



UM10850

LPC5410x User manual

Rev. 2.0 — 10 April 2015

User manual

Document information

| Info | Content |
|-----------------|--|
| Keywords | LPC5410x, ARM Cortex-M4, ARM Cortex-M0+, microcontroller, sensor hub |
| Abstract | LPC5410x User Manual |



Revision history

| Rev | Date | Description |
|-----|----------|---|
| 2.0 | 20150410 | <ul style="list-style-type: none">Registers supporting use of dual processors on LPC54102 devices has been added to the Syscon chapter.A Power Management chapter has been added.Some pins in the Pin description chapter have the type changed to Z for open drain pins.A section has been added to the ADC chapter describing how to configure sample times for different conversion configurations.The ADC operating speed is increased to 5 Ms/s.Updated Section 4.5.33 “Flash configuration register” text and Table 62.Typographic errors have been corrected and minor pieces of information added or clarified throughout the document. |
| 1.1 | 20141121 | <ul style="list-style-type: none">In the NVIC chapter, the bit numbers for the priority register fields have been corrected.In the Syscon chapter, a functional description section has been added following the register descriptions to provide additional information about some functions.The SCTimer/PWM chapter has been revised to better explain the function.Description of the ISP-AP interface and commands is added to the Debug chapter.References to Timer 0, 1, 2, 3, and 4 have been updated to use the terminology of the Standard counter/timers chapter (CT32B0, 1, 2, 3, 4).Typographic errors have been corrected and minor pieces of information added or clarified throughout the document. |
| 1.0 | 20141104 | Initial release of the LPC5410x User Manual |

Contact information

For more information, please visit: <http://www.nxp.com>

For sales office addresses, please send an email to: salesaddresses@nxp.com

1.1 Introduction

The LPC5410x are ARM Cortex-M4 based microcontrollers for embedded applications. these devices include an optional ARM Cortex-M0+ coprocessor, 104 KB of on-chip SRAM, 512 KB on-chip flash, five general-purpose timers, one State-Configurable Timer with PWM capabilities (SCTimer/PWM), one RTC/alarm timer, one 24-bit Multi-Rate Timer (MRT), a Windowed Watchdog Timer (WWDT), four USARTs, two SPIs, three Fast-mode plus I²C-bus interfaces with high-speed slave mode, and one 12-bit 5 Msamples/sec ADC.

The ARM Cortex-M4 is a 32-bit core that offers system enhancements such as low power consumption, enhanced debug features, and a high level of support block integration. The ARM Cortex-M4 CPU incorporates a 3-stage pipeline, uses a Harvard architecture with separate local instruction and data buses as well as a third bus for peripherals, and includes an internal prefetch unit that supports speculative branching. The ARM Cortex-M4 supports single-cycle digital signal processing and SIMD instructions. The Cortex-M4 is the Cortex-M4 with the inclusion of the 32-bit Floating Point Unit.

The ARM Cortex-M0+ coprocessor available on some devices is an energy-efficient and easy-to-use 32-bit core which is code- and tool-compatible with the Cortex-M4 core. The Cortex-M0+ coprocessor offers up to 100 MHz performance with a simple instruction set and reduced code size.

Refer to LPC5410x data sheets for complete details on specific products and configurations.

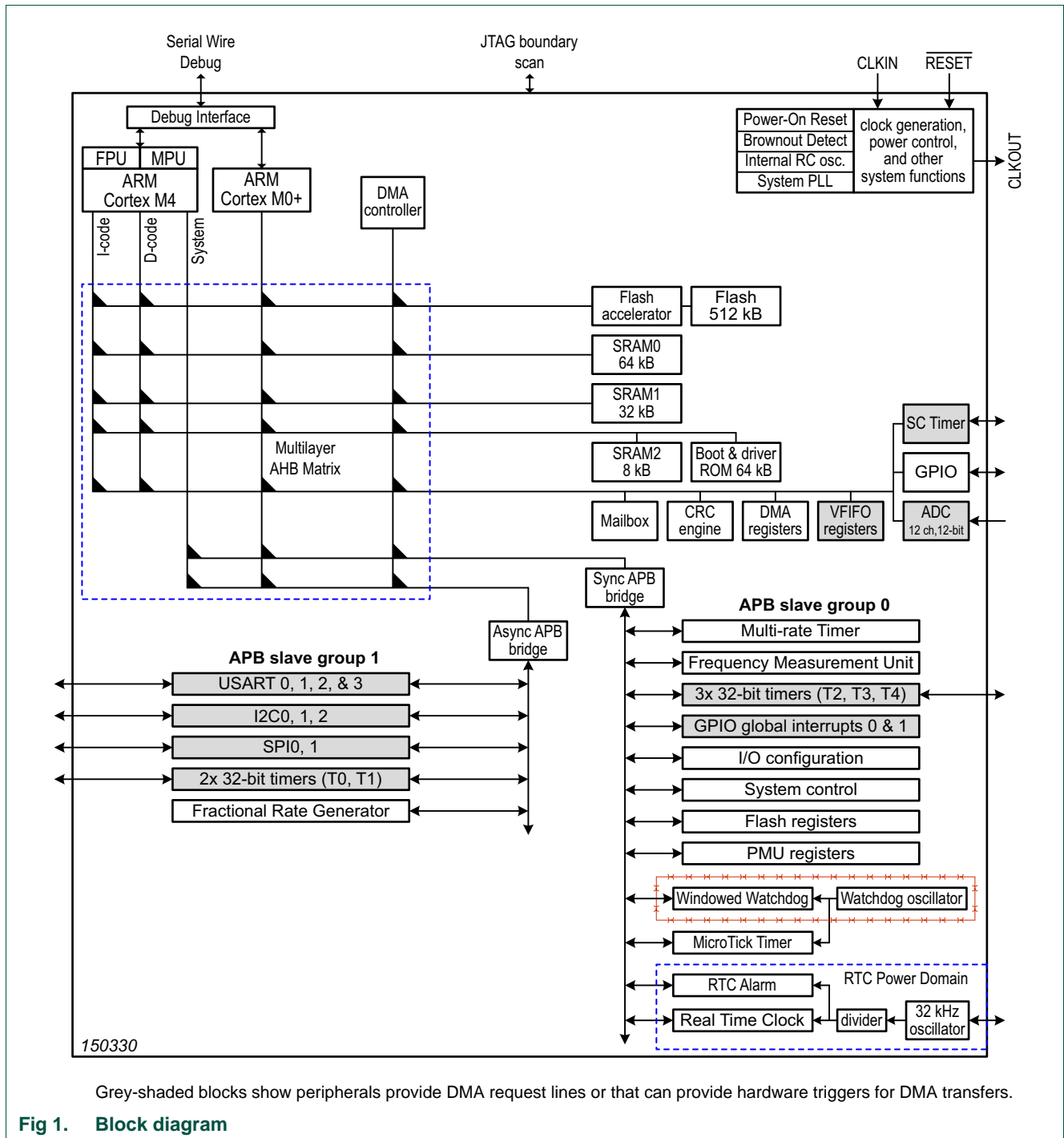
1.2 Features

- Dual processor core: ARM Cortex-M4 and ARM Cortex-M0+ included for LPC54102 devices. Cortex-M4 only is present on LPC54101 devices.
- ARM Cortex-M4 CPU:
 - ARM Cortex-M4 processor, running at a frequency of up to 100 MHz.
 - Floating Point Unit (FPU) and Memory Protection Unit (MPU).
 - The CPU can operate at frequencies of up to 100 MHz.
 - ARM Cortex-M4 built-in Nested Vectored Interrupt Controller (NVIC).
 - Non-maskable Interrupt (NMI) with a selection of sources.
 - Serial Wire Debug (SWD) with 8 breakpoints and 4 watchpoints. Includes Serial Wire Output for enhanced debug capabilities.
 - System tick timer.
- ARM Cortex-M0+ CPU (present on LPC54102 devices):
 - ARM Cortex-M0+ processor, running at a frequency of up to 100 MHz (using the same clock as the Cortex-M4).
 - The CPU can operate at frequencies of up to 100 MHz.
 - ARM Cortex-M0+ built-in Nested Vectored Interrupt Controller (NVIC).

- Non-maskable Interrupt (NMI) with a selection of sources.
- Serial Wire Debug (SWD) with 4 breakpoints and 2 watchpoints.
- System tick timer.
- On-Chip memory:
 - Up to 512 KB on-chip flash programming memory with flash accelerator and 256 Byte page write and erase.
 - Up to 104 KB total SRAM composed of:
 - Up to 96 KB contiguous main SRAM.
 - An additional 8 KB SRAM.
- ROM API support:
 - Flash In-Application Programming (IAP) and In-System Programming (ISP).
 - Power Control API.
- Serial interfaces:
 - Four USART interfaces with synchronous mode and 32 kHz mode for wake-up from Deep-sleep and Power-down modes. The USARTs include a FIFO buffer, and share a fractional baud-rate generator.
 - Two SPI interfaces, each with 4 slave selects and flexible data configuration. The SPIs include a FIFO buffer. Able to wake up the device from Deep-sleep and Power-down modes when used in slave mode.
 - Three I²C-bus interfaces supporting fast mode and Fast-mode Plus with data rates of up to 1Mbit/s and with multiple address recognition and monitor mode. Each I²C-bus interface also supports High Speed Mode as a slave. The slave function is able to wake up the device from Deep-sleep and Power-down modes.
- Digital peripherals:
 - DMA controller with 22 channels and 20 programmable triggers, able to access all memories and DMA-capable peripherals.
 - Up to 50 General-Purpose I/O (GPIO) pins. Most GPIOs have configurable pull-up/pull-down resistors, open-drain mode, and input inverter.
 - GPIO registers are located on AHB for fast access.
 - Up to eight GPIOs can be selected as pin interrupts (PINT), triggered by rising, falling or both input edges.
 - Two GPIO grouped interrupts (GINT) enable an interrupt based on a logical (AND/OR) combination of input states.
 - CRC engine.
- Timers
 - Five 32-bit standard general purpose timers/counters, four of which support up to 4 capture inputs and 4 compare outputs, PWM mode, and external count input. Specific timer events can be selected to generate DMA requests. The fifth timer does not have external pin connections and may be used for internal timing operations.
 - One State Configurable Timer/PWM (SCTimer/PWM) 6 input and 8 output functions (including capture and match). Inputs and outputs can be routed to/from external pins and internally to/from selected peripherals. Internally, the SCT supports 13 captures/matches, 13 events and 13 states.

