change Disable Bit**<br>Write Operation:<br>0 = No effect.<br>1 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Disabled.<br>Read Operation:<br>0 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Disabled.<br>1 = Interrupt generation due to RHSC (HcInterruptStatus[6]) Enabled. |

| | | |
|---|---|---|
| [5] | **FNO** | **Frame Number Overflow Disable Bit**<br>Write Operation:<br>0 = No effect.<br>1 = Interrupt generation due to FNO (HcInterruptStatus[5]) Disabled.<br>Read Operation:<br>0 = Interrupt generation due to FNO (HcInterruptStatus[5]) Disabled.<br>1 = Interrupt generation due to FNO (HcInterruptStatus[5]) Enabled. |
| [4] | **Reserved** | Reserved. |
| [3] | **RD** | **Resume Detected Disable Bit**<br>Write Operation:<br>0 = No effect.<br>1 = Interrupt generation due to RD (HcInterruptStatus[3]) Disabled.<br>Read Operation:<br>0 = Interrupt generation due to RD (HcInterruptStatus[3]) Disabled.<br>1 = Interrupt generation due to RD (HcInterruptStatus[3]) Enabled. |
| [2] | **SF** | **Start of Frame Disable Bit**<br>Write Operation:<br>0 = No effect.<br>1 = Interrupt generation due to SF (HcInterruptStatus[2]) Disabled.<br>Read Operation:<br>0 = Interrupt generation due to SF (HcInterruptStatus[2]) Disabled.<br>1 = Interrupt generation due to SF (HcInterruptStatus[2]) Enabled. |
| [1] | **WDH** | **Write Back Done Head Disable Bit**<br>Write Operation:<br>0 = No effect.<br>1 = Interrupt generation due to WDH (HcInterruptStatus[1]) Disabled.<br>Read Operation:<br>0 = Interrupt generation due to WDH (HcInterruptStatus[1]) Disabled.<br>1 = Interrupt generation due to WDH (HcInterruptStatus[1]) Enabled. |
| [0] | **SO** | **Scheduling Overrun Disable Bit**<br>Write Operation:<br>0 = No effect.<br>1 = Interrupt generation due to SO (HcInterruptStatus[0]) Disabled.<br>Read Operation:<br>0 = Interrupt generation due to SO (HcInterruptStatus[0]) Disabled.<br>1 = Interrupt generation due to SO (HcInterruptStatus[0]) Enabled. |

## Host Controller Communication Area Register (HcHCCA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HCHCCA** | USBH_BA+0x018 | R/W | Host Controller Communication Area Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:8] | **HCCA** | **Host Controller Communication Area**<br>Pointer to indicate base address of the Host Controller Communication Area (HCCA). |
| [7:0] | **Reserved** | Reserved. |

### Host Controller Period Current ED Register (HcPeriodCurrentED)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HCPERIODCURRENTED** | USBH_BA+0x01C | R/W | Host Controller Period Current ED Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:4] | **PCED** | **Periodic Current ED**<br>Pointer to indicate physical address of the current Isochronous or Interrupt Endpoint Descriptor. |
| [3:0] | **Reserved** | Reserved. |

### Host Controller Control Head ED Register (HcControlHeadED)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HCCONTROLHEADED** | USBH_BA+0x020 | R/W | Host Controller Control Head ED Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:4] | **CHED** | **Control Head ED**<br>Pointer to indicate physical address of the first Endpoint Descriptor of the Control list. |
| [3:0] | **Reserved** | Reserved. |

### Host Controller Control Current ED Register (HcControlCurrentED)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HCCONTROLCURRENTED** | USBH_BA+0x024 | R/W | Host Controller Control Current ED Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:4] | **CCED** | **Control Current Head ED**<br>Pointer to indicate the physical address of the current Endpoint Descriptor of the Control list. |
| [3:0] | **Reserved** | Reserved. |

### Host Controller Bulk Head ED Register (HcBulkHeadED)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HCBULKHEADED | USBH_BA+0x028 | R/W | Host Controller Bulk Head ED Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:4] | **BHED** | **Bulk Head ED**<br>Pointer to indicate the physical address of the first Endpoint Descriptor of the Bulk list. |
| [3:0] | **Reserved** | Reserved. |

### Host Controller Bulk Current Head ED Register (HcBulkCurrentED)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HCBULKCURRENTED** | USBH_BA+0x02C | R/W | Host Controller Bulk Current ED Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:4] | **BCED** | **Bulk Current Head ED**<br>Pointer to indicate the physical address of the current endpoint of the Bulk list. |
| [3:0] | **Reserved** | Reserved. |

### Host Controller Done Head Register (HcDoneHead)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HCDONEHEAD** | USBH_BA+0x030 | R/W | Host Controller Done Head Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:4] | **DH** | **Done Head**<br>Pointer to indicate the physical address of the last completed Transfer Descriptor that was added to the Done queue. |
| [3:0] | **Reserved** | Reserved. |

### Host Controller Frame Interval Register (HcFmInterval)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HCFMINTERVAL | USBH_BA+0x034 | R/W | Host Controller Frame Interval Register | 0x0000_2EDF |

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

| Bits | Description | |
|---|---|---|
| [31] | **FIT** | **Frame Interval Toggle**<br>This bit is toggled by Host Controller Driver when it loads a new value into FI (HcFmInterval[13:0]).<br>0 = Host Controller Driver didn't load new value into FI (HcFmInterval[13:0]).<br>1 = Host Controller Driver loads a new value into FI (HcFmInterval[13:0]). |
| [30:16] | **FSMPS** | **FS Largest Data Packet**<br>This field specifies a value that is loaded into the Largest Data Packet Counter at the beginning of each frame. |
| [15:14] | **Reserved** | Reserved. |
| [13:0] | **FI** | **Frame Interval**<br>This field specifies the length of a frame as (bit times - 1). For 12,000 bit times in a frame, a value of 11,999 is stored here. |

## Host Controller Frame Remaining Register (HcFmRemaining)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HCFMREMAINING** | USBH_BA+0x038 | R | Host Controller Frame Remaining Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31] | **FRT** | **Frame Remaining Toggle**<br>This bit is loaded from the FIT (HcFmInterval[31]) whenever FR (HcFmRemaining[13:0]) reaches 0. |
| [30:14] | **Reserved** | Reserved. |
| [13:0] | **FR** | **Frame Remaining**<br>When the Host Controller is in the USBOPERATIONAL state, this 14-bit field decrements each 12 MHz clock period. When the count reaches 0, (end of frame) the counter reloads with Frame Interval. In addition, the counter loads when the Host Controller transitions into USBOPERATIONAL. |

### Host Controller Frame Number Register (HcFmNumber)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HCFMNUMBER** | USBH_BA+0x03C | R | Host Controller Frame Number Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **Reserved** | Reserved. |
| [15:0] | **FN** | **Frame Number**<br>This 16-bit incrementing counter field is incremented coincident with the loading of FR (HcFmRemaining[13:0]). The count rolls over from ‘FFFFh’ to ‘0h.’ |

### Host Controller Periodic Start Register (HcPeriodicStart)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HCPERIODICSTART | USBH_BA+0x040 | R/W | Host Controller Periodic Start Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:14] | **Reserved** | Reserved. |
| [13:0] | **PS** | **Periodic Start**<br>This field contains a value used by the List Processor to determine where in a frame the Periodic List processing must begin. |

### Host Controller Low-speed Threshold Register (HcLSThreshold)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HCLSTHRESHOLD** | USBH_BA+0x044 | R/W | Host Controller Low-speed Threshold Register | 0x0000_0628 |

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

| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:0] | **LST** | **Low-speed Threshold**<br>This field contains a value which is compared to the FR (HcFmRemaining[13:0]) field prior to initiating a Low-speed transaction. The transaction is started only if FR (HcFmRemaining[13:0]) >= this field. The value is calculated by Host Controller Driver with the consideration of transmission and setup overhead. |

### Host Controller Root Hub Descriptor A Register (HcRhDescriptorA)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HCRHDESCRIPTORA | USBH_BA+0x048 | R/W | Host Controller Root Hub Descriptor A Register | 0x0000_0902 |

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

| Bits | Description | |
|---|---|---|
| [31:13] | **Reserved** | Reserved. |
| [12] | **NOCP** | **No Overcurrent Protection**<br>This bit describes how the overcurrent status for the Root Hub ports reported.<br>0 = Overcurrent status is reported.<br>1 = Overcurrent status is not reported. |
| [11] | **OCPM** | **Overcurrent Protection Mode**<br>This bit describes how the overcurrent status for the Root Hub ports reported. This bit is only valid when NOCP (HcRhDescriptorA[12]) is cleared.<br>0 = Global Overcurrent.<br>1 = Individual Overcurrent. |
| [10:9] | **Reserved** | Reserved. |
| [8] | **PSM** | **Power Switching Mode**<br>This bit is used to specify how the power switching of the Root Hub ports is controlled.<br>0 = Global Switching.<br>1 = Individual Switching. |
| [7:0] | **NDP** | **Number Downstream Ports**<br>Root Hub supports two downstream ports. It's 2 in this Root Hub. |

## Host Controller Root Hub Descriptor B Register (HcRhDescriptorB)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HCRHDESCRIPTORB | USBH_BA+0x04C | R/W | Host Controller Root Hub Descriptor B Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:16] | **PPCM** | **Port Power Control Mask**<br>Global power switching. This field is only valid if PowerSwitchingMode is set (individual port switching). When set, the port only responds to individual port power switching commands (Set/ClearPortPower). When cleared, the port only responds to global power switching commands (Set/ClearGlobalPower).<br>0 = Port power controlled by global power switching.<br>1 = Port power controlled by port power switching.<br>PPCM[15:3] and PPCM[0] are reserved. |
| [15:0] | **Reserved** | Reserved. |

### Host Controller Root Hub Status Register (HcRhStatus)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HCRHSTATUS** | USBH_BA+0x050 | R/W | Host Controller Root Hub Status Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **CRWE** | **Reserved** | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | **OCIC** | **LPSC** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **DRWE** | **Reserved** | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | | | | **OCI** | **LPS** |

| Bits | Description | |
|---|---|---|
| [31] | **CRWE** | **Clear Remote Wake-up Enable Bit**<br>This bit is used to clear DRWE (HcRhStatus[15]).<br>This bit is always read as zero.<br>Write Operation:<br>0 = No effect.<br>1 = Clear DRWE (HcRhStatus[15]). |
| [31:18] | **Reserved** | Reserved. |
| [17] | **OCIC** | **Overcurrent Indicator Change**<br>This bit is set by hardware when a change has occurred in OCI (HcRhStatus[1]).<br>Write 1 to clear this bit to zero.<br>0 = OCI (HcRhStatus[1]) is not changed.<br>1 = OCI (HcRhStatus[1]) is changed. |
| [16] | **LPSC** | **Set Global Power**<br>In global power mode (PSM (HcRhDescriptorA[8]) = 0), this bit is written to one to enable power to all ports.<br>This bit always read as zero.<br>Write Operation:<br>0 = No effect.<br>1 = Set global power. |

| | | |
|---|---|---|
| [15] | **DRWE** | **Device Remote Wake-up Enable Bit**<br>This bit controls if port's Connect Status Change as a remote wake-up event.<br>Write Operation:<br>0 = No effect.<br>1 = Connect status changed as a remote wake-up event Enabled.<br>Read Operation:<br>0 = Connect status changed as a remote wake-up event Disabled.<br>1 = Connect status changed as a remote wake-up event Enabled. |
| [14:2] | **Reserved** | Reserved. |
| [1] | **OCI** | **Overcurrent Indicator**<br>This bit reflects the state of the overcurrent status pin. This field is only valid if NOCP (HcRhDesA[12]) and OCPM (HcRhDesA[11]) are cleared.<br>0 = No overcurrent condition.<br>1 = Overcurrent condition. |
| [0] | **LPS** | **Clear Global Power**<br>In global power mode (PSM (HcRhDescriptorA[8]) = 0), this bit is written to one to clear all ports' power.<br>This bit always read as zero.<br>Write Operation:<br>0 = No effect.<br>1 = Clear global power. |