- 32-bit Real-time clock (RTC) with 1 s resolution running in the always-on power domain. A timer in the RTC can be used for wake-up from all low power modes including Deep power-down, with 1 ms resolution.
- Multiple-channel multi-rate 24-bit timer (MRT) for repetitive interrupt generation at up to four programmable, fixed rates.
- Windowed Watchdog timer (WWDT).
- Ultra-low power Micro-tick Timer, running from the Watchdog oscillator, that can be used to wake up the device from low power modes.
- Repetitive interrupt timer for general purpose use and use with debug time-stamping.
- Analog peripheral: 12-bit ADC with 12 input channels and with multiple internal and external trigger inputs and sample rates of up to 5 MS/s. The ADC supports two independent conversion sequences.
- Clock generation:
 - 12 MHz internal RC oscillator, factory trimmed for accuracy, that can optionally be used as a system clock.
 - External clock input for up to 24 MHz.
 - Internal, low-power, watchdog oscillator (WDOSC) with a nominal frequency of 500 kHz.
 - 32 kHz low-power RTC oscillator.
 - System PLL allows CPU operation up to the maximum CPU rate without the need for a high-frequency external clock. May be run from the internal RC oscillator, the external clock input CLKIN, or the RTC oscillator.
 - Clock output function with divider that can reflect many internal clocks.
 - Frequency measurement unit for measuring the frequency of any on-chip or off-chip clock signal.
- Power control:
 - Integrated PMU (Power Management Unit) to minimize power consumption.
 - Reduced power modes: Sleep mode, Deep-sleep mode, Power-down mode, and Deep power-down mode.
 - Wake-up from Deep-sleep and Power-down modes on activity on USART, SPI, and I²C peripherals.
 - Wake-up from Sleep, Deep-sleep, Power-down, and Deep power-down modes from the RTC alarm.
 - Power-On Reset (POR).
 - Brownout detect.
- JTAG boundary scan supported.
- Unique device serial number (128-bit) for identification.
- Single power supply 1.62 V to 3.6 V.
- Operating temperature range of -40°C to +105°C.
- Available in a 3.288 x 3.288 mm WLCSP49 package and LQFP64 package.

1.3 Block diagram



1.4 Architectural overview

The ARM Cortex-M4 includes three AHB-Lite buses, one system bus and the I-code and D-code buses. One bus is dedicated for instruction fetch (I-code), and one bus is dedicated for data access (D-code). The use of two core buses allows for simultaneous operations if concurrent operations target different devices.

A multi-layer AHB matrix connects the CPU buses and other bus masters to peripherals in a flexible manner that optimizes performance by allowing peripherals on different slaves ports of the matrix to be accessed simultaneously by different bus masters. More information on the multilayer matrix can be found in [Section 2.1.3](#). Connections in the multilayer matrix are shown in [Figure 1](#).

APB peripherals are connected to the AHB matrix via two APB buses using separate slave ports from the multilayer AHB matrix. This allows for better performance by reducing collisions between the CPU and the DMA controller, and also for peripherals on the asynchronous bridge to have a fixed clock that does not track the system clock.

1.5 ARM Cortex-M4 processor

The Cortex-M4 is a general purpose 32-bit microprocessor, which offers high performance and very low power consumption. The Cortex-M4 offers a Thumb-2 instruction set, low interrupt latency, interruptible/continuable multiple load and store instructions, automatic state save and restore for interrupts, tightly integrated interrupt controller, and multiple core buses capable of simultaneous accesses.

A 3-stage pipeline is employed so that all parts of the processing and memory systems can operate continuously. Typically, while one instruction is being executed, its successor is being decoded, and a third instruction is being fetched from memory.

Information about Cortex-M4 configuration options can be found in [Chapter 33](#).

1.6 ARM Cortex-M0+ processor (present on LPC54102 devices)

The Cortex-M0+ is a general purpose 32-bit microprocessor with extremely low power consumption. The Cortex-M0+ includes the bulk of the Thumb instruction set and a small subset of Thumb-2 Instructions. The Cortex-M0+ has a 2-stage pipeline in order to decrease power consumption, and includes a 32-cycle multiplier.

Information about Cortex-M0+ configuration options can be found in [Chapter 33](#).

2.1 General description

The LPC5410x incorporates several distinct memory regions. [Figure 2](#) shows the overall map of the entire address space from the user program viewpoint following reset.

The APB peripheral area is 512 KB in size and is divided to allow for up to 32 peripherals. Each peripheral is allocated 16 KB of space, simplifying the address decoding.

The registers incorporated into the CPU, such as NVIC, SysTick, and sleep mode control, are located on the private peripheral bus.

2.1.1 Main SRAM

The parts contain up to a total 96 KB of contiguous, on-chip static RAM memory (this is in addition to SRAM2 as noted in the next section below, so the total device SRAM can be up to 104 KB). For each SRAM configuration, the SRAM is divided into two blocks: SRAM0 (up to 64 KB) and SRAM1 (up to 32 KB). The bottom 8 KB of SRAM can be enabled separately in order to allow saving data with minimal power usage during Power-down mode. The remaining portion of SRAM0 and the entire SRAM1 can also be disabled or enabled individually in the SYSCON block to save power. See [Section 4.5.22 “AHB Clock Control register 0”](#) and [Section 4.5.38 “Power Configuration register”](#).

Table 1. Main SRAM configuration

| | SRAM0 | SRAM1 |
|--|--|--|
| (total main SRAM = up to 96 KB) | | |
| Size | Up to 64 KB | Up to 32 KB |
| Address range | <ul style="list-style-type: none"> Always begins at 0x0200 0000. Continues to 0x0200 FFFF (for full 64 KB). | <ul style="list-style-type: none"> Begins at end of SRAM0, 0x0201 0000 when SRAM0 is a full 64 KB. Ends at 0x0201 7FFF for 32 KB SRAM1 with 64 KB SRAM0. |
| Power Control (via Power API) | <ul style="list-style-type: none"> First 8 KB is has a separate power switch. Remaining SRAM0 has a single power switch. | <ul style="list-style-type: none"> All of SRAM1 has a single power switch. |

2.1.1.1 SRAM2

An additional on-chip static RAM memory, SRAM2, is available that is not contiguous to SRAM0 and SRAM1. This can be used, for example, as the location for the program stack, or any other use. SRAM2 can be disabled or enabled in the SYSCON block to save power. See [Section 4.5.22 “AHB Clock Control register 0”](#) and [Section 4.5.38 “Power Configuration register”](#).

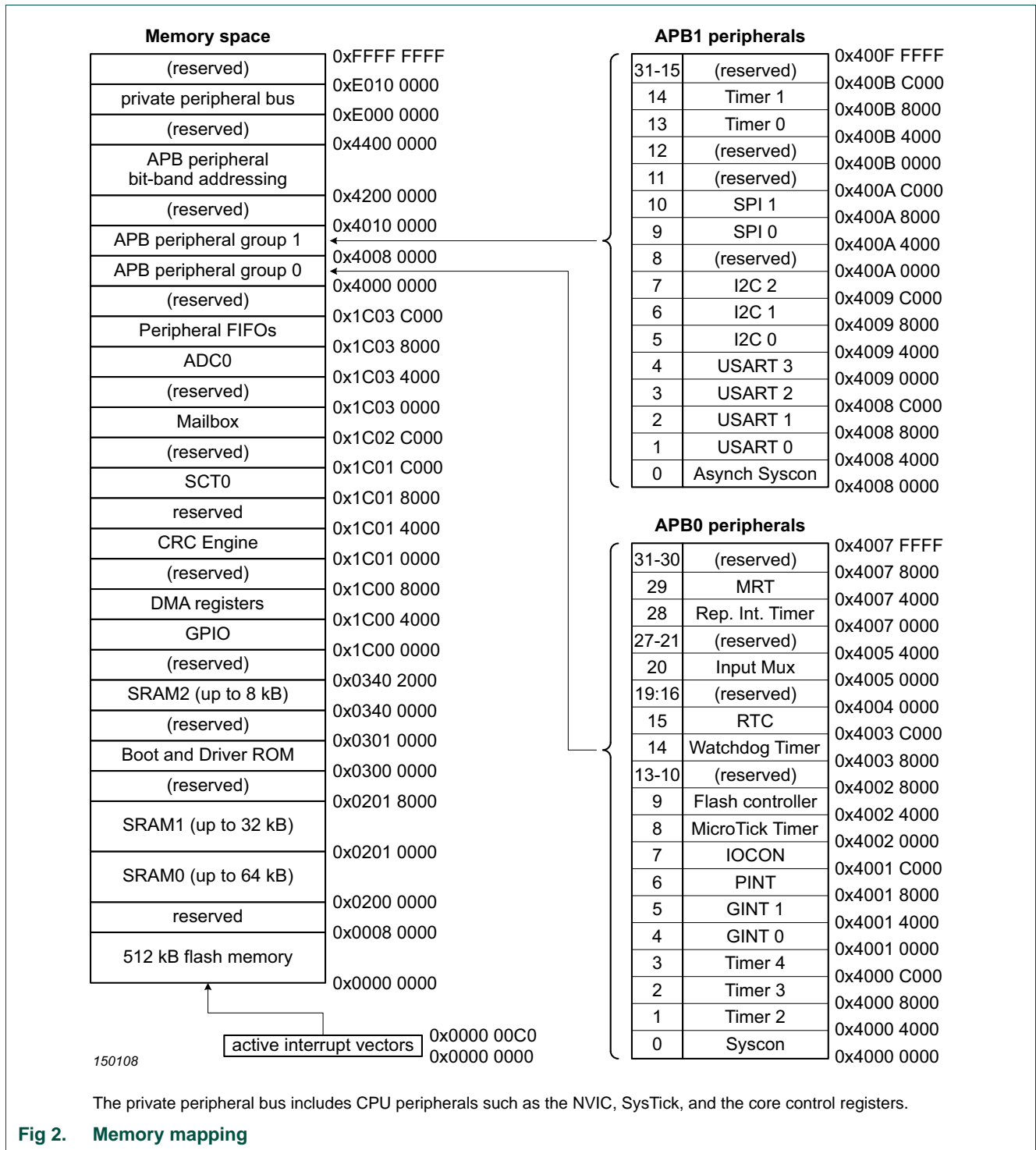
2.1.1.2 SRAM usage notes

Although always contiguous on all LPC5410x devices, SRAM0 and SRAM1 are placed on different AHB matrix ports. This allows user programs to potentially obtain better performance by dividing RAM usage among the 2 ports. For example, simultaneous access to SRAM0 by the CPU and SRAM1 by the system DMA controller does not result in any bus stalls for either master.

Generally speaking, the CPU will read or write all peripheral data at some point, even when all such data is read from or sent to a peripheral by DMA. So, minimizing stalls is likely to involve putting data to/from different peripherals in RAM on each port.

Alternatively, sequences of data from the same peripheral could be alternated between RAM on each port. this could be helpful if DMA fills or empties a RAM buffer, then signals the CPU before proceeding on to a second buffer. the CPU would then tend to access the data while the DMA is using the other RAM.

2.1.2 Memory mapping



2.1.3 AHB multilayer matrix

The LPC5410x uses a multi-layer AHB matrix to connect the CPU buses and other bus masters to peripherals in a flexible manner that optimizes performance by allowing peripherals that are on different slave ports of the matrix to be accessed simultaneously by different bus masters. [Figure 1](#) shows details of the potential matrix connections.