## Host Controller Root Hub Port Status (HcRhPrt [1: 2])

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HCRHPORTSTATUS1** | USBH_BA+0x054 | R/W | Host Controller Root Hub Port Status [1] | 0x0000_0000 |
| **HCRHPORTSTATUS2** | USBH_BA+0x058 | R/W | Host Controller Root Hub Port Status [2] | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| Reserved | | | PRSC | OCIC | PSSC | PESC | CSC |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | | | | | | LSDA | PPS |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| Reserved | | | PRS | POCI | PSS | PES | CCS |

| Bits | Description | |
|---|---|---|
| [31:21] | **Reserved** | Reserved. |
| [20] | **PRSC** | **Port Reset Status Change**<br>This bit indicates that the port reset signal has completed.<br>Write 1 to clear this bit to zero.<br>0 = Port reset is not completed.<br>1 = Port reset is completed. |
| [19] | **OCIC** | **Port Overcurrent Indicator Change**<br>This bit is set when POCI (HcRhPortStatus1[3]) changes.<br>Write 1 to clear this bit to zero.<br>0 = POCI (HcRhPortStatus1[3]) is not changed.<br>1 = POCI (HcRhPortStatus1[3]) is changed. |
| [18] | **PSSC** | **Port Suspend Status Change**<br>This bit indicates the completion of the selective resume sequence for the port.<br>Write 1 to clear this bit to zero.<br>0 = Port resume is not completed.<br>1 = Port resume is completed. |
| [17] | **PESC** | **Port Enable Status Change**<br>This bit indicates that the port has been disabled (PES (HcRhPortStatus1[1]) cleared) due to a hardware event.<br>Write 1 to clear this bit to zero.<br>0 = PES (HcRhPortStatus1[1]) is not changed.<br>1 = PES (HcRhPortStatus1[1]) is changed. |

| | | |
|---|---|---|
| [16] | **CSC** | **Connect Status Change**<br>This bit indicates connect or disconnect event has been detected (CCS (HcRhPortStatus1[0]) changed).<br>Write 1 to clear this bit to zero.<br>0 = No connect/disconnect event (CCS (HcRhPortStatus1[0]) didn't change).<br>1 = Hardware detection of connect/disconnect event (CCS (HcRhPortStatus1[0]) changed). |
| [15:10] | **Reserved** | Reserved. |
| [9] | **LSDA** | **Low Speed Device Attached (Read) or Clear Port Power (Write)**<br>This bit defines the speed (and bud idle) of the attached device. It is only valid when CCS (HcRhPortStatus1[0]) is set.<br>This bit is also used to clear port power.<br>Write Operation:<br>0 = No effect.<br>1 = Clear PPS (HcRhPortStatus1[8]).<br>Read Operation:<br>0 = Full Speed device.<br>1 = Low-speed device. |
| [8] | **PPS** | **Port Power Status**<br>This bit reflects the power state of the port regardless of the power switching mode.<br>Write Operation:<br>0 = No effect.<br>1 = Port Power Enabled.<br>Read Operation:<br>0 = Port power Diabled.<br>1 = Port power Enabled. |
| [7:5] | **Reserved** | Reserved. |
| [4] | **PRS** | **Port Reset Status**<br>This bit reflects the reset state of the port.<br>Write Operation:<br>0 = No effect.<br>1 = Set port reset.<br>Read Operation<br>0 = Port reset signal is not active.<br>1 = Port reset signal is active. |
| [3] | **POCI** | **Port Overcurrent Indicator (Read) or Clear Port Suspend (Write)**<br>This bit reflects the state of the overcurrent status pin dedicated to this port. This field is only valid if NOCP (HcRhDescriptorA[12]) is cleared and OCPM (HcRhDescriptorA[11]) is set.<br>This bit is also used to initiate the selective result sequence for the port.<br>Write Operation:<br>0 = No effect.<br>1 = Clear port suspend.<br>Read Operation:<br>0 = No overcurrent condition.<br>1 = Overcurrent condition. |

| | | |
|---|---|---|
| [2] | **PSS** | **Port Suspend Status**<br>This bit indicates the port is suspended<br>Write Operation:<br>0 = No effect.<br>1 = Set port suspend.<br>Read Operation:<br>0 = Port is not suspended.<br>1 = Port is selectively suspended. |
| [1] | **PES** | **Port Enable Status**<br>Write Operation:<br>0 = No effect.<br>1 = Set port Enabled.<br>Read Operation:<br>0 = Port Disabled.<br>1 = Port Enabled. |
| [0] | **CCS** | **Current Connect Status (Read) or Clear Port Enable Bit (Write)**<br>Write Operation:<br>0 = No effect.<br>1 = Clear port Enabled.<br>Read Operation:<br>0 = No device connected.<br>1 = Device connected. |

### Host Controller PHY Control Register (HcPhyControl)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| HCPHYCONTROL | USBH_BA+0x200 | R/W | Host Controller PHY Control Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:28] | **Reserved** | Reserved. |
| [27] | **STBYEN** | **USB Transceiver Standby Enable Bit**<br>This bit controls if USB transceiver could enter the standby mode to reduce power consumption.<br>0 = The USB transceiver would never enter the standby mode.<br>1 = The USB transceiver will enter Standby mode while port is in power off state (port power is inactive). |
| [26:0] | **Reserved** | Reserved. |

### Host Controller Miscellaneous Control Register (HcMiscControl)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **HCMISCCONTROL** | USBH_BA+0x204 | R/W | Host Controller Miscellaneous Control Register | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | **DPRT2** | **DPRT1** |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | | **SIEPD** |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **Reserved** | | | **PCAL** | **OCAL** | **Reserved** | **ABORT** | **DBR16** |

| Bits | Description | |
|---|---|---|
| [31:18] | **Reserved** | Reserved. |
| [17] | **DPRT2** | **Port 2 Disable Bit**<br>This bit controls if the connection between USB host controller and transceiver of port 2 is disabled. If the connection is disabled, the USB host controller will not recognize any event of USB bus.<br>Set this bit high, the transceiver of port 2 will also be forced into the standby mode no matter what USB host controller operation is.<br>0 = The connection between USB host controller and transceiver of port 2 is enabled.<br>1 = The connection between USB host controller and transceiver of port 2 is disabled and the transceiver of port 2 will also be forced into the standby mode. |
| [16] | **DPRT1** | **Port 1 Disable Bit**<br>This bit controls if the connection between USB host controller and transceiver of port 1 is disabled. If the connection is disabled, the USB host controller will not recognize any event of USB bus.<br>Set this bit high, the transceiver of port 1 will also be forced into the standby mode no matter what USB host controller operation is.<br>0 = The connection between USB host controller and transceiver of port 1 is enabled.<br>1 = The connection between USB host controller and transceiver of port 1 is disabled and the transceiver of port 1 will also be forced into the standby mode. |
| [15:9] | **Reserved** | Reserved. |
| [8] | **SIEPD** | **SIE Pipeline Disable Bit**<br>When set, waits for all USB bus activity to complete prior to returning completion status to the List Processor. This is a failsafe mechanism to avoid potential problems with the clk_dr transition between 1.5 MHz and 12 MHz. |
| [7:5] | **Reserved** | Reserved. |

| | | |
|---|---|---|
| [4] | **PCAL** | **Port Power Control Active Low**<br>This bit controls the polarity of port power control to external power IC.<br>0 = Port power control is high active.<br>1 = Port power control is low active. |
| [3] | **OCAL** | **Overcurrent Active Low**<br>This bit controls the polarity of overcurrent flag from external power IC.<br>0 = Overcurrent flag is high active.<br>1 = Overcurrent flag is low active. |
| [2] | **Reserved** | Reserved. |
| [1] | **ABORT** | **AHB Bus ERROR Response**<br>This bit indicates there is an ERROR response received in AHB bus.<br>0 = No ERROR response received.<br>1 = ERROR response received. |
| [0] | **DBR16** | **Data Buffer Region 16**<br>When set, the size of the data buffer region is 16 bytes. Otherwise, the size is 32 bytes. |

## 6.17 Secure-Digital Host Controller (SDHC)

### 6.17.1 Overview

The Secure-Digital Host Controller (SDH Controller) includes a DMAC (Direct Memory Access Controller) unit and a SD unit. The DMAC unit provides a DMA (Direct Memory Access) function for SD to exchange data between system memory and shared buffer (128 bytes), and the SD unit controls the interface of SD/SDHC/MMC. The SD HOST controller can support SD/SDHC/MMC with DMAC to provide a fast data transfer between system memory and cards.

### 6.17.2 Features

- Supports single DMA channel.
- Supports hardware Scatter-Gather function.
- Using single 128 Bytes shared buffer for data exchange between system memory and cards.
- Interface with DMAC for register read/write and data transfer.
- Supports SD/SDHC/MMC card.
- The frequency of HCLK should be higher than the frequency of peripheral clock.

### 6.17.3 Block Diagram and Card Pad Assignment

The block diagram and Card Pad Assignment of SDH Controller is shown as follows.

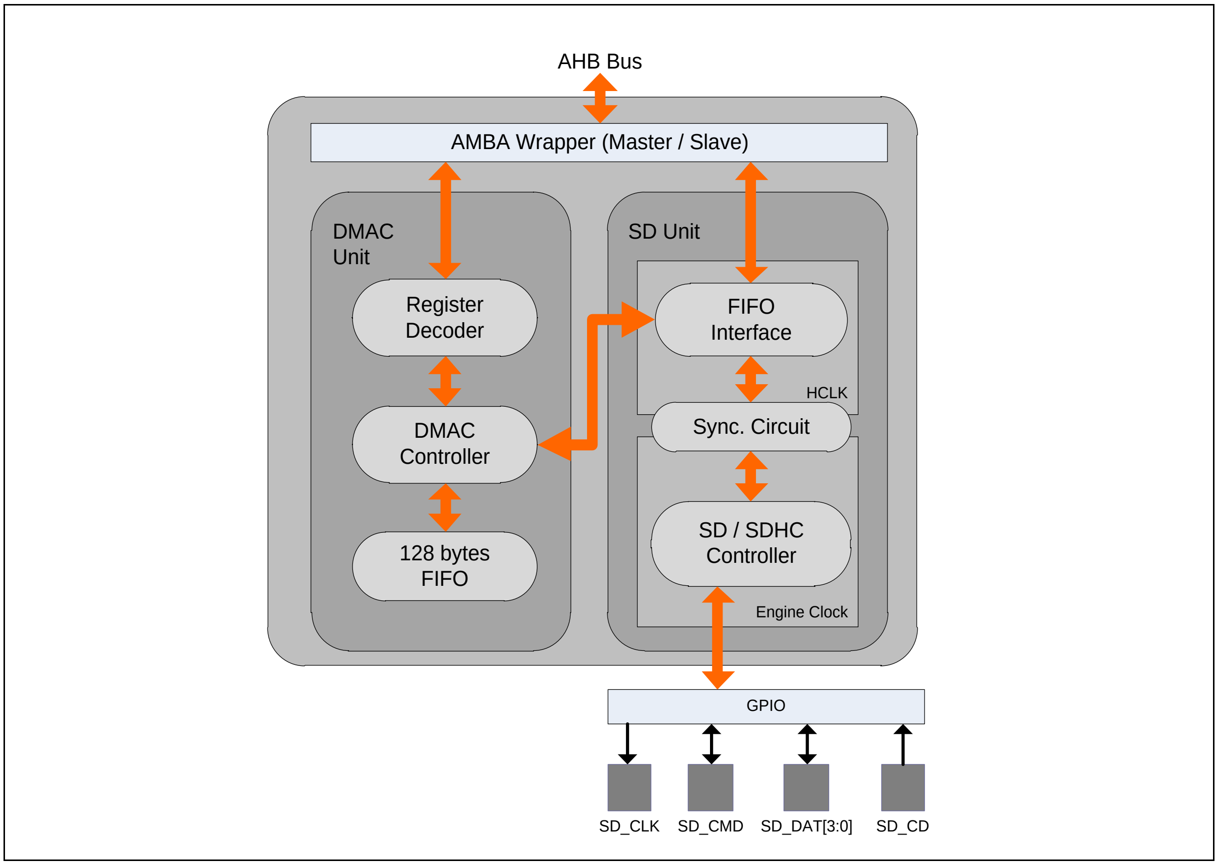


Figure 6.17-1 SDH Controller Block Diagram

| Name | Description |
|---|---|
| SD_DAT0 | SD Data (bit 0) |
| SD_DAT1 | SD Data (bit 1) |
| SD_DAT2 | SD Data (bit 2) |
| SD_DAT3 | SD Data (bit 3) |
| SD_CMD | SD Command / Response |
| SD_CLK | SD Clock pin |
| SD_CD | Card Detect (Source can be GPIO or DAT3 (in SDIER) |

Table 6.17-1 SD/SDHC/MMC Pad Assignment

### 6.17.4 SD HOST DMA Controller

The SD Host DMA Controller provides a DMA (Direct Memory Access) function to exchange data between system memory (e.g. SRAM) and shared buffer (128 bytes). Software just simply fills in the starting address and enables DMAC, then allowing DMAC to handle the data transfer automatically.