2.1.4 Memory Protection Unit (MPU)

The Cortex-M4 processor has a memory protection unit (MPU) that provides fine grain memory control, enabling applications to implement security privilege levels, separating code, data and stack on a task-by-task basis. Such requirements are critical in many embedded applications.

The MPU register interface is located on the private peripheral bus and is described in detail in [Ref. 1 "Cortex-M4 TRM"](#).

3.1 How to read this chapter

Available interrupt sources may vary with specific LPC5410x device type.

3.2 Features

- Nested Vectored Interrupt Controller that is an integral part of each CPU.
- Tightly coupled interrupt controller provides low interrupt latency.
- Controls system exceptions and peripheral interrupts.
- The NVIC of the Cortex-M4 supports:
 - 37 vectored interrupts.
 - 8 programmable interrupt priority levels with hardware priority level masking.
 - Vector table offset register VTOR.
 - Software interrupt generation.
- The Cortex-M0+ (present on LPC54102 devices) supports: the first 32 interrupts.
 - 32 vectored interrupts.
 - 4 programmable interrupt priority levels with hardware priority level masking.
 - Vector table offset register VTOR.
- Support for NMI from any interrupt (see [Section 4.5.3](#)).

3.3 General description

The tight coupling to the NVIC to the CPU allows for low interrupt latency and efficient processing of late arriving interrupts.

3.3.1 Interrupt sources

[Table 2](#) lists the interrupt sources for each peripheral function. Each peripheral device may have one or more interrupt lines to the Vectored Interrupt Controller. Each line may represent more than one interrupt source. The interrupt number does not imply any interrupt priority.

See [Ref. 1 “Cortex-M4 TRM”](#) and [Ref. 2 “Cortex-M0+ TRM”](#) for detailed descriptions of the NVIC and the NVIC registers.

Table 2. Connection of interrupt sources to the NVIC

| Interrupt | Name | Description | Flags |
|-----------|------------|-----------------------------------|--|
| 0 | WDT | Windowed watchdog timer interrupt | WARNINT - watchdog warning interrupt |
| 1 | BOD | BOD interrupt | BODINTVAL - BOD interrupt level |
| 2 | (reserved) | - | - |
| 3 | DMA | DMA interrupts | Interrupt A and interrupt B, error interrupt |

Table 2. Connection of interrupt sources to the NVIC

| Interrupt | Name | Description | Flags |
|---|------------|---|--|
| 4 | GINT0 | GPIO group 0 interrupt | Enabled pin interrupts |
| 5 | PIN_INT0 | Pin interrupt 0 or pattern match engine slice 0 int | PSTAT - pin interrupt status |
| 6 | PIN_INT1 | Pin interrupt 1 or pattern match engine slice 1 int | PSTAT - pin interrupt status |
| 7 | PIN_INT2 | Pin interrupt 2 or pattern match engine slice 2 int | PSTAT - pin interrupt status |
| 8 | PIN_INT3 | Pin interrupt 3 or pattern match engine slice 3 int | PSTAT - pin interrupt status |
| 9 | UTICK | Micro-tick Timer interrupt | INTR |
| 10 | MRT | Multi-rate timer interrupt | Global MRT interrupts: GFLAG0, 1, 2, 3 |
| 11 | CT32B0 | Standard counter/timer CT32B0 interrupt | Match and Capture interrupts |
| 12 | CT32B1 | Standard counter/timer CT32B1 interrupt | Match and Capture interrupts |
| 13 | CT32B2 | Standard counter/timer CT32B2 interrupt | Match and Capture interrupts |
| 14 | CT32B3 | Standard counter/timer CT32B3 interrupt | Match and Capture interrupts |
| 15 | CT32B4 | Standard counter/timer CT32B4 interrupt | Match and Capture interrupts |
| 16 | SCT0 | State configurable timer interrupt | EVFLAG SCT event |
| 17 | UART0 | USART0 interrupt | See Table 307 . |
| 18 | UART1 | USART1 interrupt | Same as USART0 |
| 19 | UART2 | USART2 interrupt | Same as USART0 |
| 20 | UART3 | USART3 interrupt | Same as USART0 |
| 21 | I2C0 | I2C0 interrupt | See Table 345 . |
| 22 | I2C1 | I2C1 interrupt | Same as I2C0 |
| 23 | I2C2 | I2C2 interrupt | Same as I2C0 |
| 24 | SPI0 | SPI0 interrupt | See Table 322 . |
| 25 | SPI1 | SPI1 interrupt | Same as SPI0 |
| 26 | ADC0_SEQA | ADC0 sequence A completion. | See Table 425 . |
| 27 | ADC0_SEQB | ADC0 sequence B completion. | See Table 425 . |
| 28 | ADC0_THCMP | ADC0 threshold compare and error. | See Table 425 . |
| 29 | RTC | RTC alarm and wake-up interrupts | See Table 273 . |
| 30 | (reserved) | - | - |
| 31 | MAILBOX | Mailbox interrupt (present on LPC54102 devices) | Mailbox Interrupt |
| The following interrupts are supported only on the Cortex-M4 | | | |
| 32 | GINT1 | GPIO group 1 interrupt | Enabled pin interrupts |
| 33 | PIN_INT4 | Pin interrupt 4 or pattern match engine slice 4 int | PSTAT - pin interrupt status |
| 34 | PIN_INT5 | Pin interrupt 5 or pattern match engine slice 5 int | PSTAT - pin interrupt status |
| 35 | PIN_INT6 | Pin interrupt 6 or pattern match engine slice 6 int | PSTAT - pin interrupt status |
| 36 | PIN_INT7 | Pin interrupt 7 or pattern match engine slice 7 int | PSTAT - pin interrupt status |
| 39:37 | (reserved) | - | - |
| 40 | RIT | Repetitive Interrupt Timer | RITINT; masked compare interrupt |

3.4 Register description

The NVIC registers are located on the ARM private peripheral bus.

Table 3. Register overview: NVIC (base address 0xE000 E000)

| Name | Access | Address offset | Description | Reset value | Reference |
|----------------------|--------|----------------|---|-------------|--------------------------|
| ISER0 | R/W | 0x100 | Interrupt Set Enable Register 0. This register allows enabling interrupts and reading back the interrupt enables for peripheral functions. | 0 | Table 4 |
| ISER1 | R/W | 0x104 | Interrupt Set Enable Register 1. See ISER0 description. | 0 | Table 5 |
| ICER0 | R/W | 0x180 | Interrupt Clear Enable Register 0. This register allows disabling interrupts and reading back the interrupt enables for peripheral functions. | 0 | Table 6 |
| ICER1 | R/W | 0x184 | Interrupt Clear Enable Register 1. See ISER0 description. | 0 | Table 7 |
| ISPR0 | R/W | 0x200 | Interrupt Set Pending Register 0. This register allows changing the interrupt state to pending and reading back the interrupt pending state for peripheral functions. | 0 | Table 8 |
| ISPR1 | R/W | 0x204 | Interrupt Set Pending Register 1. See ISPR0 description. | 0 | Table 9 |
| ICPR0 | R/W | 0x280 | Interrupt Clear Pending Register 0. This register allows changing the interrupt state to not pending and reading back the interrupt pending state for peripheral functions. | 0 | Table 10 |
| ICPR1 | R/W | 0x284 | Interrupt Clear Pending Register 1. See ICPR0 description. | 0 | Table 11 |
| IABR0 ^[1] | RO | 0x300 | Interrupt Active Bit Register 0. This register allows reading the current interrupt active state for specific peripheral functions. | 0 | Table 12 |
| IABR1 ^[1] | RO | 0x304 | Interrupt Active Bit Register 1. See IABR0 description. | 0 | Table 13 |
| IPR0 | R/W | 0x400 | Interrupt Priority Register 0. This register contains the 3-bit priority fields for interrupts 0 to 3. | 0 | Table 14 |
| IPR1 | R/W | 0x404 | Interrupt Priority Register 1. This register contains the 3-bit priority fields for interrupts 4 to 7. | 0 | Table 15 |
| IPR2 | R/W | 0x408 | Interrupt Priority Register 2. This register contains the 3-bit priority fields for interrupts 8 to 11. | 0 | Table 16 |
| IPR3 | R/W | 0x40C | Interrupt Priority Register 3. This register contains the 3-bit priority fields for interrupts 12 to 15. | 0 | Table 17 |
| IPR4 | R/W | 0x410 | Interrupt Priority Register 4. This register contains the 3-bit priority fields for interrupts 16 to 19. | 0 | Table 18 |
| IPR5 | R/W | 0x414 | Interrupt Priority Register 5. This register contains the 3-bit priority fields for interrupts 20 to 23. | 0 | Table 19 |
| IPR6 | R/W | 0x418 | Interrupt Priority Register 6. This register contains the 3-bit priority fields for interrupts 24 to 27. | 0 | Table 20 |
| IPR7 | R/W | 0x41C | Interrupt Priority Register 7. This register contains the 3-bit priority fields for interrupts 28 to 31. | 0 | Table 21 |
| IPR8 | R/W | 0x420 | Interrupt Priority Register 8. This register contains the 3-bit priority fields for interrupts 32 to 35. | 0 | Table 22 |
| IPR9 | R/W | 0x424 | Interrupt Priority Register 9. This register contains the 3-bit priority fields for interrupts 36 to 39. | 0 | Table 23 |
| IPR10 | R/W | 0x428 | Interrupt Priority Register 10. This register contains the 3-bit priority field for interrupt 40. | 0 | Table 24 |
| STIR ^[1] | WO | 0xF00 | Software Trigger Interrupt Register, allows software to generate interrupts. | - | Table 25 |

[1] This register is not available for the Cortex-M0+.

3.4.1 Interrupt Set-Enable Register 0 register

The ISER0 register allows enabling the first 32 peripheral interrupts, or for reading the enabled state of those interrupts. The remaining interrupts are enabled via the ISER1 register ([Section 3.4.2](#)). Disabling interrupts is done through the ICER0 and ICER1 registers ([Section 3.4.3](#) and [Section 3.4.4](#)).

Table 4. Interrupt Set-Enable Register 0 register

| Bit | Name | Value | Function |
|-----|--------------|---------------------|---|
| 0 | ISE_WDT | [1] | Watchdog Timer interrupt enable. |
| 1 | ISE_BOD | [1] | BOD interrupt enable. |
| 2 | - | - | Reserved. Read value is undefined, only zero should be written. |
| 3 | ISE_DMA | [1] | DMA interrupt enable. |
| 4 | ISE_GINT0 | [1] | GPIO group 0 interrupt enable. |
| 5 | ISE_PINT0 | [1] | Pin interrupt / pattern match engine slice 0 interrupt. |
| 6 | ISE_PINT1 | [1] | Pin interrupt / pattern match engine slice 1 interrupt. |
| 7 | ISE_PINT2 | [1] | Pin interrupt / pattern match engine slice 2 interrupt. |
| 8 | ISE_PINT3 | [1] | Pin interrupt / pattern match engine slice 3 interrupt. |
| 9 | ISE_UTICK | [1] | Micro-Tick Timer interrupt enable. |
| 10 | ISE_MRT | [1] | Multi-Rate Timer interrupt enable. |
| 11 | ISE_CT32B0 | [1] | Standard counter/timer CT32B0 interrupt enable. |
| 12 | ISE_CT32B1 | [1] | Standard counter/timer CT32B1 interrupt enable. |
| 13 | ISE_CT32B2 | [1] | Standard counter/timer CT32B2 interrupt enable. |
| 14 | ISE_CT32B3 | [1] | Standard counter/timer CT32B3 interrupt enable. |
| 15 | ISE_CT32B4 | [1] | Standard counter/timer CT32B4 interrupt enable. |
| 16 | ISE_SCT0 | [1] | SCT0 interrupt enable. |
| 17 | ISE_USART0 | [1] | USART0 interrupt enable. |
| 18 | ISE_USART1 | [1] | USART1 interrupt enable. |
| 19 | ISE_USART2 | [1] | USART2 interrupt enable. |
| 20 | ISE_USART3 | [1] | USART3 interrupt enable. |
| 21 | ISE_I2C0 | [1] | I ² C0 interrupt enable. |
| 22 | ISE_I2C1 | [1] | I ² C1 interrupt enable. |
| 23 | ISE_I2C2 | [1] | I ² C2 interrupt enable. |
| 24 | ISE_SPI0 | [1] | SPI0 interrupt enable. |
| 25 | ISE_SPI1 | [1] | SPI1 interrupt enable. |
| 26 | ISE_ADC0SEQA | [1] | ADC0 sequence A interrupt enable. |
| 27 | ISE_ADC0SEQB | [1] | ADC0 sequence B interrupt enable. |
| 28 | ISE_ADC0THOV | [1] | ADC0 threshold and error interrupt enable. |
| 29 | ISE_RTC | [1] | Real Time Clock (RTC) interrupt enable. |
| 30 | - | - | Reserved. Read value is undefined, only zero should be written. |
| 31 | ISE_MAILBOX | [1] | Mailbox interrupt enable (present on LPC54102 devices). |

- [1] Write: writing 0 has no effect, writing 1 enables the interrupt.
Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.

3.4.2 Interrupt Set-Enable Register 1 register

The ISER1 register allows enabling the second group of peripheral interrupts, or for reading the enabled state of those interrupts. Disabling interrupts is done through the ICER0 and ICER1 registers ([Section 3.4.3](#) and [Section 3.4.4](#)).

Table 5. Interrupt Set-Enable Register 1 register

| Bit | Name | Value | Function |
|------|-----------|---------------------|---|
| 0 | ISE_GINT1 | [1] | GPIO group 1 interrupt enable. |
| 1 | ISE_PINT4 | [1] | Pin interrupt / pattern match engine slice 4 interrupt. |
| 2 | ISE_PINT5 | [1] | Pin interrupt / pattern match engine slice 5 interrupt. |
| 3 | ISE_PINT6 | [1] | Pin interrupt / pattern match engine slice 6 interrupt. |
| 4 | ISE_PINT7 | [1] | Pin interrupt / pattern match engine slice 7 interrupt. |
| 7:5 | - | - | Reserved. Read value is undefined, only zero should be written. |
| 8 | ISE_RIT | [1] | Repetitive Interrupt Timer interrupt enable. |
| 31:9 | - | - | Reserved. Read value is undefined, only zero should be written. |

[1] Write: writing 0 has no effect, writing 1 enables the interrupt.
Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled.

3.4.3 Interrupt Clear-Enable Register 0

The ICER0 register allows disabling the first 32 peripheral interrupts, or for reading the enabled state of those interrupts. The remaining interrupts are disabled via the ICER1 register ([Section 3.4.4](#)). Enabling interrupts is done through the ISER0 and ISER1 registers ([Section 3.4.1](#) and [Section 3.4.2](#)).

Table 6. Interrupt Clear-Enable Register 0

| Bit | Name | Function |
|------|---------|--|
| 31:0 | ICE_... | Peripheral interrupt disables. Bit numbers match ISER0 registers (Table 4). Unused bits are reserved. Write: writing 0 has no effect, writing 1 disables the interrupt. Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled. |

3.4.4 Interrupt Clear-Enable Register 1 register

The ICER1 register allows disabling the second group of peripheral interrupts, or for reading the enabled state of those interrupts. Enabling interrupts is done through the ISER0 and ISER1 registers ([Section 3.4.1](#) and [Section 3.4.2](#)).

Table 7. Interrupt Clear-Enable Register 1 register

| Bit | Name | Function |
|------|---------|--|
| 31:0 | ICE_... | Peripheral interrupt disables. Bit numbers match ISER1 registers (Table 5). Unused bits are reserved. Write: writing 0 has no effect, writing 1 disables the interrupt. Read: 0 indicates that the interrupt is disabled, 1 indicates that the interrupt is enabled. |

3.4.5 Interrupt Set-Pending Register 0 register

The ISPR0 register allows setting the pending state of the first 32 peripheral interrupts, or for reading the pending state of those interrupts. The remaining interrupts can have their pending state set via the ISPR1 register ([Section 3.4.6](#)). Clearing the pending state of interrupts is done through the ICPR0 and ICPR1 registers ([Section 3.4.7](#) and

[Section 3.4.8](#)).

Table 8. Interrupt Set-Pending Register 0 register

| Bit | Name | Function |
|------|---------|--|
| 31:0 | ISP_... | Peripheral interrupt pending set. Bit numbers match ISER0 registers (Table 4). Unused bits are reserved. Write: writing 0 has no effect, writing 1 changes the interrupt state to pending. Read: 0 indicates that the interrupt is not pending, 1 indicates that the interrupt is pending. |

3.4.6 Interrupt Set-Pending Register 1 register

The ISPR1 register allows setting the pending state of the second group of peripheral interrupts, or for reading the pending state of those interrupts. Clearing the pending state of interrupts is done through the ICPR0 and ICPR1 registers ([Section 3.4.7](#) and [Section 3.4.8](#)).

Table 9. Interrupt Set-Pending Register 1 register

| Bit | Name | Function |
|------|---------|--|
| 31:0 | ISP_... | Peripheral interrupt pending set. Bit numbers match ISER1 registers (Table 5). Unused bits are reserved. Write: writing 0 has no effect, writing 1 changes the interrupt state to pending. Read: 0 indicates that the interrupt is not pending, 1 indicates that the interrupt is pending. |

3.4.7 Interrupt Clear-Pending Register 0 register

The ICPR0 register allows clearing the pending state of the first 32 peripheral interrupts, or for reading the pending state of those interrupts. The remaining interrupts can have their pending state cleared via the ICPR1 register ([Section 3.4.8](#)). Setting the pending state of interrupts is done through the ISPR0 and ISPR1 registers ([Section 3.4.5](#) and [Section 3.4.6](#)).

Table 10. Interrupt Clear-Pending Register 0 register

| Bit | Name | Function |
|------|---------|--|
| 31:0 | ICP_... | Peripheral interrupt pending clear. Bit numbers match ISER0 registers (Table 4). Unused bits are reserved. Write: writing 0 has no effect, writing 1 changes the interrupt state to not pending. Read: 0 indicates that the interrupt is not pending, 1 indicates that the interrupt is pending. |

3.4.8 Interrupt Clear-Pending Register 1 register

The ICPR1 register allows clearing the pending state of the second group of peripheral interrupts, or for reading the pending state of those interrupts. Setting the pending state of interrupts is done through the ISPR0 and ISPR1 registers ([Section 3.4.5](#) and [Section 3.4.6](#)).

Table 11. Interrupt Clear-Pending Register 1 register

| Bit | Name | Function |
|------|---------|--|
| 31:0 | ICP_... | Peripheral interrupt pending clear. Bit numbers match ISER1 registers (Table 5). Unused bits are reserved. Write: writing 0 has no effect, writing 1 changes the interrupt state to not pending. Read: 0 indicates that the interrupt is not pending, 1 indicates that the interrupt is pending. |

3.4.9 Interrupt Active Bit Register 0

The IABR0 register is a read-only register that allows reading the active state of the first 32 peripheral interrupts. Bits in IABR are set while the corresponding interrupt service routines are in progress. Additional interrupts can have their active state read via the IABR1 register ([Section 3.4.10](#)). IABR registers are not available for the Cortex-M0+.

Table 12. Interrupt Active Bit Register 0

| Bit | Name | Function |
|------|---------|--|
| 31:0 | IAB_... | Peripheral interrupt active. Bit numbers match ISER0 registers (Table 4). Unused bits are reserved. Read: 0 indicates that the interrupt is not active, 1 indicates that the interrupt is active. |

3.4.10 Interrupt Active Bit Register 1

The IABR1 register is a read-only register that allows reading the active state of the second group of peripheral interrupts. Bits in IABR are set while the corresponding interrupt service routines are in progress. IABR registers are not available for the Cortex-M0+.

Table 13. Interrupt Active Bit Register 1

| Bit | Name | Function |
|------|---------|--|
| 31:0 | IAB_... | Peripheral interrupt active. Bit numbers match ISER1 registers (Table 5). Unused bits are reserved. Read: 0 indicates that the interrupt is not active, 1 indicates that the interrupt is active. |

3.4.11 Interrupt Priority Register 0

The IPR0 register controls the priority of the first 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 14. Interrupt Priority Register 0

| Bit | Name | Function |
|-------|--------|---|
| 4:0 | - | Unused |
| 7:5 | IP_WDT | Watchdog Timer interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 12:8 | - | Unused |
| 15:13 | IP_BOD | BOD interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 20:16 | - | Unused |
| 23:21 | - | Reserved. |
| 28:24 | - | Unused |
| 31:29 | IP_DMA | DMA interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |

3.4.12 Interrupt Priority Register 1

The IPR1 register controls the priority of the second group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 15. Interrupt Priority Register 1

| Bit | Name | Function |
|------|----------|---|
| 4:0 | - | Unused |
| 7:5 | IP_GINT0 | GPIO Group 0 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 12:8 | - | Unused |

Table 15. Interrupt Priority Register 1 ...continued

| Bit | Name | Function |
|-------|----------|---|
| 15:13 | IP_PINT0 | Pin interrupt / pattern match engine slice 0 priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 20:16 | - | Unused |
| 23:21 | IP_PINT1 | Pin interrupt / pattern match engine slice 1 priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 28:24 | - | Unused |
| 31:29 | IP_PINT2 | Pin interrupt / pattern match engine slice 2 priority. 0 = highest priority. 31 (0x1F) = lowest priority. |