#### *6.17.4.1 Features*

- Supports single DMA channel and address in non-word boundary.
- Supports SD/SDHC/MMC cards in byte-access.
- Supports hardware Scatter-Gather function.
- One 128 bytes shared buffer is embedded.

#### *6.17.4.2 Block Diagram*

The block diagram of DMA Controller is shown below.

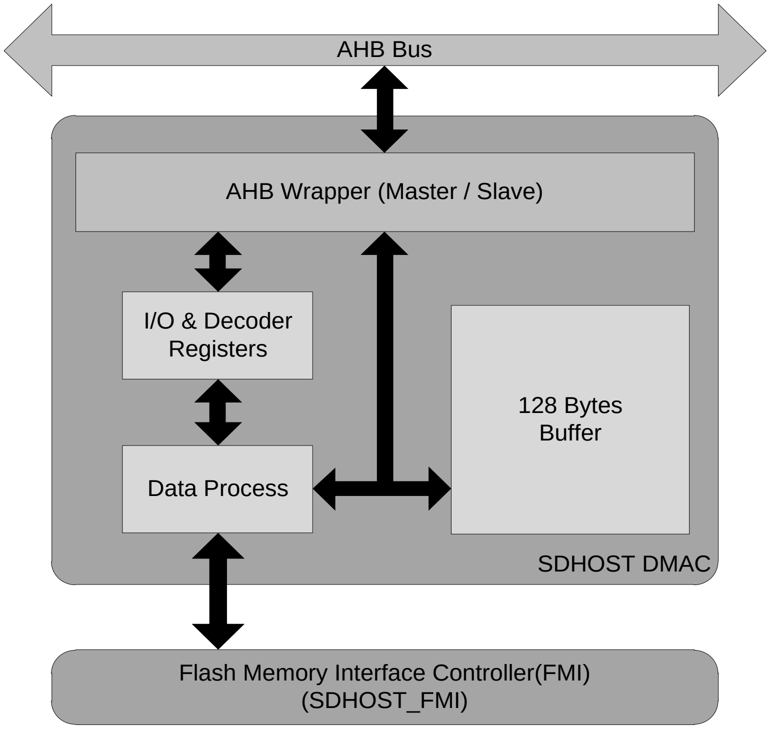

AHB Bus
AHB Wrapper (Master / Slave)
I/O & Decoder Registers
Data Process
128 Bytes Buffer
SDHOST DMAC
Flash Memory Interface Controller(FMI) (SDHOST_FMI)

Figure 6.17-2 DMA Controller Block Diagram

#### *6.17.4.3 Programming Flow*

Here is a simple example programming flow without DMA Scatter-Gather enabled.

1. Set DMAEN (SDH_DMACTL[0]) to enable DMAC.
2. Fill in the corresponding starting address in SDH_DMASA for DMAC.
3. Enable SDHOST_FMI to start DMA transfer.

4. Wait until SDHOST_FMI is finished.

Here is a simple example programming flow with DMA Scatter-Gather enabled.

1. Set DMAEN (SDH_DMACTL[0]) to enable DMAC and SGEN (SDH_DMACTL[3]) to enable Scatter-Gather function.
2. Fill in the corresponding starting address of Physical Address Descriptor (PAD) table in SDH_DMASA for DMAC.
3. When SDH_DMASA[0] is 1, the PAD will fetch in out of order; otherwise, it's fetched in order from PAD. The first time of writing bit-0 with 1 or 0 is not available for this function. The bits will be available in PAD table (Figure 6.17-3 Format of PAD TableFigure 6.17-3).
4. Enable SDHOST_FMI to start DMA transfer.
5. Wait until SDHOST_FMI is finished.

### 6.17.5 Secure-Digital SD/SDHC Controller

The Secure-Digital SD/SDHC/MMC is cooperated with DMAC to provide a fast data transfer between system memory and cards. There is a single 128 bytes buffer embedded in DMAC for temporary data storage. Since DMAC only has a single channel, this means only one interface can be active at one time.

#### *6.17.5.1 Features*

- Interface with DMAC for register read/write and data transfer
- Supports SD/SDHC/MMC card
- Supports SD/SDHC/MMC programmable timing cycle
- Using single 128 bytes shared buffer for data exchange between system memory and cards
- The frequency of HCLK should be higher than the frequency of engine clock

### *6.17.5.2 Functional Description*

The SD controller provides one SD port – Port0, which provides 1-bit/4-bit data bus mode for SD.

The SD controller uses an independent clock source named SDCLK as engine clock. SDCLK can be completely asynchronous with system clock HCLK, and software can change SD clock arbitrary. Note that HCLK should be faster than SDCLK.

The SD controller can generate all types of 48-bit command to SD card and retrieve all types of response from SD card. After response in, the content of response will be stored at SDH_RESP0 and SDH_RESP1. SD controller will calculate CRC-7 and check its correctness for response. If CRC-7 is incorrect, CRCIF (SDH_INTSTS[1]) will be set and CRC7 (SDH_INTSTS[2]) will be ‘0’. For the response R1b, after receiving response software should inform that the SD card will send a busy signal to data line DAT0; software should check this status with clock polling until it became high. For the response R3, CRC-7 is invalid; but the SD controller will still calculate CRC-7 and get an incorrect result, software should ignore this error and clear the CRCIF flag (SDH_INTSTS[1]).

This SD controller is composed of two state machines – command/response part and data part. For command/response part, the trigger bits are COEN, RIEN, R2EN, CLK74OEN and CLK8OEN in SDH_CTL. If software enables all of these bits, the execution priority will be CLK74OEN → COEN → RIEN /R2EN → CLK8OEN. Note that RIEN and R2EN cannot be triggered at the same time. For data part, there are DIEN and DOEN for selection. Software can only trigger one of them at one time. If DIEN is triggered, the SD controller will wait the start bit from data line DAT0 immediately, and then get the specified amount data from SD card. After data-in, the SD controller will check correctness of the CRC-16; if it is incorrect, the CRCIF (SDH_INTSTS[1]) will be set and CRC16 (SDH_INTSTS[3]) will be ‘0’. If DOEN is triggered, the SD controller will wait until response is finished, and then send the specified amount data to SD card. After data-out, the SD controller will get CRC status from SD card and check its correctness; it should be ‘010’, otherwise CRCIF (SDH_INTSTS[1]) will be set and CRCSTS (SDH_INTSTS[6:4]) will be the value it received.

If R2EN is triggered, the SD controller will receive response R2 (136 bits) from SD card, and CRC-7 and end bit will be dropped. The received data will be placed at DMAC’s buffer, starting from address offset 0x0.

The SD controller also provides multiple block transfer function (change BLKLEN to change the block length). Software can use this function to accelerate data transfer throughput. If CRC7, CRC16 or CRC status is incorrect, the SD controller will stop transfer and set the CRCIF (SDH_INTSTS[1]), and software should do engine reset when this situation occurred.

There is a hardware time-out mechanism for response in and data in inside SD engine. Software can specify a 24-bit time-out value at TOUT, and then the SD controller will determine when is time-out according to this value.

### 6.17.6 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH Base address:**<br>**SDH_BA = 0x4000_A000** | | | | |
| **SDH_FB_N**<br>**n=0,1,2...31** | SDH_BA+0x000+0x4 * n | R/W | SDH Receiving/Transmit Flash Buffer | 0x0000_0000 |
| **SDH_DMACTL** | SDH_BA+0x400 | R/W | DMAC Control and Status Register | 0x0000_0000 |
| **SDH_DMASA** | SDH_BA+0x408 | R/W | DMAC Transfer Starting Address Register | 0x0000_0000 |
| **SDH_DMABCNT** | SDH_BA+0x40C | R | DMAC Transfer Byte Count Register | 0x0000_0000 |
| **SDH_DMAINTEN** | SDH_BA+0x410 | R/W | DMAC Interrupt Enable Control Register | 0x0000_0001 |
| **SDH_DMAINTSTS** | SDH_BA+0x414 | R/W | DMAC Interrupt Status Register | 0x0000_0000 |
| **SDH_GCTL** | SDH_BA+0x800 | R/W | Global Control and Status Register | 0x0000_0000 |
| **SDH_GINTEN** | SDH_BA+0x804 | R/W | Global Interrupt Control Register | 0x0000_0001 |
| **SDH_GINTSTS** | SDH_BA+0x808 | R/W | Global Interrupt Status Register | 0x0000_0000 |
| **SDH_CTL** | SDH_BA+0x820 | R/W | SD Control and Status Register | 0x0101_0000 |
| **SDH_CMDARG** | SDH_BA+0x824 | R/W | SD Command Argument Register | 0x0000_0000 |
| **SDH_INTEN** | SDH_BA+0x828 | R/W | SD Interrupt Control Register | 0x0000_0A00 |
| **SDH_INTSTS** | SDH_BA+0x82C | R/W | SD Interrupt Status Register | 0x000X_008C |
| **SDH_RESP0** | SDH_BA+0x830 | R | SD Receiving Response Token Register 0 | 0x0000_0000 |
| **SDH_RESP1** | SDH_BA+0x834 | R | SD Receiving Response Token Register 1 | 0x0000_0000 |
| **SDH_BLEN** | SDH_BA+0x838 | R/W | SD Block Length Register | 0x0000_01FF |
| **SDH_TOUT** | SDH_BA+0x83C | R/W | SD Response/Data-in Time-out Register | 0x0000_0000 |

### 6.17.7 Register Description

#### *6.17.7.1 DMAC Register Description*

**SDH Receiving/Transmit Flash Buffer (SDH_FB_n)**

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_FB_N**<br>**n=0,1..31** | SDH_BA+0x000+0x4 * n | R/W | SDH Receiving/Transmit Flash Buffer | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **DATA** | **SDH Receiving/Transmit Flash Buffer**<br>This buffer is used to receive/transmit data. It can be accessed by CPU or DMAC. |

### DMAC Control and Status Register (SDH_DMACTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_DMACTL** | SDH_BA+0x400 | R/W | DMAC Control and Status Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:10] | **Reserved** | Reserved. |
| [9] | **DMABUSY** | **DMA Transfer in Progress**<br>This bit indicates that the DMA is transferred or not.<br>0 = DMA transfer is not in progress.<br>1 = DMA transfer is in progress. |
| [8:4] | **Reserved** | Reserved. |
| [3] | **SGEN** | **Scatter-gather Function Enable Control**<br>0 = Scatter-gather function Disabled (DMAC will treat the starting address in SDH_DMASA as starting pointer of a single block memory).<br>1 = Scatter-gather function Enabled (DMAC will treat the starting address in SDH_DMASA as a starting address of Physical Address Descriptor (PAD) table. The format of these PADs' will be described later). |
| [2] | **Reserved** | Reserved. |
| [1] | **DMARST** | **Software Engine Reset**<br>0 = No effect.<br>1 = Reset internal state machine and pointers. The contents of control register will not be cleared. This bit will auto be cleared after few clock cycles.<br>**Note:** Software resets DMA region. |
| [0] | **DMAEN** | **DMAC Engine Enable Control**<br>0 = DMAC Disabled.<br>1 = DMAC Enabled.<br>**Note1:** If this bit is cleared, DMAC will ignore all DMA request and force Bus Master into IDLE state.<br>**Note2:** If a target abort occurs, DMAEN will be cleared. |

### DMAC Transfer Starting Address Register (SDH_DMASA)

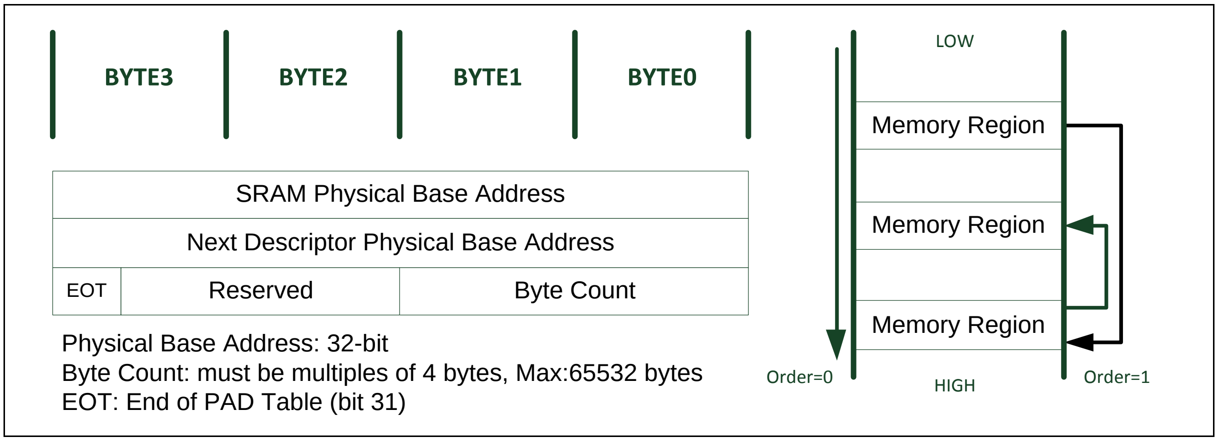
| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_DMASA** | SDH_BA+0x408 | R/W | DMAC Transfer Starting Address Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:1] | **DMASA** | **DMA Transfer Starting Address**<br>This field pads 0 as least significant bit indicates a 32-bit starting address of system memory (SRAM) for DMAC to retrieve or fill in data.<br>If DMAC is not in Normal mode, this field will be interpreted as a starting address of Physical Address Descriptor (PAD) table.<br>**Note:** The address can't be set smaller than 0x2000_0000. |
| [0] | **ORDER** | **Determined to the PAD Table Fetching in Order or Out of Order**<br>0 = PAD table is fetched in order.<br>1 = PAD table is fetched out of order.<br>**Note:** The bit0 is valid in scatter-gather mode when SGEN = 1. |

**Note:** The starting address of the SRAM must be word-aligned, for example, 0x0000_0000, 0x0000_0004…

The format of PAD table must like below. Note that the total byte count of all PADs must be equal to the byte count filled in DMAC engine. EOT should be set to 1 in the last descriptor.



| | | |
|---|---|---|
| SRAM Physical Base Address | | |
| Next Descriptor Physical Base Address | | |
| EOT | Reserved | Byte Count |

Physical Base Address: 32-bit
Byte Count: must be multiples of 4 bytes, Max:65532 bytes
EOT: End of PAD Table (bit 31)

Figure 6.17-3 Format of PAD Table

### DMAC Transfer Byte Count Register (SDH_DMABCNT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_DMABCNT** | SDH_BA+0x40C | R | DMAC Transfer Byte Count Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:26] | **Reserved** | Reserved. |
| [25:0] | **BCNT** | **DMA Transfer Byte Count (Read Only)**<br>This field indicates the remained byte count of DMAC transfer. The value of this field is valid only when DMAC is busy; otherwise, it is 0. |

### DMAC Interrupt Enable Control Register (SDH_DMAINTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_DMAINTEN** | SDH_BA+0x410 | R/W | DMAC Interrupt Enable Control Register | 0x0000_0001 |

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

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1] | **WEOTIEN** | **Wrong EOT Encountered Interrupt Enable Control**<br>0 = Interrupt generation Disabled when wrong EOT is encountered.<br>1 = Interrupt generation Enabled when wrong EOT is encountered. |
| [0] | **ABORTIEN** | **DMA Read/Write Target Abort Interrupt Enable Control**<br>0 = Target abort interrupt generation Disabled during DMA transfer.<br>1 = Target abort interrupt generation Enabled during DMA transfer. |

### DMAC Interrupt Status Register (SDH_DMAINTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_DMAINTSTS** | SDH_BA+0x414 | R/W | DMAC Interrupt Status Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1] | **WEOTIF** | **Wrong EOT Encountered Interrupt Flag**<br>When DMA Scatter-Gather function is enabled, and EOT of the descriptor is encountered before DMA transfer finished (that means the total sector count of all PAD is less than the sector count of DMAC), this bit will be set.<br>0 = No EOT encountered before DMA transfer finished.<br>1 = EOT encountered before DMA transfer finished.<br>**Note:** This bit is read only, but can be cleared by writing ‘1’ to it. |
| [0] | **ABORTIF** | **DMA Read/Write Target Abort Interrupt Flag**<br>0 = No bus ERROR response received.<br>1 = Bus ERROR response received.<br>**Note:** This bit is read only, but can be cleared by writing ‘1’ to it. |

**Note:** When DMAC’s bus master received ERROR response, it means that target abort happened. DMAC will stop transfer and respond this event to software, then go to IDLE state. When target abort occurred or WEOTIF is set, software must reset DMAC and peripheral, and then transfer those data again.

### 6.17.7.2 SD Register Description

#### Global Control and Status Register (SDH_GCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_GCTL** | SDH_BA+0x800 | R/W | Global Control and Status Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:2] | **Reserved** | Reserved. |
| [1] | **SDEN** | **Secure-digital Functionality Enable Control**<br>0 = SD functionality Disabled.<br>1 = SD functionality Enabled. |
| [0] | **GCTLRST** | **Software Engine Reset**<br>0 = No effect.<br>1 = Reset all SDH engines. The contents of control registers will not be cleared. This bit will auto cleared after few clock cycles. |

### Global Interrupt Control Register (SDH_GINTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_GINTEN** | SDH_BA+0x804 | R/W | Global Interrupt Control Register | 0x0000_0001 |

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

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **DTAIEN** | **DMAC READ/WRITE Target Abort Interrupt Enable Control**<br>0 = DMAC READ/WRITE target abort interrupt generation Disabled.<br>1 = DMAC READ/WRITE target abort interrupt generation Enabled. |

### Global Interrupt Status Register (SDH_GINTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_GINTSTS** | SDH_BA+0x808 | R/W | Global Interrupt Status Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:1] | **Reserved** | Reserved. |
| [0] | **DTAIF** | **DMAC READ/WRITE Target Abort Interrupt Flag (Read Only)**<br>This bit indicates DMAC received an ERROR response from internal AHB bus during DMA read/write operation. When Target Abort is occurred, please reset all engines.<br>0 = No bus ERROR response received.<br>1 = Bus ERROR response received.<br>**Note:** This bit is read only, but can be cleared by writing ‘1’ to it. |

No matter interrupt enable is turn on or not, the interrupt flag will be set when target condition is occurred.

### SD Control and Status Register (SDH_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDH_CTL | SDH_BA+0x820 | R/W | SD Control and Status Register | 0x0101_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | SDNWR | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| BLKCNT | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DBW | CTLRST | CMDCODE | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CLKKEEP0 | CLK8OEN | CLK74OEN | R2EN | DOEN | DIEN | RIEN | COEN |

| Bits | Description | |
|---|---|---|
| [31:28] | Reserved | Reserved. |
| [27:24] | SDNWR | **NWR Parameter for Block Write Operation**<br>This value indicates the NWR parameter for data block write operation in SD clock counts. The actual clock cycle will be SDNWR+1. |
| [23:16] | BLKCNT | **Block Counts to Be Transferred or Received**<br>This field contains the block counts for data-in and data-out transfer. For READ_MULTIPLE_BLOCK and WRITE_MULTIPLE_BLOCK command, software can use this function to accelerate data transfer and improve performance. Don't fill 0x0 to this field.<br>For READ_MULTIPLE_BLOCK and WRITE_MULTIPLE_BLOCK command, the actual total length is BLKCNT * (BLKLEN +1). |
| [15] | DBW | **SD Data Bus Width (for 1-bit / 4-bit Selection)**<br>0 = Data bus width is 1-bit.<br>1 = Data bus width is 4-bit. |
| [14] | CTLRST | **Software Engine Reset**<br>0 = No effect.<br>1 = Reset the internal state machine and counters. The contents of control register will not be cleared (but RIEN, DIEN, DOEN and R2EN will be cleared). This bit will be auto cleared after few clock cycles. |
| [13:8] | CMDCODE | **SD Command Code**<br>This register contains the SD command code (0x00 – 0x3F). |
| [7] | CLKKEEP0 | **SD Clock Enable Control for Port 0**<br>0 = SD host decides when to output clock and when to disable clock output automatically.<br>1 = SD clock always keeps free running. |

| | | |
|---|---|---|
| [6] | **CLK8OEN** | **Generating 8 Clock Cycles Output Enable Control**<br>0 = No effect. (Please use SDH_CTL[CTLRST] to clear this bit.)<br>1 = Enabled, and SD host will output 8 clock cycles.<br>**Note:** When operation is finished, this bit will be cleared automatically. Thus, don't write 0 to this bit (the controller will be abnormal). |
| [5] | **CLK74OEN** | **Generating 74 Clock Cycles Output Enable Control**<br>0 = No effect. (Please use SDH_CTL[CTLRST] to clear this bit.)<br>1 = Enabled, SD host will output 74 clock cycles to SD card.<br>**Note:** When operation is finished, this bit will be cleared automatically. Thus, don't write 0 to this bit (the controller will be abnormal). |
| [4] | **R2EN** | **Response R2 Input Enable Control**<br>0 = No effect. (Please use SDH_CTL[CTLRST] to clear this bit.)<br>1 = Enabled, SD host will wait to receive a response R2 from SD card and store the response data into DMAC's flash buffer (exclude CRC-7).<br>**Note:** When operation is finished, this bit will be cleared automatically. Thus, don't write 0 to this bit (the controller will be abnormal). |
| [3] | **DOEN** | **Data Output Enable Control**<br>0 = No effect. (Please use SDH_CTL[CTLRST] to clear this bit.)<br>1 = Enabled, SD host will transfer block data and the CRC-16 value to SD card.<br>**Note:** When operation is finished, this bit will be cleared automatically. Thus, don't write 0 to this bit (the controller will be abnormal). |
| [2] | **DIEN** | **Data Input Enable Control**<br>0 = No effect. (Please use SDH_CTL[CTLRST] to clear this bit.)<br>1 = Enabled, SD host will wait to receive block data and the CRC-16 value from SD card.<br>**Note:** When operation is finished, this bit will be cleared automatically. Thus, don't write 0 to this bit (the controller will be abnormal). |
| [1] | **RIEN** | **Response Input Enable Control**<br>0 = No effect. (Please use SDH_CTL[CTLRST] to clear this bit.)<br>1 = Enabled, SD host will wait to receive a response from SD card.<br>**Note:** When operation is finished, this bit will be cleared automatically. Thus, don't write 0 to this bit (the controller will be abnormal). |
| [0] | **COEN** | **Command Output Enable Control**<br>0 = No effect. (Please use SDH_CTL[CTLRST] to clear this bit.)<br>1 = Enabled, SD host will output a command to SD card.<br>**Note:** When operation is finished, this bit will be cleared automatically. Thus, don't write 0 to this bit (the controller will be abnormal). |

### SD Command Argument Register (SDH_CMDARG)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_CMDARG** | SDH_BA+0x824 | R/W | SD Command Argument Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:0] | **ARGUMENT** | **SD Command Argument**<br>This register contains a 32-bit value specifies the argument of SD command from host controller to SD card. Before triggering SDH_CTL [COEN], software should fill argument in this field. |