3.4.13 Interrupt Priority Register 2

The IPR2 register controls the priority of the third group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 16. Interrupt Priority Register 2

| Bit | Name | Function |
|-------|-----------|---|
| 4:0 | - | Unused |
| 7:5 | IP_PINT3 | Pin interrupt / pattern match engine slice 3 priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 12:8 | - | Unused |
| 15:13 | IP_UTICK | Micro-Tick Timer interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 20:16 | - | Unused |
| 23:21 | IP_MRT | Multi-Rate Timer interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 28:24 | - | Unused |
| 31:29 | IP_CT32B0 | Standard counter/timer CT32B0 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |

3.4.14 Interrupt Priority Register 3

The IPR3 register controls the priority of the fourth group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 17. Interrupt Priority Register 3

| Bit | Name | Function |
|-------|-----------|--|
| 4:0 | - | Unused |
| 7:5 | IP_CT32B1 | Standard counter/timer CT32B1 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 12:8 | - | Unused |
| 15:13 | IP_CT32B2 | Standard counter/timer CT32B2 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 20:16 | - | Unused |
| 23:21 | IP_CT32B3 | Standard counter/timer CT32B3 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 28:24 | - | Unused |
| 31:29 | IP_CT32B4 | Standard counter/timer CT32B4 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |

3.4.15 Interrupt Priority Register 4

The IPR4 register controls the priority of the fifth group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 18. Interrupt Priority Register 4

| Bit | Name | Function |
|-------|-----------|--|
| 4:0 | - | Unused |
| 7:5 | IP_SCT0 | SCT0 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 12:8 | - | Unused |
| 15:13 | IP_USART0 | USART 0 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 20:16 | - | Unused |
| 23:21 | IP_USART1 | USART 1 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 28:24 | - | Unused |
| 31:29 | IP_USART2 | USART 2 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |

3.4.16 Interrupt Priority Register 5

The IPR5 register controls the priority of the sixth group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 19. Interrupt Priority Register 5

| Bit | Name | Function |
|-------|-----------|--|
| 4:0 | - | Unused |
| 7:5 | IP_USART3 | USART 3 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 12:8 | - | Unused |
| 15:13 | IP_I2C0 | I2C 0 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 20:16 | - | Unused |
| 23:21 | IP_I2C1 | I2C 1 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 28:24 | - | Unused |
| 31:29 | IP_I2C2 | I2C 2 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |

3.4.17 Interrupt Priority Register 6

The IPR6 register controls the priority of the seventh group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 20. Interrupt Priority Register 6

| Bit | Name | Function |
|-------|-------------|---|
| 4:0 | - | Unused |
| 7:5 | IP_SPI0 | SPI 0 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 12:8 | - | Unused |
| 15:13 | IP_SPI1 | SPI 1 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 20:16 | - | Unused |
| 23:21 | IP_ADC0SEQA | ADC 0 sequence A interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 28:24 | - | Unused |
| 31:29 | IP_ADC0SEQB | ADC 0 sequence B interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |

3.4.18 Interrupt Priority Register 7

The IPR7 register controls the priority of the eighth group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 21. Interrupt Priority Register 7

| Bit | Name | Function |
|-------|-------------|--|
| 4:0 | - | Unused |
| 7:5 | IP_ADC0THOV | ADC 0 threshold and error interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 12:8 | - | Unused |
| 15:13 | IP_RTC | Real Time clock (RTC) interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 20:16 | - | Unused |
| 23:21 | - | Reserved |
| 28:24 | - | Unused |
| 31:29 | IP_MAILBOX | Mailbox interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority (present on LPC54102 devices). |

3.4.19 Interrupt Priority Register 8

The IPR8 register controls the priority of the ninth and last group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 22. Interrupt Priority Register 8

| Bit | Name | Function |
|-------|----------|---|
| 4:0 | - | Unused |
| 7:5 | IP_GINT1 | GPIO Group 1 interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 12:8 | - | Unused |
| 15:13 | IP_PINT4 | Pin interrupt / pattern match engine slice 4 priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 20:16 | - | Unused |
| 23:21 | IP_PINT5 | Pin interrupt / pattern match engine slice 5 priority 0 = highest priority. 31 (0x1F) = lowest priority. |
| 28:24 | - | Unused |
| 31:29 | IP_PINT6 | Pin interrupt / pattern match engine slice 6 priority. 0 = highest priority. 31 (0x1F) = lowest priority. |

3.4.20 Interrupt Priority Register 9

The IPR9 register controls the priority of the tenth group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 23. Interrupt Priority Register 9

| Bit | Name | Function |
|------|----------|---|
| 4:0 | - | Unused |
| 7:5 | IP_PINT7 | Pin interrupt / pattern match engine slice 7 priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 31:8 | - | Reserved |

3.4.21 Interrupt Priority Register 10

The IPR10 register controls the priority of the eleventh group of 4 peripheral interrupts. Each interrupt can have one of 32 priorities, where 0 is the highest priority.

Table 24. Interrupt Priority Register 10

| Bit | Name | Function |
|------|--------|---|
| 4:0 | - | Unused |
| 7:5 | IP_RIT | Repetitive interrupt Timer interrupt priority. 0 = highest priority. 31 (0x1F) = lowest priority. |
| 31:8 | - | Reserved |

3.4.22 Software Trigger Interrupt Register

The STIR register provides an alternate way for software to generate an interrupt, in addition to using the ISPR registers. This mechanism can only be used to generate peripheral interrupts, not system exceptions. the STIR register is not available for the Cortex-M0+.

By default, only privileged software can write to the STIR register. Unprivileged software can be given this ability if privileged software sets the USERSETMPEND bit in the CCR register.

The interrupt number to be programmed in this register is listed in [Table 2](#).

Table 25. Software Trigger Interrupt Register (STIR)

| Bit | Symbol | Description |
|------|--------|--|
| 8:0 | INTID | Writing a value to this field generates an interrupt for the specified the interrupt number. |
| 31:9 | - | Reserved. Read value is undefined, only zero should be written. |

4.1 Features

- System and bus configuration.
- Clock select and control.
- PLL configuration
- Reset control.
- Wake-up control.
- BOD configuration.
- High-accuracy frequency measurement function for on-chip and off-chip clocks.
- Uses a selection of on-chip clocks as reference clock.
- Device ID register.

4.2 Basic configuration

Configure the SYSCON block as follows:

- The SYSCON uses the CLKIN, and CLKOUT pins which can be configured through IOCON. See [Section 4.3](#). RESET is a dedicated pin.
- No clock configuration is needed. The clock to the SYSCON block is always enabled. By default, the SYSCON block is clocked by the IRC.
- Target and reference clocks for the frequency measurement function are selected in the input mux block. See [Table 131](#).

4.2.1 Set up the PLL

The PLL creates a stable output clock at a higher frequency than the input clock. If a main clock is needed with a frequency higher than the 12 MHz IRC clock, use the PLL to boost the input frequency. The PLL can be set up by calling an API supplied by NXP Semiconductors. Also see [Section 4.6.4 “PLL functional description”](#) and [Section 4.5.37 “PLL registers”](#).

4.2.2 Configure the main clock and system clock

The clock source for the registers and memories is derived from main clock. The main clock can be selected from the sources listed in step 1 below.

The divided main clock is called the system clock and clocks the core, the memories, and the peripherals (register interfaces and peripheral clocks).

1. Select the main clock. The following options are available:
 - IRC: 12 MHz internal oscillator (default)
 - CLKIN
 - Watchdog oscillator
 - The output of the system PLL

- The RTC 32 kHz oscillator

[Section 4.5.16 “Main clock source select register A”](#) and [Section 4.5.17 “Main clock source select register B”](#).

2. Select the divider value for the system clock. A divider value of 0 disables the system clock.

[Section 4.5.29 “System clock divider register”](#)

3. Select the memories and peripherals that are operating in the application and therefore must have an active clock. The core is always clocked.

[Section 4.5.22 “AHB Clock Control register 0”](#) and [Section 4.5.23 “AHB Clock Control register 1”](#).

4.2.3 Measure the frequency of a clock signal

The frequency of any on-chip or off-chip clock signal can be measured accurately with a selectable reference clock. For example, the frequency measurement function can be used to accurately determine the frequency of the watchdog oscillator which varies over a wide range depending on process and temperature.

The clock frequency to be measured and the reference clock are selected in the input mux block. See [Section 9.6.4 “Frequency measure function reference clock select register”](#) and [Section 9.6.5 “Frequency measure function target clock select register”](#).

Details on the accuracy and measurement process are described in [Section 4.6.5 “Frequency measure function”](#).

To start a frequency measurement cycle and read the result, see [Table 61](#).

4.3 Pin description

Table 26. SYSCON pin description

| Function | Direction | Pin | Description | Reference |
|----------|-----------|---------|-----------------------|---------------------------|
| CLKOUT | O | PIO0_21 | CLKOUT clock output. | Chapter 8 |
| CLKIN | I | PIO0_22 | External clock input. | Chapter 8 |