### SD Interrupt Control Register (SDH_INTEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| SDH_INTEN | SDH_BA+0x828 | R/W | SD Interrupt Control Register | 0x0000_0A00 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | CDSRC0 | Reserved | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| Reserved | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| Reserved | Reserved | DITOIEN | RTOIEN | Reserved | SDHIEN0 | Reserved | CDIEN0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Reserved | | | | | | CRCIEN | BLKDIEN |

| Bits | Description | |
|---|---|---|
| [31] | **Reserved** | Reserved. |
| [30] | **CDSRC0** | **SD0 Card Detect Source Selection**<br>0 = From SD0 card's DAT3 pin.<br>Host need clock to get data on pin DAT3. Please make sure SDH_CTL[CLKKEEP0] is 1 in order to generate free running clock for DAT3 pin.<br>1 = From GPIO pin. |
| [29:15] | **Reserved** | Reserved. |
| [14] | **Reserved** | Reserved. |
| [13] | **DITOIEN** | **Data Input Time-out Interrupt Enable Control**<br>Enable/Disable interrupt generation of SD controller when data input time-out. Time-out value is specified at TOUT.<br>0 = Data Input Time-out Interrupt Disabled.<br>1 = Data Input Time-out Interrupt Enabled. |
| [12] | **RTOIEN** | **Response Time-out Interrupt Enable Control**<br>Enable/Disable interrupt generation of SD controller when receiving response or R2 time-out. Time-out value is specified at TOUT.<br>0 = Response Time-out Interrupt Disabled.<br>1 = Response Time-out Interrupt Enabled. |
| [11] | **Reserved** | Reserved. |
| [10] | **SDHIEN0** | **SDH Interrupt Enable Control for Port 0**<br>Enable/Disable interrupt generation of SD host when SDH card 0 issues an interrupt via DAT [1] to host.<br>0 = SDH Port 0 Interrupt Disabled.<br>1 = SDH Port 0 Interrupt Enabled. |

| | | |
|---|---|---|
| [9] | **Reserved** | Reserved. |
| [8] | **CDIEN0** | **SD0 Card Detection Interrupt Enable Control**<br>Enable/Disable interrupt generation of SD controller when card 0 is inserted or removed.<br>0 = SD0 Card Detection Interrupt Disabled.<br>1 = SD0 Card Detection Interrupt Enabled. |
| [7:2] | **Reserved** | Reserved. |
| [1] | **CRCIEN** | **CRC7, CRC16 and CRC Status Error Interrupt Enable Control**<br>0 = SD host will not generate interrupt when CRC7, CRC16 and CRC status is error.<br>1 = SD host will generate interrupt when CRC7, CRC16 and CRC status is error. |
| [0] | **BLKDIEN** | **Block Transfer Done Interrupt Enable Control**<br>0 = SD host will not generate interrupt when data-in (out) transfer done.<br>1 = SD host will generate interrupt when data-in (out) transfer done. |

## SD Interrupt Status Register (SDH_INTSTS)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_INTSTS** | SDH_BA+0x82C | R/W | SD Interrupt Status Register | 0x000X_008C |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| Reserved | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| Reserved | | | | | DAT1STS | Reserved | CDSTS0 |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| Reserved | | DITOIF | RTOIF | Reserved | SDHIF0 | Reserved | CDIF0 |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| DAT0STS | CRCSTS | | | CRC16 | CRC7 | CRCIF | BLKDIF |

| Bits | Description | |
|---|---|---|
| [31:19] | **Reserved** | Reserved. |
| [18] | **DAT1STS** | **DAT1 Pin Status of SD Port (Read Only)**<br>This bit indicates the DAT1 pin status of SD port. |
| [17] | **Reserved** | Reserved. |
| [16] | **CDSTS0** | **Card Detect Status of SD0 (Read Only)**<br>This bit indicates the card detect pin status of SD0, and is used for card detection. When there is a card inserted in or removed from SD0, software should check this bit to confirm if there is really a card insertion or removal.<br>If CDSRC0 (SDH_INTEN[30]) = 0, to select DAT3 for card detection:.<br>0 = Card removed.<br>1 = Card inserted.<br>If CDSRC0 (SDH_INTEN[30]) = 1, to select GPIO for card detection:.<br>0 = Card inserted.<br>1 = Card removed. |
| [15:14] | **Reserved** | Reserved. |
| [13] | **DITOIF** | **Data Input Time-out Interrupt Flag (Read Only)**<br>This bit indicates that SD host counts to time-out value when receiving data (waiting start bit).<br>0 = No time-out.<br>1 = Data input time-out.<br>**Note:** This bit is read only, but can be cleared by writing ‘1’ to it. |

| | | |
|---|---|---|
| [12] | **RTOIF** | **Response Time-out Interrupt Flag (Read Only)**<br>This bit indicates that SD host counts to time-out value when receiving response or R2 (waiting start bit).<br>0 = No time-out.<br>1 = Response time-out.<br>**Note:** This bit is read only, but can be cleared by writing ‘1’ to it. |
| [11] | **Reserved** | Reserved. |
| [10] | **SDHIF0** | **SDH 0 Interrupt Flag (Read Only)**<br>This bit indicates that SDH card 0 issues an interrupt to host. This interrupt is designed to level sensitive. Before clearing it, turn off SDHIEN0 (SDH_INTEN[10]) first.<br>0 = No interrupt is issued by SDH card 0.<br>1 = an interrupt is issued by SDH card 0.<br>**Note:** This bit is read only, but can be cleared by writing ‘1’ to it. |
| [9] | **Reserved** | Reserved. |
| [8] | **CDIF0** | **SD0 Card Detection Interrupt Flag (Read Only)**<br>This bit indicates that SD card 0 is inserted or removed. Only when CDIEN0 (SDH_INTEN[8]) is set to 1, this bit is active.<br>0 = No card is inserted or removed.<br>1 = There is a card inserted in or removed from SD0.<br>**Note:** This bit is read only, but can be cleared by writing ‘1’ to it. |
| [7] | **DAT0STS** | **DAT0 Pin Status of Current Selected SD Port (Read Only)**<br>This bit is the DAT0 pin status of current selected SD port. |
| [6:4] | **CRCSTS** | **CRC Status Value of Data-out Transfer (Read Only)**<br>SD host will record CRC status of data-out transfer. Software could use this value to identify what type of error is during data-out transfer.<br>010 = Positive CRC status.<br>101 = Negative CRC status.<br>111 = SD card programming error occurs.<br>Others = Reserved. |
| [3] | **CRC16** | **CRC-16 Check Status of Data-in Transfer (Read Only)**<br>SD host will check CRC-16 correctness after data-in transfer.<br>0 = CRC-16 Transfer incorrectness.<br>1 = CRC-16 Transfer correctness. |
| [2] | **CRC7** | **CRC-7 Check Status (Read Only)**<br>SD host will check CRC-7 correctness during each response in. If that response does not contain CRC-7 information (e.g. R3), then software should turn off CRCIEN (SDH_INTEN[1]) and ignore this bit.<br>0 = CRC-7 Transfer incorrectness.<br>1 = CRC-7 Transfer correctness. |

| | | |
|---|---|---|
| [1] | **CRCIF** | **CRC7, CRC16 and CRC Status Error Interrupt Flag (Read Only)**<br>This bit indicates that SD host has occurred CRC error during response in, data-in or data-out (CRC status error) transfer. When CRC error occurred, software should reset SD engine. Some responses (e.g. R3) do not have CRC-7 information with it; the SD host will still calculate CRC-7, get CRC error and set this flag. In this condition, software should ignore CRC error and clear this bit manually.<br>0 = No CRC error is occurred.<br>1 = CRC error is occurred.<br>**Note:** This bit is read only, but can be cleared by writing ‘1’ to it. |
| [0] | **BLKDIF** | **Block Transfer Done Interrupt Flag (Read Only)**<br>This bit indicates that SD host has finished all data-in or data-out block transfer. If there is a CRC16 error or incorrect CRC status during multiple block data transfer, the transfer will be broken and this bit will also be set.<br>0 = Not finished yet.<br>1 = Done.<br>**Note:** This bit is read only, but can be cleared by writing ‘1’ to it. |

## SD Receiving Response Token Register 0 (SDH_RESP0)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_RESP0** | SDH_BA+0x830 | R | SD Receiving Response Token Register 0 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **RESPTK0** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **RESPTK0** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **RESPTK0** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **RESPTK0** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:0] | **RESPTK0** | **SD Receiving Response Token 0**<br>SD host controller will receive a response token for getting a reply from SD card when RIEN (SDH_CTL[1]) is set. This field contains response bit 47-16 of the response token. |

### SD Receiving Response Token Register 1 (SDH_RESP1)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_RESP1** | SDH_BA+0x834 | R | SD Receiving Response Token Register 1 | 0x0000_0000 |

| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
|---|---|---|---|---|---|---|---|
| **Reserved** | | | | | | | |
| **23** | **22** | **21** | **20** | **19** | **18** | **17** | **16** |
| **Reserved** | | | | | | | |
| **15** | **14** | **13** | **12** | **11** | **10** | **9** | **8** |
| **Reserved** | | | | | | | |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| **RESPTK1** | | | | | | | |

| Bits | Description | |
|---|---|---|
| [31:8] | **Reserved** | Reserved. |
| [7:0] | **RESPTK1** | **SD Receiving Response Token 1**<br>SD host controller will receive a response token for getting a reply from SD card when RIEN (SDH_CTL[1]) is set. This register contains the bit 15-8 of the response token. |

### SD Block Length Register (SDH_BLEN)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_BLEN** | SDH_BA+0x838 | R/W | SD Block Length Register | 0x0000_01FF |

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

| Bits | Description | |
|---|---|---|
| [31:11] | **Reserved** | Reserved. |
| [10:0] | **BLKLEN** | **SD BLOCK LENGTH in Byte Unit**<br>An 11-bit value specifies the SD transfer byte count of a block. The actual byte count is equal to BLKLEN +1.<br>**Note:** The default SD block length is 512 bytes. |

### SD Response/Data-in Time-out Register (SDH_TOUT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **SDH_TOUT** | SDH_BA+0x83C | R/W | SD Response/Data-in Time-out Register | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **Reserved** | Reserved. |
| [23:0] | **TOUT** | **SD Response/Data-in Time-out Value**<br>A 24-bit value specifies the time-out count of response and data input. SD host controller will wait the start bit of response or data-in until this value reached. The time period depends on SD engine clock frequency. Do not write a small number into this field, or you may never get response or data due to time-out.<br>**Note:** Filling 0x0 into this field will disable hardware time-out function. |

## 6.18 12-bit Analog-to-Digital Converter (ADC)

### 6.18.1 Overview

The NUC505 series contains one 12-bit successive approximation analog-to-digital converter (ADC) with 8 single-end external input channels (ADC_CH0, ADC_CH1, … ADC_CH7). The ADC_CH0 has an internal 10 kΩ resistor divider for battery detection. The ADC_CH2 also supports key pad comparator function. User can control the A/D conversion by setting the SWTRG (ADC_CTL[0]).

### 6.18.2 Features

- Analog input voltage range: 0~$AV_{DDADC}$.
- 12-bit resolution and 10-bit accuracy guaranteed.
- Up to 8 single-end analog input channels.
- ADC clock frequency up to 16 MHz.
- Up to 1 MSPS conversion rate when using in ADC_CH1 channel.
- Up to 200 kSPS conversion rate when using in ADC_CH2, …ADC_CH7 channels.
- Configurable ADC internal sampling time.
- Supports key pad comparator (ADC_CH2).
- Built-in 10 kΩ resistor divider for battery detection (ADC_CH0).

### 6.18.3 Block Diagram

Figure 6.18-1 is the whole ADC control block diagram. The ADC supports 8 single-end analog input channels. The channel selection is controlled by the CHSEL (ADC_CTL[18:16]).

The ADC_CH0 has an internal 10 kΩ resistor divider for supporting battery voltage detection. The switch of the resistor divider is still controlled by CHSEL (ADC_CTL[18:16]). If CHSEL is equal to 0x0, the switch is shorted. Oppositely, if CHSEL is not equal to 0x0, the switch will be opened.

The ADC_CH1 supports high speed conversion rate (1 MSPS). The calculation of the conversion rate is controlled by the ADC clock frequency.

Conversion rate = (ADC clock frequency) / 16

where ‘16’ means one A/D conversion needs 16 clock. ADC_CH2 ~ ADC_CH7 are up to 200 kSPS conversion rate. ADC_CH0 needs lengthy conversion time because of the 10 kΩ resistance. So, ADC_CH0 only can be up to 100 KSPS.

The ADC_CH2 also supports key pad application. The ADC has a key pad comparator block. When enabling key pad comparator block (setting PDKEY (ADC_CTL[5]) to 1), if ADC_CH2 is not equal to $AV_{DDADC}$, the flag KEYIF (ADC_INTCTL[1]) will be set. The KEYIF (ADC_INTCTL[1]) can be cleared by writing 1.