4.4 General description

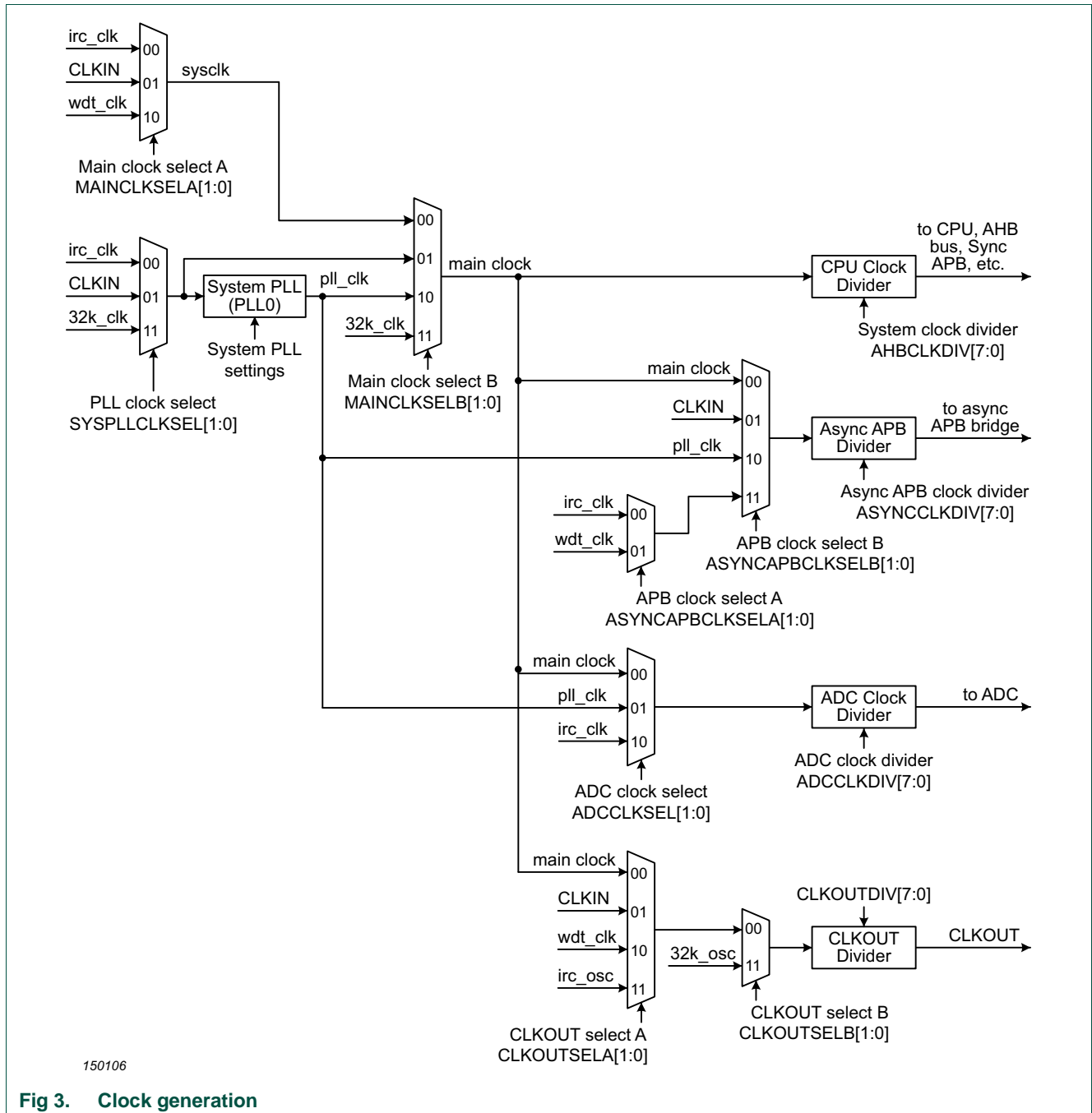
4.4.1 Clock generation

The system control block facilitates the clock generation. Many clocking variations are possible. [Figure 3](#) gives an overview of potential clock options. The maximum clock frequency is 100 MHz.

Remark: In order for any of the clock multiplexers shown in [Figure 3](#) to operate, the currently selected clock must be running, and the clock to be switched to must also be running. This is so that the multiplexer can gracefully switch between the two clocks without glitches.

The low-power watchdog oscillator provides a fixed clock of approximately 500 kHz. The accuracy of this clock is limited to +/- 40% over temperature, voltage, and silicon processing variations. To determine the actual watchdog oscillator output, use the frequency measure block. See [Section 4.2.3](#).

The part contains one system PLL that can be configured to use a number of clock inputs and produce an output clock in the range of 1.2 MHz up to the maximum chip frequency, and can be used to run most on-chip functions. The output of the PLL can be monitored through the CLKOUT pin.



4.5 Register description

All system control block registers reside on word address boundaries. Details of the registers appear in the description of each function. System configuration functions are divided into 3 groups: Main system configuration at base address 0x4000 0000 (see [Table 27](#)), Asynchronous system configuration at base address 0x4008 0000 (see [Table 28](#)), and Other system registers at base address 0x4002 C000 (see [Table 29](#)).

All address offsets not shown in the tables are reserved and should not be written to.

Remark: The reset value column shows the reset value seen when the boot loader executes and the flash contains valid user code. During code development, a different value may be seen if a debugger is used to halt execution prior to boot completion.

Table 27. Register overview: Main system configuration (base address 0x4000 0000)

| Name | Access | Offset | Description | Reset value ^[1] | Reference |
|----------------|--------|--------|--|----------------------------|--------------------------|
| AHBMATPRIO | R/W | 0x004 | AHB multilayer matrix priority control | 0x0 | Table 30 |
| SYSTCKCAL | R/W | 0x014 | System tick counter calibration | 0x0 | Table 31 |
| NMISRC | R/W | 0x01C | NMI Source Select | 0x0 | Table 32 |
| ASYNCAPBCTRL | R/W | 0x020 | Asynchronous APB Control | 0x1 | Table 33 |
| SYSRSTSTAT | R/W | 0x040 | System reset status register | Note [2] | Table 34 |
| PRESETCTRL0 | R/W | 0x044 | Peripheral reset control 0 | 0x0 | Table 35 |
| PRESETCTRL1 | R/W | 0x048 | Peripheral reset control 1 | 0x0 | Table 36 |
| PRESETCTRLSET0 | WO | 0x04C | Set bits in PRESETCTRL0 | - | Table 37 |
| PRESETCTRLSET1 | WO | 0x050 | Set bits in PRESETCTRL1 | - | Table 38 |
| PRESETCTRLCLR0 | WO | 0x054 | Clear bits in PRESETCTRL0 | - | Table 39 |
| PRESETCTRLCLR1 | WO | 0x058 | Clear bits in PRESETCTRL1 | - | Table 40 |
| PIOPORCAP0 | RO | 0x05C | POR captured value of port 0 | Note [3] | Table 41 |
| PIOPORCAP1 | RO | 0x060 | POR captured value of port 1 | Note [3] | Table 42 |
| PIORESCAP0 | RO | 0x068 | Reset captured value of port 0 | Note [4] | Table 43 |
| PIORESCAP1 | RO | 0x06C | Reset captured value of port 1 | Note [4] | Table 44 |
| MAINCLKSELA | R/W | 0x080 | Main clock source select A | 0x0 | Table 45 |
| MAINCLKSELB | R/W | 0x084 | Main clock source select B | 0x0 | Table 46 |
| ADCCLKSEL | R/W | 0x08C | ADC clock source select | 0x0 | Table 47 |
| CLKOUTSELA | R/W | 0x094 | CLKOUT clock source select A | 0x0 | Table 48 |
| CLKOUTSELB | R/W | 0x098 | CLKOUT clock source select B | 0x0 | Table 49 |
| SYSPLLCLKSEL | R/W | 0x0A0 | PLL clock source select | 0x0 | Table 50 |
| AHBCLKCTRL0 | R/W | 0x0C0 | AHB Clock control 0 | 0x18B | Table 51 |
| AHBCLKCTRL1 | R/W | 0x0C4 | AHB Clock control 1 | 0x0 | Table 52 |
| AHBCLKCTRLSET0 | WO | 0x0C8 | Set bits in AHBCLKCTRL0 | - | Table 53 |
| AHBCLKCTRLSET1 | WO | 0x0CC | Set bits in AHBCLKCTRL1 | - | Table 54 |
| AHBCLKCTRLCLR0 | WO | 0x0D0 | Clear bits in AHBCLKCTRL0 | - | Table 55 |
| AHBCLKCTRLCLR1 | WO | 0x0D4 | Clear bits in AHBCLKCTRL1 | - | Table 56 |
| SYSTICKCLKDIV | R/W | 0x0E0 | SYSTICK clock divider | 0x0 | Table 57 |
| AHBCLKDIV | R/W | 0x100 | System clock divider | 0x1 | Table 58 |
| ADCCLKDIV | R/W | 0x108 | ADC clock divider | 0x0 | Table 59 |

Table 27. Register overview: Main system configuration (base address 0x4000 0000) ...continued

| Name | Access | Offset | Description | Reset value ^[1] | Reference |
|--------------|--------|--------|---------------------------------------|----------------------------|--------------------------|
| CLKOUTDIV | R/W | 0x10C | CLKOUT clock divider | 0x0 | Table 60 |
| FREQMECTRL | R/W | 0x120 | Frequency measure register | 0x0 | Table 61 |
| FLASHCFG | R/W | 0x124 | Flash wait states configuration | 0x001A | Table 62 |
| FIFOCTRL | R/W | 0x148 | Serial interface FIFO enables | 0 | Table 63 |
| IRCCTRL | R/W | 0x184 | IRC oscillator control | Note [5] | Table 64 |
| RTCOSCCTRL | R/W | 0x190 | RTC oscillator 32 kHz output control | 0x1 | Table 65 |
| SYSPLLCTRL | R/W | 0x1B0 | PLL control | 0x8000 | Table 66 |
| SYSPLLSTAT | RO | 0x1B4 | PLL status | 0x0 | Table 67 |
| SYSPLLNDDEC | R/W | 0x1B8 | PLL N decoder | 0x0 | Table 68 |
| SYSPLLPDEC | R/W | 0x1BC | PLL P decoder | 0x0 | Table 69 |
| SYSPLSSCTRL0 | R/W | 0x1C0 | PLL spread spectrum control 0 | 0x0 | Table 70 |
| SYSPLSSCTRL1 | R/W | 0x1C4 | PLL spread spectrum control 1 | 0x1000 0000 | Table 71 |
| PDRUNCFG | R/W | 0x210 | Power configuration register | 0xD80500 | Table 72 |
| PDRUNCFGSET | WO | 0x214 | Set bits in PDRUNCFG | - | Table 73 |
| PDRUNCFGCLR | WO | 0x218 | Clear bits in PDRUNCFG | - | Table 74 |
| STARTER0 | R/W | 0x240 | Start logic 0 wake-up enable register | 0x0 | Table 75 |
| STARTER1 | R/W | 0x244 | Start logic 1 wake-up enable register | 0x0 | Table 76 |
| STARTERSET0 | WO | 0x248 | Set bits in STARTER0 | - | Table 77 |
| STARTERSET1 | WO | 0x24C | Set bits in STARTER1 | - | Table 78 |
| STARTERCLR0 | WO | 0x250 | Clear bits in STARTER0 | - | Table 79 |
| STARTERCLR1 | WO | 0x254 | Clear bits in STARTER1 | - | Table 80 |
| CPUCTRL | R/W | 0x300 | CPU Control for multiple processors | 0x4D | Table 81 |
| CPBOOT | R/W | 0x304 | Coprocessor Boot Address | 0 | Table 82 |
| CPSTACK | R/W | 0x308 | Coprocessor Stack Address | 0 | Table 83 |
| CPSTAT | RO | 0x30C | Coprocessor Status | 0 | Table 84 |
| JTAGIDCODE | RO | 0x3F4 | JTAG ID code register | see table | Table 85 |
| DEVICE_ID0 | RO | 0x3F8 | Part ID register | Note [5] | Table 86 |
| DEVICE_ID1 | RO | 0x3FC | Boot ROM and die revision register | Note [5] | Table 88 |

[1] Reset Value reflects the data stored in defined bits only. Reserved bits assumed to be 0.

[2] Depends on the source of the most recent reset.

[3] Determined by the voltage levels on device pins upon power-on reset.

[4] Determined by the voltage levels on device pins when a reset other than power-on reset occurs.

[5] Part dependent.

Table 28. Register overview: Asynchronous system configuration (base address 0x4008 0000)

| Name | Access | Offset | Description | Reset value ^[1] | Reference |
|--------------------|--------|--------|--------------------------------|----------------------------|--------------------------|
| ASYNCPRESETCTRL | R/W | 0x000 | Async peripheral reset control | 0x0 | Table 89 |
| ASYNCPRESETCTRLSET | WO | 0x004 | Set bits in ASYNCPRESETCTRL | - | Table 90 |
| ASYNCPRESETCTRLCLR | WO | 0x008 | Clear bits in ASYNCPRESETCTRL | - | Table 91 |
| ASYNCPBCLKCTRL | R/W | 0x010 | Async peripheral clock control | 0x0 | Table 92 |
| ASYNCPBCLKCTRLSET | WO | 0x014 | Set bits in ASYNCPBCLKCTRL | - | Table 93 |

Table 28. Register overview: Asynchronous system configuration (base address 0x4008 0000) ...continued

| Name | Access | Offset | Description | Reset value [1] | Reference |
|--------------------|--------|--------|---|-----------------|--------------------------|
| ASYNCAPBCLKCTRLCLR | WO | 0x018 | Clear bits in ASYNCAPBCLKCTRL | - | Table 94 |
| ASYNCAPBCLKSELA | R/W | 0x020 | Async APB clock source select A | 0x0 | Table 95 |
| ASYNCAPBCLKSELB | R/W | 0x024 | Async APB clock source select B | 0x0 | Table 96 |
| ASYNCLCKDIV | R/W | 0x028 | Async APB clock divider | 0x1 | Table 97 |
| FRGCTRL | R/W | 0x030 | USART fractional rate generator control | 0xFF | Table 98 |

[1] Reset Value reflects the data stored in defined bits only. Reserved bits assumed to be 0.

Table 29. Register overview: Other system configuration (base address 0x4002 C000)

| Name | Access | Offset | Description | Reset value [1] | Reference |
|---------|--------|--------|--------------------------|-----------------|--------------------------|
| BODCTRL | R/W | 0x44 | Brown-Out Detect control | 0x0 | Table 99 |

[1] Reset Value reflects the data stored in defined bits only. Reserved bits assumed to be 0.

4.5.1 AHB matrix priority register

The Multilayer AHB Matrix arbitrates between several masters, only if they attempt to access the same matrix slave port at the same time. Care should be taken if the value in this register is changed, improper settings can seriously degrade performance.

Priority values are 3 = highest, 0 = lowest. When the priority is the same, the master with the lower number is given priority. An example setting could put the Cortex-M4 D-code bus as the highest priority, followed by the I-Code bus. All other masters could share a lower priority.

Table 30. AHB matrix priority register 0 (AHBMATPRIO, address 0x4000 0004) bit description

| Bit | Symbol | Description | Reset value |
|-------|-----------|--|-------------|
| 1:0 | PRI_ICODE | I-Code bus priority (master 0). Should be lower than PRI_DCODE for proper operation. | 0 |
| 3:2 | PRI_DCODE | D-Code bus priority (master 1). | 0 |
| 5:4 | PRI_SYS | System bus priority (master 2). | 0 |
| 7:6 | - | Reserved. Read value is undefined, only zero should be written. | - |
| 9:8 | PRI_DMA | DMA controller priority (master 5). | 0 |
| 13:10 | - | Reserved. Read value is undefined, only zero should be written. | - |
| 15:14 | PRI_FIFO | System FIFO bus priority (master 9). | 0 |
| 17:16 | PRI_M0 | Cortex-M0+ bus priority (master 10). Present on LPC54102 devices. | 0 |
| 31:18 | - | Reserved. Read value is undefined, only zero should be written. | - |

4.5.2 System tick counter calibration register

This register allows software to set up a default value for the SYST_CALIB register in the System Tick Timer of each CPU. See [Chapter 20](#).

Table 31. System tick timer calibration register (SYSTCKCAL, address 0x4000 0014) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--------------------------------------|-------------|
| 23:0 | CAL | System tick timer calibration value. | 0 |
| 24 | SKEW | Initial value for the SysTick timer. | |
| 25 | NOREF | Initial value for the SysTick timer. | |
| 31:26 | - | Reserved. | - |

4.5.3 NMI source selection register

The NMI source selection register selects a peripheral interrupts as source for the NMI interrupt of both CPUs. For a list of all peripheral interrupts and their IRQ numbers see [Table 2](#). For a description of the NMI functionality, see [Ref. 1 "Cortex-M4 TRM"](#).

Remark: In order to change the interrupt source for the NMI, the NMI source must first be disabled by writing 0 to the NMIEN bit. Then change the source by updating the IRQN bits and re-enable the NMI source by setting NMIEN.

Table 32. NMI source selection register (NMISRC, address 0x4000 001C) bit description

| Bit | Symbol | Description | Reset value |
|-------|---------|---|-------------|
| 5:0 | IRQM4 | The IRQ number of the interrupt that acts as the Non-Maskable Interrupt (NMI) for the Cortex-M4, if enabled by NMIENM4. | 0 |
| 7:6 | - | Reserved. Read value is undefined, only zero should be written. | - |
| 13:8 | IRQM0 | The IRQ number of the interrupt that acts as the Non-Maskable Interrupt (NMI) for the Cortex-M0+, if enabled by NMIENM0. Present on LPC54102 devices. | 0 |
| 29:14 | - | Reserved. Read value is undefined, only zero should be written. | - |
| 30 | NMIENM0 | Write a 1 to this bit to enable the Non-Maskable Interrupt (NMI) source selected by IRQM0. Present on LPC54102 devices. | 0 |
| 31 | NMIENM4 | Write a 1 to this bit to enable the Non-Maskable Interrupt (NMI) source selected by IRQM4. | 0 |

Remark: If the NMISRC register is used to select an interrupt as the source of Non-Maskable interrupts, and the selected interrupt is enabled, one interrupt request can result in both a Non-Maskable and a normal interrupt. This can be avoided by disabling the normal interrupt in the NVIC.

4.5.4 Asynchronous APB Control register

ASYNCAPBCTRL contains a global enable bit for the asynchronous APB bridge and subsystem, allowing connection to the associated peripherals.

Table 33. Asynchronous APB Control register (ASYNCAPBCTRL, address 0x4000 0020) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---|-------------|
| 0 | ENABLE | | Enables the asynchronous APB bridge and subsystem. | 1 |
| | | 0 | Disabled. Asynchronous APB bridge is disabled. | |
| | | 1 | Enabled. Asynchronous APB bridge is enabled. | |
| 31:1 | - | - | Reserved. Read value is undefined, only zero should be written. | - |

4.5.5 System reset status register

The SYSRSTSTAT register shows the source of the latest reset event. The bits are cleared by writing a one to any of the bits. The POR event clears all other bits in this register. If another reset signal - for example the external RESET pin - remains asserted after the POR signal is negated, then its bit is set to detected. Write a one to clear the reset.

Table 34. System reset status register (SYSRSTSTAT, address 0x4000 0040) bit description

| Bit | Symbol | Value | Description |
|------|--------|-------|---|
| 0 | POR | | POR reset status |
| | | 0 | No POR detected |
| | | 1 | POR detected. Writing a one clears this reset. |
| 1 | EXTRST | | Status of the external <u>RESET</u> pin. External reset status. |
| | | 0 | No reset event detected. |
| | | 1 | Reset detected. Writing a one clears this reset. |
| 2 | WDT | | Status of the Watchdog reset |
| | | 0 | No WDT reset detected |
| | | 1 | WDT reset detected. Writing a one clears this reset. |
| 3 | BOD | | Status of the Brown-out detect reset |
| | | 0 | No BOD reset detected |
| | | 1 | BOD reset detected. Writing a one clears this reset. |
| 4 | SYSRST | | Status of the software system reset |
| | | 0 | No System reset detected |
| | | 1 | System reset detected. Writing a one clears this reset. |
| 31:5 | - | - | Reserved |

4.5.6 Peripheral reset control register 0

The PRESETCTRL0 register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

Table 35. Peripheral reset control register 0 (PRESETCTRL0, address 0x4000 0044) bit description

| Bit | Symbol | Description | Reset value |
|------|-----------|--|-------------|
| 6:0 | - | Reserved. Read value is undefined, only zero should be written. | 0 |
| 7 | FLASH_RST | Flash controller reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 8 | FMC_RST | Flash accelerator reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 10:9 | - | Reserved. Read value is undefined, only zero should be written. | 0 |
| 11 | MUX_RST | Input mux reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 12 | - | Reserved. Read value is undefined, only zero should be written. | 0 |
| 13 | IOCON_RST | IOCON reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |

Table 35. Peripheral reset control register 0 (PRESETCTRL0, address 0x4000 0044) bit description

| Bit | Symbol | Description | Reset value |
|-------|-----------|---|-------------|
| 14 | GPIO0_RST | GPIO0 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 15 | GPIO1_RST | GPIO1 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 17:16 | - | Reserved. Read value is undefined, only zero should be written. | 0 |
| 18 | PINT_RST | Pin interrupt (PINT) reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 19 | GINT_RST | Grouped interrupt (GINT) reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 20 | DMA_RST | DMA reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 21 | CRC_RST | CRC generator reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 22 | WWDT_RST | Watchdog timer reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 26:23 | - | Reserved. Read value is undefined, only zero should be written. | 0 |
| 27 | ADC0_RST | ADC0 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 31:28 | - | Reserved. Read value is undefined, only zero should be written. | - |

4.5.7 Peripheral reset control register 1

The PRESETCTRL1 register allows software to reset specific peripherals. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

Table 36. Peripheral reset control register 1 (PRESETCTRL1, address 0x4000 0048) bit description

| Bit | Symbol | Description | Reset value |
|-----|-----------|---|-------------|
| 0 | MRT_RST | Multi-rate timer (MRT) reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 1 | RIT_RST | Repetitive interrupt timer (RIT) reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 2 | SCT0_RST | State configurable timer 0 (SCT0) reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 8:3 | - | Reserved. Read value is undefined, only zero should be written. | 0 |
| 9 | FIFO_RST | System FIFO reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 10 | UTICK_RST | Micro-tick Timer reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |

Table 36. Peripheral reset control register 1 (PRESETCTRL1, address 0x4000 0048) bit description

| Bit | Symbol | Description | Reset value |
|-------|------------|--|-------------|
| 21:11 | - | Reserved. Read value is undefined, only zero should be written. | 0 |
| 22 | CT32B2_RST | CT32B2 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 25:23 | - | Reserved. Read value is undefined, only zero should be written. | 0 |
| 26 | CT32B3_RST | CT32B3 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 27 | CT32B4_RST | CT32B4 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 31:28 | - | Reserved. Read value is undefined, only zero should be written. | - |

4.5.8 Peripheral reset control set register 0

Writing a 1 to a bit position in PRESETCTRLSET0 sets the corresponding position in PRESETCTRL0. This is a write-only register. For bit assignments, see [Table 35](#).

Table 37. Peripheral reset control set register 0 (PRESETCTRLSET0, address 0x4000 004C) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|--|-------------|
| 31:0 | RST_SET0 | Writing ones to this register sets the corresponding bit or bits in the PRESETCTRL0 register, if they are implemented. Bits that do not correspond to defined bits in PRESETCTRL0 are reserved and only zeroes should be written to them. | - |

4.5.9 Peripheral reset control set register 1

Writing a 1 to a bit position in PRESETCTRLSET1 sets the corresponding position in PRESETCTRL1. This is a write-only register. For bit assignments, see [Table 36](#).