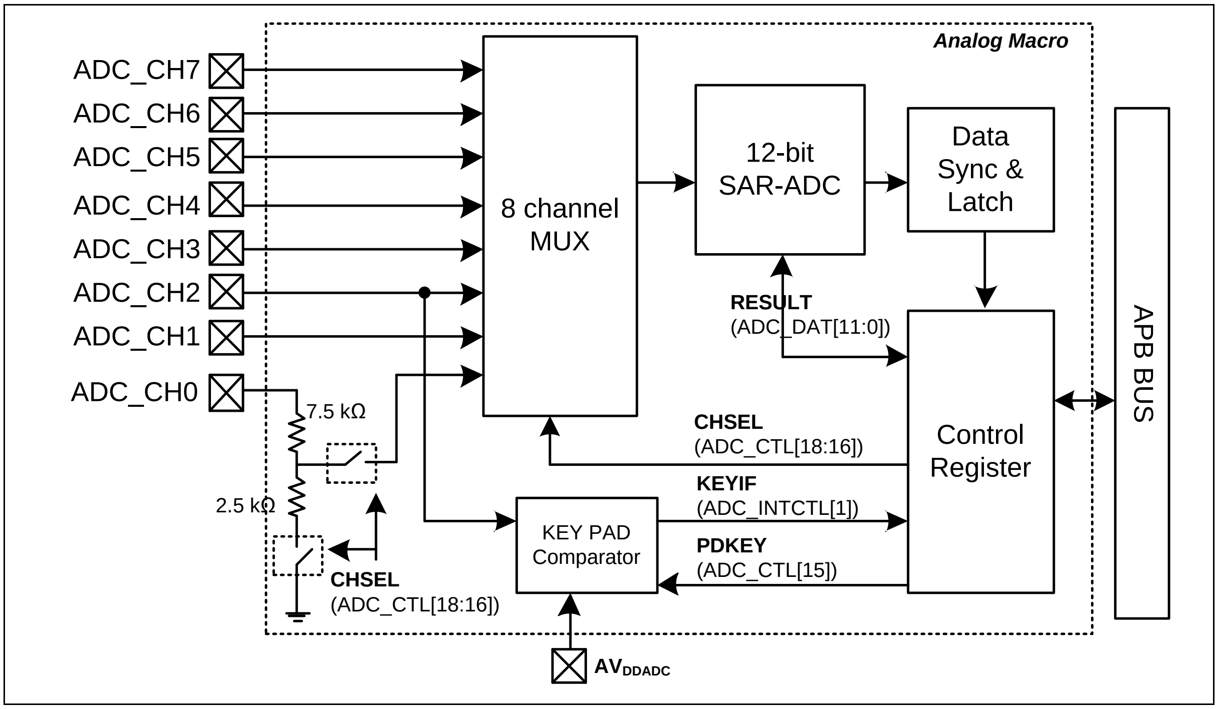


Figure 6.18-1 ADC Control Block Diagram

### 6.18.4 Operation Description

#### *6.18.4.1 Operation Timing Description*

During A/D operation at high ADC clock rate, the sampling time of analog input voltage may not be enough if the analog channel has heavy loading to cause fully charge time is longer. User can set extended sampling time by writing EXTSMPT (ADC_CTL[31:24]). The A/D extend sampling time is present between A/D controller to judge which channel to be converted and start A/D conversion. The range of extend sampling time is from 0 ~255 ADC clock. The extended sampling time is shown in Figure 6.18-2. From Figure 6.18-2, one A/D conversion needs 16 ADC clock.

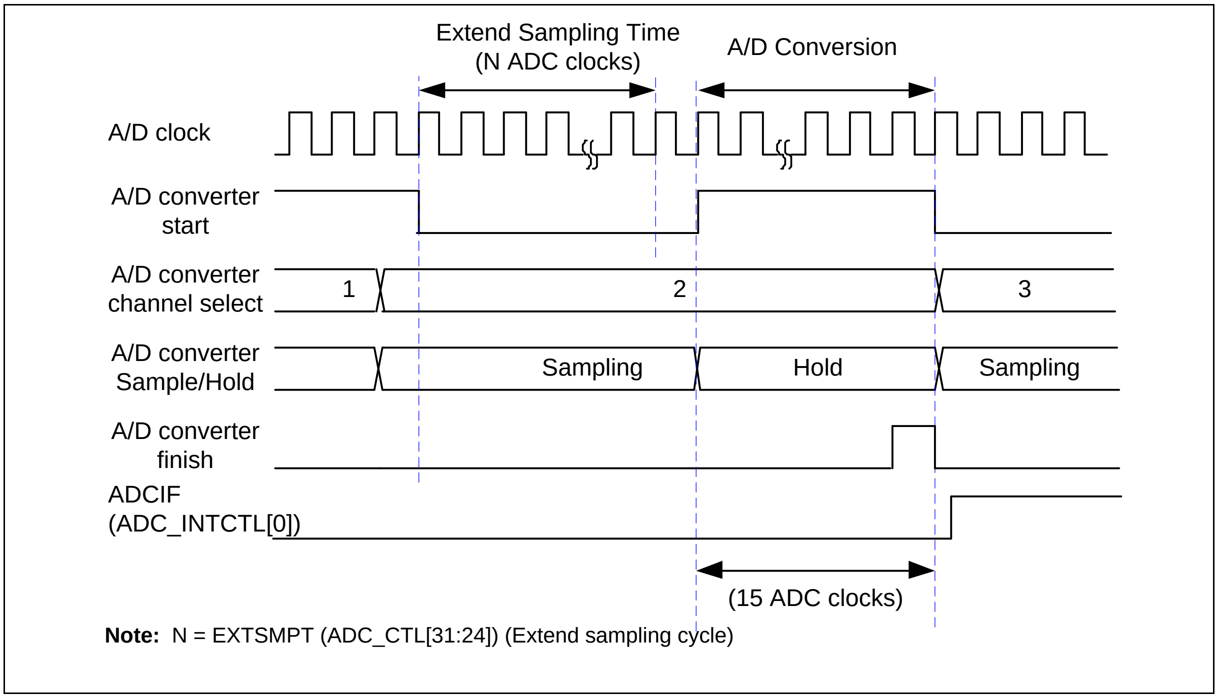


Figure 6.18-2 Conversion Timing Diagram

### *6.18.4.2 Key Pad Application*

Figure 6.18-3 shows an example of key pad application circuit. If any button has been connected, the voltage of ADC_CH2 will be lower than $AV_{DDADC}$. Therefore, a key pad comparator will compare the voltage of ADC_CH2 and $AV_{DDADC}$. Then, KEYIF (ADC_INTCTL[1]) will be set to 1.

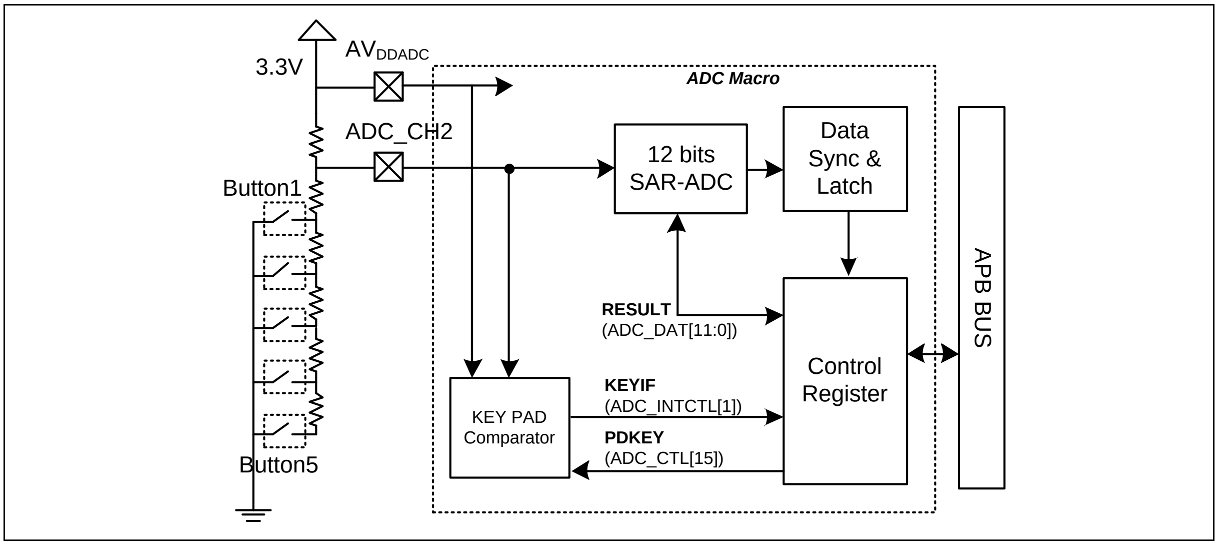


Figure 6.18-3 Key-Pad Application Block Diagram

### *6.18.4.3 Interrupt Sources*

Figure 6.18-4 shows the ADC interrupt sources. User can control KEYIEN (ADC_INTCTL[9]) and

ADCIEN (ADC_INTCTL[8]) to select the interrupt sources individually.

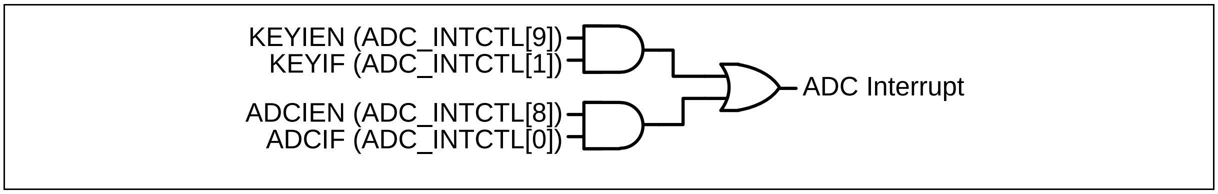


Figure 6.18-4 ADC Controller Interrupts

### 6.18.5 Register Map

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

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC Base Address:**<br>**ADC_BA = 0x400E_2000** | | | | |
| **ADC_CTL** | ADC_BA+0x00 | R/W | ADC Control Register | 0x0400_AE00 |
| **ADC_INTCTL** | ADC_BA+0x04 | R/W | ADC Interrupt State | 0x0000_0000 |
| **ADC_DAT** | ADC_BA+0x08 | R | ADC Data Register | 0x0000_0000 |

### 6.18.6 Register Description

#### ADC Control Register (ADC_CTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_CTL** | ADC_BA+0x00 | R/W | ADC Control Register | 0x0400_AE00 |

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

| Bits | Description | |
|---|---|---|
| [31:24] | **EXTSMPT** | **ADC Extend Sampling Time**<br>When A/D converting at high conversion rate, the sampling time of analog input voltage may not enough if the input channel loading is heavy, software can extend A/D sampling time after trigger source is coming to get enough sampling time.<br>**Note:** The unit is ADC clock. |
| [23:19] | **Reserved** | Reserved. |
| [18:16] | **CHSEL** | **Analog Input Selection Signals**<br>000 = ADC_CH0.<br>001 = ADC_CH1.<br>010 = ADC_CH2.<br>011 = ADC_CH3.<br>100 = ADC_CH4.<br>101 = ADC_CH5.<br>110 = ADC_CH6.<br>111 = ADC_CH7.<br>**Note1:** ADC_CH0 is used for battery voltage detection. It includes an inherent resistor divider and a switch.<br>**Note2:** User needs to pay attention to electric leakage with ADC_CH0 because the default CHSEL is selected to ADC_CH0, and ADC_CH0 has an internal resistor divider. If ADC_CH0 is connected to battery, there is a leakage path. Therefore, user can change CHSEL to other channel to cut off this path, when finishing battery detection. |
| [15] | **PDKEY** | **Power Down Keypad Detection**<br>0 = Power down keypad detection Disabled.<br>1 = Power down keypad detection Enabled. |

NUC505 SERIES TECHNICAL REFERENCE MANUAL

| | | |
|---|---|---|
| [14] | **Reserved** | Reserved. |
| [13] | **PD** | **Power Down ADC**<br>0 = ADC is in normal state.<br>1 = ADC is in power down state.<br>**Note1:** ADC power must be enabled before a trigger to get the data.<br>**Note2:** It needs 100ms to wait analog block stable when setting PD from 1 to 0. |
| [12:1] | **Reserved** | Reserved. |
| [0] | **SWTRG** | **A/D Conversion Start**<br>A trigger to start one A/D conversion process.<br>0 = A/D conversion enters idle state.<br>1 = Start conversion.<br>**Note:** This bit will be cleared to '0' automatically. |

### ADC Interrupt State (ADC_INTCTL)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_INTCTL** | ADC_BA+0x04 | R/W | ADC Interrupt State | 0x0000_0000 |

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

| Bits | Description | |
|---|---|---|
| [31:10] | **Reserved** | Reserved. |
| [9] | **KEYIEN** | **Keypad Interrupt Enable Control**<br>0 = Keypad down interrupt Disabled.<br>1 = Keypad down interrupt Enabled. |
| [8] | **ADCIEN** | **ADC Interrupt Enable Control**<br>0 = ADC conversion done interrupt Disabled.<br>1 = ADC conversion done interrupt Enabled. |
| [7:2] | **Reserved** | Reserved. |
| [1] | **KEYIF** | **Keypad Interrupt Flag**<br>In the process of checking keypad, the KEYIF shows the state. ADC_CH2 is used for keypad. When ADC_CH2 is not equal to $AV_{DDADC}$, the interrupt flag will be raised.<br>0 = Keypad is not pressing state.<br>1 = keypad is pressing state. |
| [0] | **ADCIF** | **ADC Conversion Done Interrupt Flag**<br>When finishing the sample process, the ADCIF bit will be set. If the ADCIEN is set, the interrupt will be transferred to NVIC.<br>0 = ADC conversion done flag is not set.<br>1 = ADC conversion done flag is set. |

### ADC Data Register (ADC_DAT)

| Register | Offset | R/W | Description | Reset Value |
|---|---|---|---|---|
| **ADC_DAT** | ADC_BA+0x08 | R | ADC Data Register | 0x0000_0000 |

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

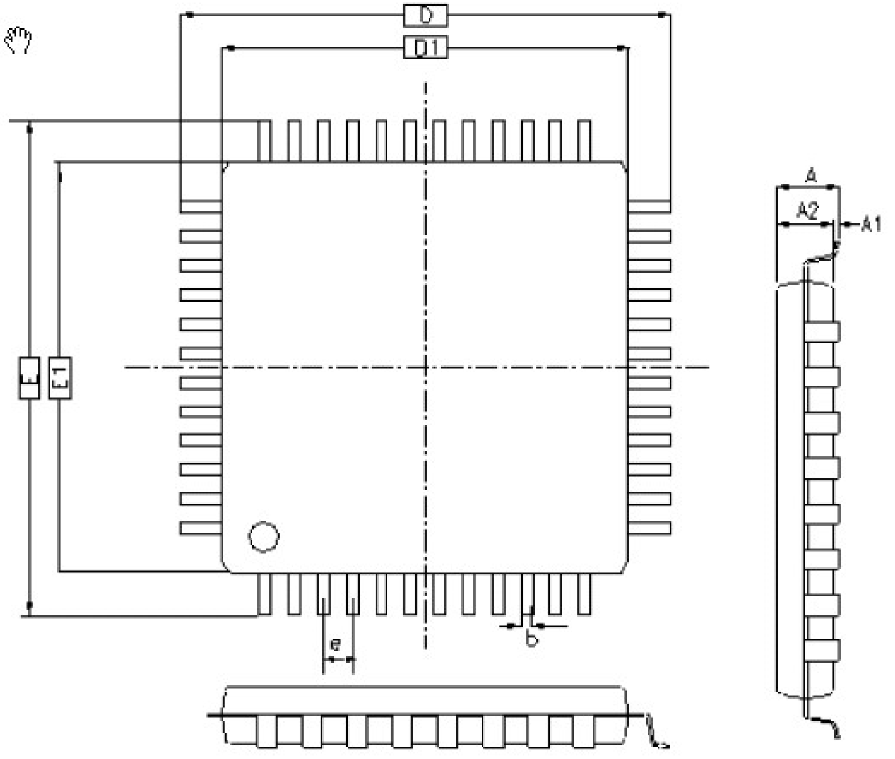
| Bits | Description | |
|---|---|---|
| [31:12] | **Reserved** | Reserved. |
| [11:0] | **RESULT** | **A/D Conversion Result**<br>This field contains conversion result of ADC. When A/D conversion done, 12-bit ADC conversion result with unsigned format will be filled in RESULT[11:0]. |

# 7 ELECTRICAL CHARACTERISTICS

For information on the NUC505 series electrical characteristics, please refer to NuMicro® NUC505 Series Datasheet.

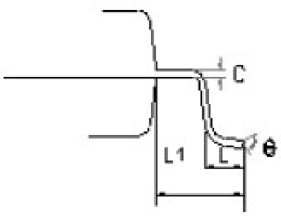
# 8 PACKAGE DIMENSIONS

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

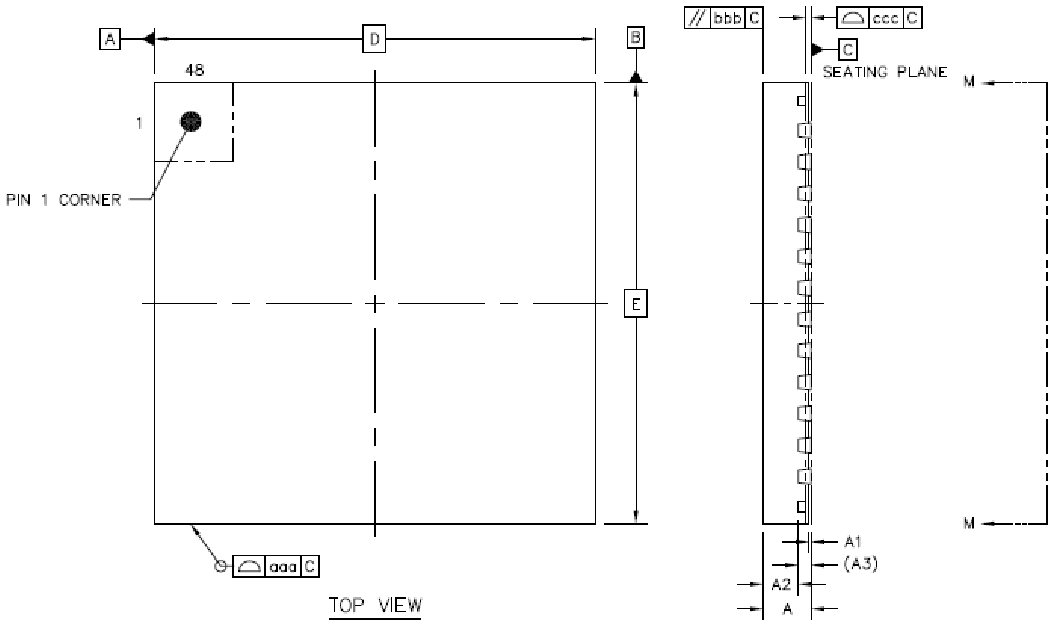


COTROL DIMENSIONS ARE IN MILLIMETERS

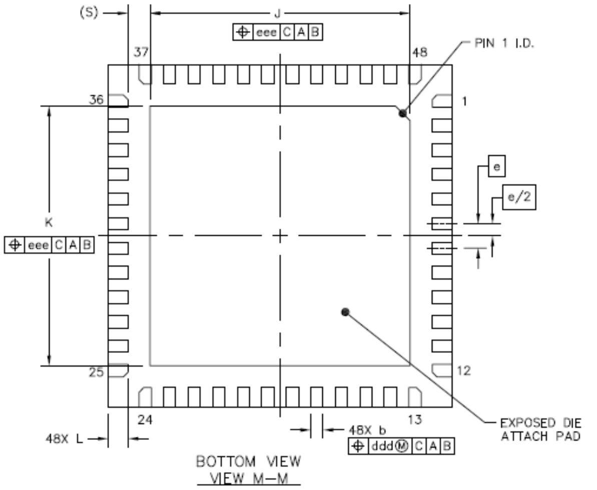
| SYMBOL | MILLIMETER | | | INCH | | |
|---|---|---|---|---|---|---|
| | MIN. | NOM. | MAX. | MIN. | NOM. | MAX. |
| A | — | — | 1.60 | — | — | 0.063 |
| A1 | 0.05 | 0.10 | 0.15 | 0.002 | 0.004 | 0 006 |
| A2 | 1.35 | 1.40 | 1.45 | 0.053 | 0.055 | 0.057 |
| D1 | 6.90 | 7.00 | 7.10 | 0.272 | 0.276 | 0.280 |
| E1 | 6.90 | 7.00 | 7.10 | 0.272 | 0.276 | 0.280 |
| e | 0.35 | 0.50 | 0.65 | 0.014 | 0.020 | 0.026 |
| D | 8.9 | 9.00 | 9.10 | 0.350 | 0.354 | 0.358 |
| E | 8 9 | 9 00 | 9.10 | 0.350 | 0 354 | 0.358 |
| L | 0.45 | 0.60 | 0.75 | 0.018 | 0.024 | 0.030 |
| L1 | — | 1.00 | — | — | 0.039 | — |
| C | 0.09 | — | 0.20 | 0.004 | — | 0.0079 |
| θ | 0° | — | 7° | 0° | — | 7° |
| b | 0.17 | 0.22 | 0.27 | 0.007 | 0.0087 | 0.011 |



## 8.2 QFN 48 (7x7x0.8mm)



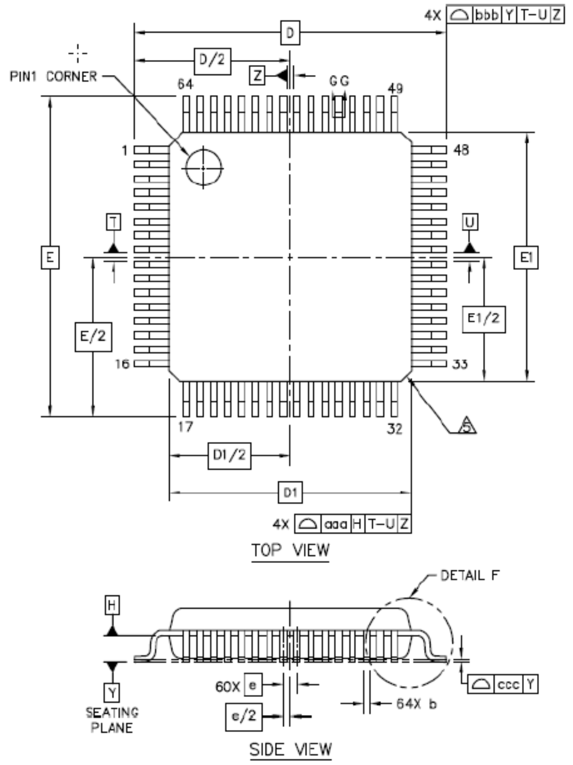
TOP VIEW



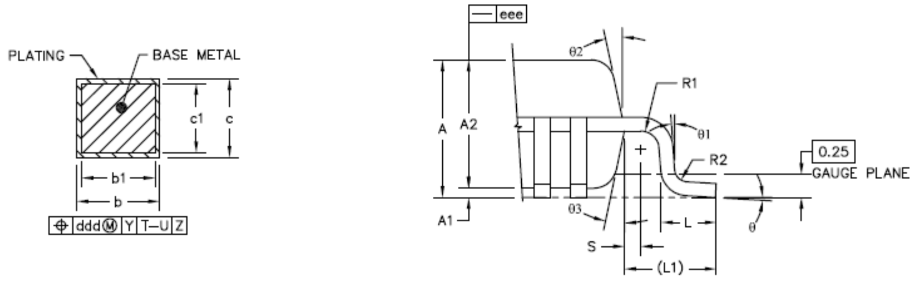
BOTTOM VIEW
VIEW M–M

| | | SYMBOL | MIN | NOM | MAX |
|---|---|---|---|---|---|
| TOTAL THICKNESS | | A | 0.7 | 0.75 | 0.8 |
| STAND OFF | | A1 | 0 | 0.035 | 0.05 |
| MOLD THICKNESS | | A2 | --- | 0.55 | --- |
| L/F THICKNESS | | A3 | | 0.203 REF | |
| LEAD WIDTH | | b | 0.2 | 0.25 | 0.3 |
| BODY SIZE | X | D | | 7 BSC | |
| | Y | E | | 7 BSC | |
| LEAD PITCH | | e | | 0.5 BSC | |
| EP SIZE | X | J | 5.2 | 5.3 | 5.4 |
| | Y | K | 5.2 | 5.3 | 5.4 |
| LEAD LENGTH | | L | 0.35 | 0.4 | 0.45 |
| PACKAGE EDGE TOLERANCE | | aaa | | 0.1 | |
| MOLD FLATNESS | | bbb | | 0.1 | |
| COPLANARITY | | ccc | | 0.08 | |
| LEAD OFFSET | | ddd | | 0.1 | |
| EXPOSED PAD OFFSET | | eee | | 0.1 | |
| | | S | | 0.45 REF | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

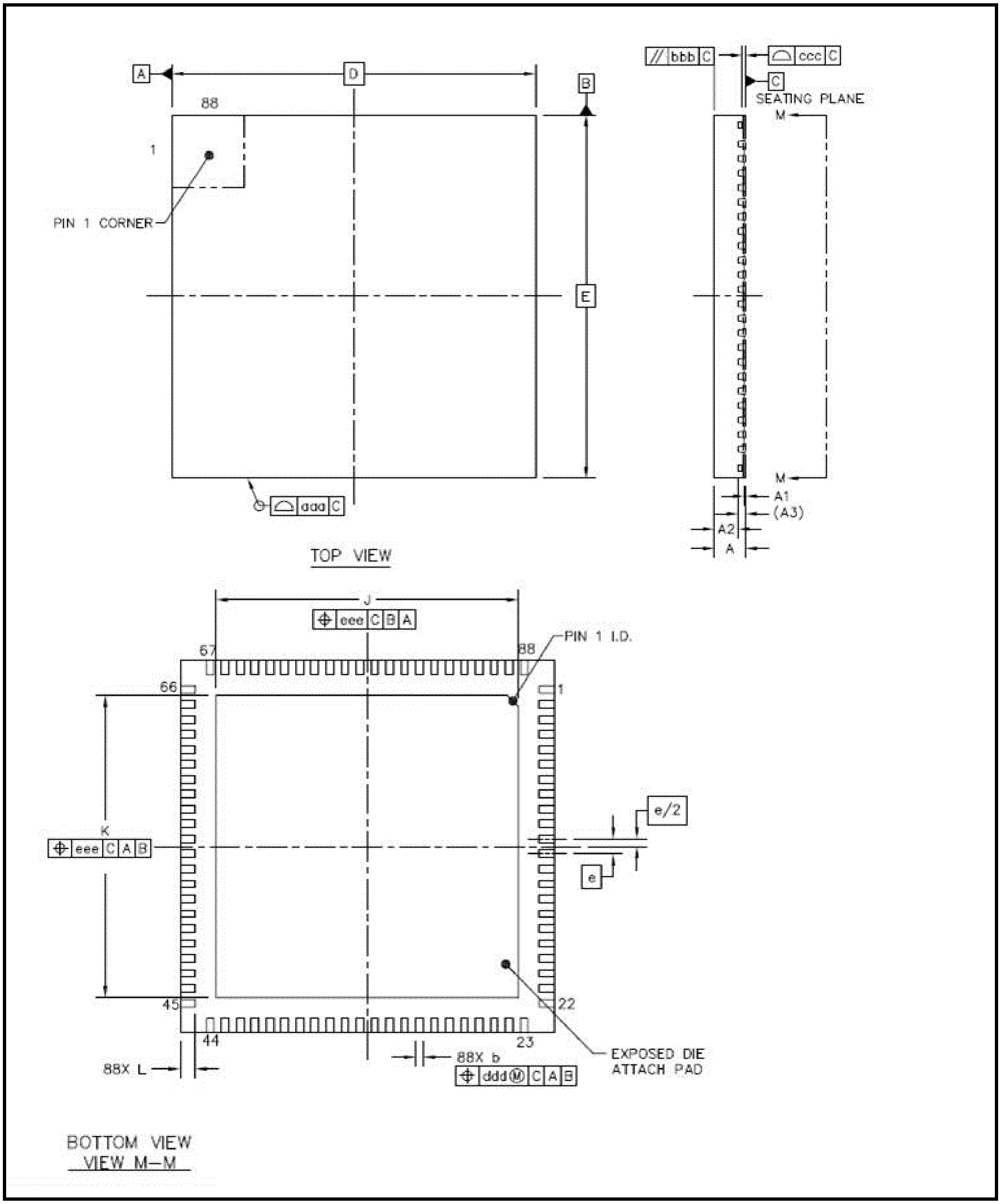
### 8.3 LQFP 64L (7x7x1.4mm footprint 2.0mm)

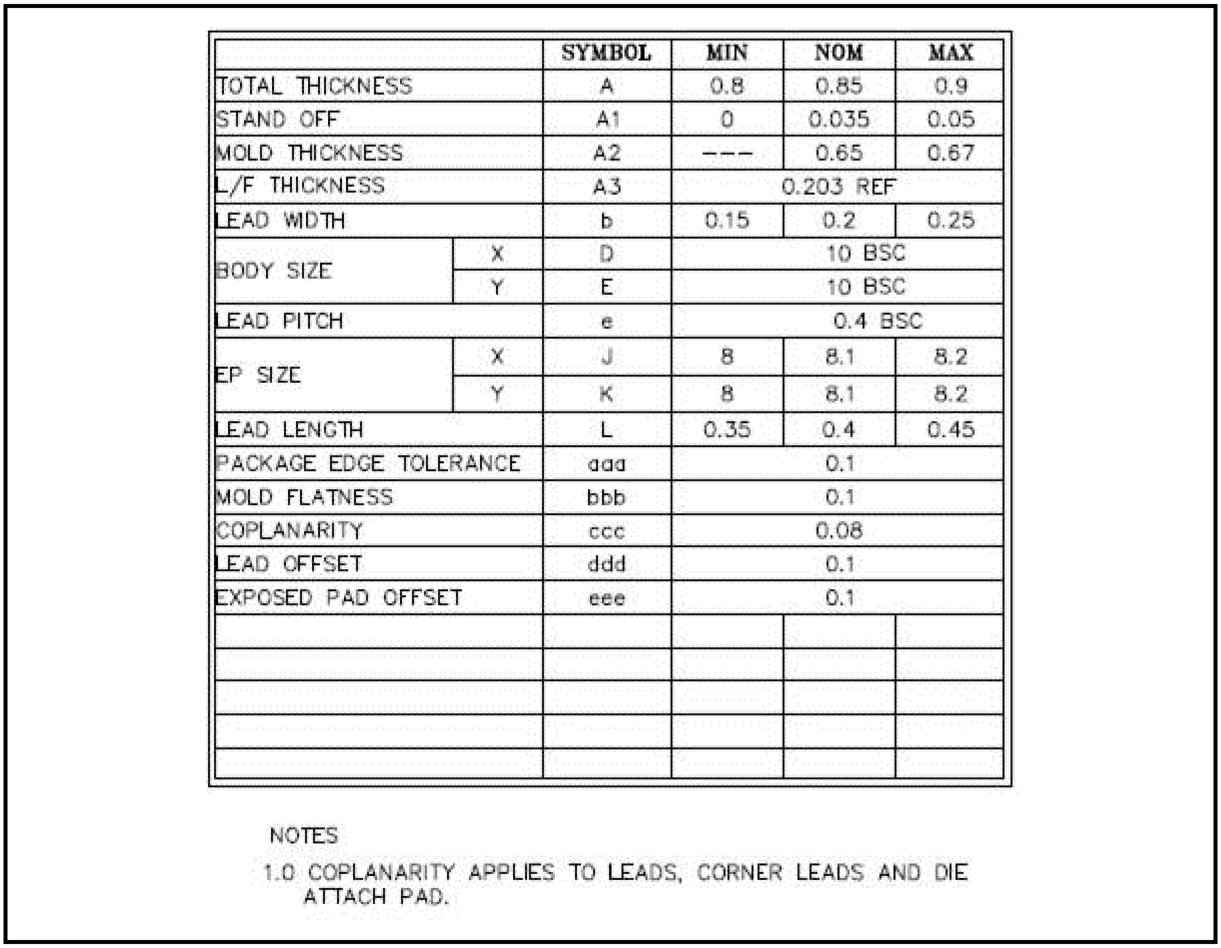


| | | SYMBOL | MIN | NOM | MAX |
|---|---|---|---|---|---|
| TOTAL THICKNESS | | A | --- | --- | 1.6 |
| STAND OFF | | A1 | 0.05 | --- | 0.15 |
| MOLD THICKNESS | | A2 | 1.35 | 1.4 | 1.45 |
| LEAD WIDTH(PLATING) | | b | 0.13 | 0.18 | 0.23 |
| LEAD WIDTH | | b1 | 0.13 | 0.16 | 0.19 |
| L/F THICKNESS(PLATING) | | 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 | | |



## 8.4 QFN 88 (10x10x0.9mm)





| | | SYMBOL | MIN | NOM | MAX |
|---|---|---|---|---|---|
| TOTAL THICKNESS | | A | 0.8 | 0.85 | 0.9 |
| STAND OFF | | A1 | 0 | 0.035 | 0.05 |
| MOLD THICKNESS | | A2 | --- | 0.65 | 0.67 |
| L/F THICKNESS | | A3 | | 0.203 REF | |
| LEAD WIDTH | | b | 0.15 | 0.2 | 0.25 |
| BODY SIZE | X | D | | 10 BSC | |
| | Y | E | | 10 BSC | |
| LEAD PITCH | | e | | 0.4 BSC | |
| EP SIZE | X | J | 8 | 8.1 | 8.2 |
| | Y | K | 8 | 8.1 | 8.2 |
| LEAD LENGTH | | L | 0.35 | 0.4 | 0.45 |
| PACKAGE EDGE TOLERANCE | | aaa | | 0.1 | |
| MOLD FLATNESS | | bbb | | 0.1 | |
| COPLANARITY | | ccc | | 0.08 | |
| LEAD OFFSET | | ddd | | 0.1 | |
| EXPOSED PAD OFFSET | | eee | | 0.1 | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

NOTES

1.0 COPLANARITY APPLIES TO LEADS, CORNER LEADS AND DIE ATTACH PAD.

# 9 REVISION HISTORY

| Date | Revision | Description |
|---|---|---|
| 2014.04.23 | 1.01 | 1. Preliminary version |
| 2015.05.08 | 1.02 | 1. Added  Figure 6.14 3 Internal CODEC Block Diagram. |
| 2015.05.28 | 1.03 | 1. Added new part number: NUC505DLA, NUC505YLA, and NUC505DSA in Chapter 4.<br>2. Updated embedded SPI Flash memory size to 512 KB for new part number. |
| 2015.11.04 | 1.04 | 1. Added a note to indicate that NUC505DS13Y only supports Headphone Out in section 4.1.1.<br>2. Added a note to indicate the packages are not pin-to-pin compatible in section 4.1.1.<br>3. Added section 8.2 QFN 48 (7x7x0.8mm) package specification.<br>4. Added part number NUC505YLA2Y in section 4.1.1, 4.2.4, and 4.3.4.<br>5. Added bit field description for LOWPWREN (USBD_PHYCTL[3]) in section 6.15.6.<br>6. Replaced power mode name of Sleep mode and Deep-sleep mode with Idle mode and Power-down mode respectively. |
| 2016.05.09 | 1.05 | 1. Added a note to Pin Diagram and Pin Description for QFN 48/88-pin packages. |
| 2016.12.02 | 1.06 | 1. Corrected the typo in the Pin Configuration section 4.2.4/4.2.5/4.2.6 and Pin Description section 4.3.5.<br>2. Modified section 4.1.1 NUC505DLA and NUC505YLA SPI should be two.<br>3. Modified pin description from VBUS to VBUS33. |
| 2017.08.10 | 1.07 | 1. Modified VBUS33 pin description.<br>2. Modified USB transceiver power description in section 6.2.4.<br>3. Fixed the CODEC register typo in section 6.14.9.<br>4. Modified USB Host clock source only from PLL in section 6.3.2.<br>5. Modified VCMBF pin description in section 4.3.7.<br>6. Modified description field of register HCFS in section 6.16.7.<br>7. Added SPIM register description in section 6.13.6 |
| 2019.01.16 | 1.08 | 1. Added RTC enable flow in section 6.9.5.<br>2. Modified Figure 6.3 1 Clock Generator Global View Diagram. |

## 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.**