Table 38. Peripheral reset control set register 1 (PRESETCTRLSET1, address 0x4000 0050) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|--|-------------|
| 31:0 | RST_SET1 | Writing ones to this register sets the corresponding bit or bits in the PRESETCTRL1 register, if they are implemented. Bits that do not correspond to defined bits in PRESETCTRL1 are reserved and only zeroes should be written to them. | - |

4.5.10 Peripheral reset control clear register 0

Writing a 1 to a bit position in PRESETCTRLCLR0 clears the corresponding position in PRESETCTRL0. This is a write-only register. For bit assignments, see [Table 35](#).

Table 39. Peripheral reset control clear register 0 (PRESETCTRLCLR0, address 0x4000 0054) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|--|-------------|
| 31:0 | RST_CLR0 | Writing ones to this register clears the corresponding bit or bits in the PRESETCTRL0 register, if they are implemented. Bits that do not correspond to defined bits in PRESETCTRL0 are reserved and only zeroes should be written to them. | - |

4.5.11 Peripheral reset control clear register 1

Writing a 1 to a bit position in PRESETCTRLCLR1 clears the corresponding position in PRESETCTRL1. This is a write-only register. For bit assignments, see [Table 36](#).

Table 40. Peripheral reset control clear register 1 (PRESETCTRLCLR1, address 0x4000 0058) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|--|-------------|
| 31:0 | RST_CLR1 | Writing ones to this register clears the corresponding bit or bits in the PRESETCTRL1 register, if they are implemented. Bits that do not correspond to defined bits in PRESETCTRL1 are reserved and only zeroes should be written to them. | - |

4.5.12 POR captured value of port 0

The PIOPORCAP0 register captures the state of GPIO port 0 at power-on-reset. Each bit represents the power-on reset state of one GPIO pin. This register is a read-only register.

Table 41. POR captured PIO status register 0 (PIOPORCAP0, address 0x4000 005C) bit description

| Bit | Symbol | Description | Reset value |
|------|-----------|---|-------------------------------|
| 31:0 | PIOPORCAP | State of PIO0_31 through PIO0_0 at power-on reset | Depends on external circuitry |

4.5.13 POR captured value of port 1

The PIOPORCAP1 register captures the state of GPIO port 1 at power-on-reset. Each bit represents the power-on reset state of one GPIO pin. This register is a read-only register.

Table 42. POR captured PIO status register 1 (PIOPORCAP1, address 0x4000 0060) bit description

| Bit | Symbol | Description | Reset value |
|------|-----------|---|-------------------------------|
| 31:0 | PIOPORCAP | State of PIO1_31 through PIO1_0 at power-on reset | Depends on external circuitry |

4.5.14 Reset captured value of port 0

The PIORESCAP0 register captures the state of GPIO port 0 when a reset other than a power-on reset occurs. Each bit represents the reset state of one GPIO pin. This register is a read-only register.

Table 43. Reset captured PIO status register 0 (PIORESCAP0, address 0x4000 0068) bit description

| Bit | Symbol | Description | Reset value |
|------|-----------|--|-------------------------------|
| 31:0 | PIORESCAP | State of PIO0_31 through PIO0_0 for resets other than POR. | Depends on external circuitry |

4.5.15 Reset captured value of port 1

The PIORESCAP1 register captures the state of GPIO port 1 when a reset other than a power-on reset occurs. Each bit represents the reset state of one GPIO pin. This register is a read-only register.

Table 44. Reset captured PIO status register 1 (PIORESCAP1, address 0x4000 006C) bit description

| Bit | Symbol | Description | Reset value |
|------|-----------|--|-------------------------------|
| 31:0 | PIORESCAP | State of PIO1_31 through PIO1_0 for resets other than POR. | Depends on external circuitry |

4.5.16 Main clock source select register A

This register selects one of the internal oscillators, IRC, system oscillator, or watchdog oscillator. The oscillator selected is then one of the inputs to the main clock source select register B (see [Table 46](#)), which selects the clock source for the main clock. All clocks to the core, memories, and peripherals on the synchronous APB bus are derived from the main clock.

Remark: Note that this selection is internally synchronized: the clock being switched from and the clock being switched to must both be running and have occurred in specific states before the selection actually changes.

Table 45. Main clock source select register A (MAINCLKSELA, address 0x4000 0080) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---|-------------|
| 1:0 | SEL | | Clock source for main clock source selector A | 0 |
| | | 0x0 | IRC Oscillator | |
| | | 0x1 | CLKIN | |
| | | 0x2 | Watchdog oscillator | |
| | | 0x3 | Reserved | |
| 31:2 | - | - | Reserved | - |

4.5.17 Main clock source select register B

This register selects the clock source for the main clock. All clocks to the core, memories, and peripherals are derived from the main clock.

One input to this register is the main clock source select register A (see [Table 45](#)), which selects one of the three internal oscillators, IRC, system oscillator, or watchdog oscillator.

Remark: Note that this selection is internally synchronized: the clock being switched from and the clock being switched to must both be running and have occurred in specific states before the selection actually changes.

Table 46. Main clock source select register B (MAINCLKSELB, address 0x4000 0084) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---|-------------|
| 1:0 | SEL | | Clock source for main clock source selector B. Selects the clock source for the main clock. | 0 |
| | | 0x0 | MAINCLKSELA. Use the clock source selected in MAINCLKSELA register. | |
| | | 0x1 | System PLL input. | |
| | | 0x2 | System PLL output. | |
| | | 0x3 | RTC oscillator output. RTC oscillator 32 kHz output. | |
| 31:2 | - | - | Reserved | - |

4.5.18 ADC clock source select register

This register selects a clock source for the 12-bit ADCs that is to the system clock. To use a clock other than the Main clock, select the asynchronous clock mode in the ADC control register.

Remark: Note that this selection is internally synchronized: the clock being switched from and the clock being switched to must both be running and have occurred in specific states before the selection actually changes.

Table 47. ADC clock source select (ADCCLKSEL, address 0x4000 008C) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|-------------------|-------------|
| 1:0 | SEL | | ADC clock source. | 0 |
| | | 0x0 | Main clock | |
| | | 0x1 | System PLL output | |
| | | 0x2 | IRC Oscillator | |
| | | 0x3 | reserved | |
| 31:2 | - | | Reserved | - |

4.5.19 CLKOUT clock source select register A

This register pre-selects one of the internal oscillators for the clock sources visible on the CLKOUT pin. The final selection for the CLKOUT clock source is done in the CLKOUT clock source B register.

Remark: Note that this selection is internally synchronized: the clock being switched from and the clock being switched to must both be running and have occurred in specific states before the selection actually changes.

Table 48. CLKOUT clock source select register (CLKOUTSELA, address 0x4000 0094) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---------------------|-------------|
| 1:0 | SEL | | CLKOUT clock source | 0 |
| | | 0x0 | Main clock | |
| | | 0x1 | CLKIN | |
| | | 0x2 | Watchdog oscillator | |
| | | 0x3 | IRC oscillator | |
| 31:2 | - | - | Reserved | - |

4.5.20 CLKOUT clock source select register B

This register selects the clock source visible on the CLKOUT pin. The internal oscillators are pre-selected in the CLKOUTSELA register (see [Table 48](#)).

Remark: Note that this selection is internally synchronized: the clock being switched from and the clock being switched to must both be running and have occurred in specific states before the selection actually changes.

Table 49. CLKOUT clock source select register (CLKOUTSELB, address 0x4000 0098) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---|-------------|
| 1:0 | SEL | | CLKOUT clock source | 0 |
| | | 0x0 | CLKOUTSELA. Clock source selected in the CLKOUTSELA register. | |
| | | 0x1 | reserved | |
| | | 0x2 | reserved | |
| | | 0x3 | RTC 32 kHz clock | |
| 31:2 | - | - | Reserved | - |

4.5.21 System PLL clock source select register

This register selects the clock source for the system PLL.

Remark: Note that this selection is internally synchronized: the clock being switched from and the clock being switched to must both be running and have occurred in specific states before the selection actually changes.

Table 50. System PLL clock source select register (SYSPLLCLKSEL, address 0x4000 00A0) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|-------------------------|-------------|
| 1:0 | SEL | | System PLL clock source | 0 |
| | | 0x0 | IRC Oscillator | |
| | | 0x1 | CLKIN | |
| | | 0x2 | Reserved | |
| | | 0x3 | RTC 32 kHz clock | |
| 31:2 | - | - | Reserved | - |

4.5.22 AHB Clock Control register 0

The AHBCLKCTRL0 register enables the clocks to individual system and peripheral blocks. The system clock (bit 0) provides the clock for the AHB, the APB bridge, the CPU, the SYSCON block, and the PMU. This clock cannot be disabled.

Regarding bits 3 and 4, see [Section 2.1.1](#) for details of SRAM configuration.

Table 51. AHB Clock Control register 0 (AHBCLKCTRL0, address 0x4000 00C0) bit description

| Bit | Symbol | Description | Reset value after boot |
|-------|----------|--|------------------------|
| 0 | - | Reserved. This read-only bit cannot be cleared. | 1 |
| 1 | ROM | Enables the clock for the Boot ROM. 0 = Disable; 1 = Enable. | 1 |
| 2 | - | Reserved. Read value is undefined, only zero should be written. | 0 |
| 3 | SRAM1 | Enables the clock for SRAM1. 0 = Disable; 1 = Enable. | 1 |
| 4 | SRAM2 | Enables the clock for SRAM2. 0 = Disable; 1 = Enable. | 0 |
| 6:5 | - | Reserved. Read value is undefined, only zero should be written. | 0 |
| 7 | FLASH | Enables the clock for the flash controller. 0 = Disable; 1 = Enable. This clock is needed for flash programming, not for flash read. | 1 |
| 8 | FMC | Enables the clock for the Flash accelerator. 0 = Disable; 1 = Enable. This clock is needed if the flash is being read. | 1 |
| 10:9 | - | Reserved. Read value is undefined, only zero should be written. | 0 |
| 11 | INPUTMUX | Enables the clock for the input muxes. 0 = Disable; 1 = Enable. | 0 |
| 12 | - | Reserved. Read value is undefined, only zero should be written. | 0 |
| 13 | IOCON | Enables the clock for the IOCON block. 0 = Disable; 1 = Enable. | 0 |
| 14 | GPIO0 | Enables the clock for the GPIO0 port registers. 0 = Disable; 1 = Enable. | 0 |
| 15 | GPIO1 | Enables the clock for the GPIO1 port registers. 0 = Disable; 1 = Enable. | 0 |
| 17:16 | - | Reserved. Read value is undefined, only zero should be written. | 0 |
| 18 | PINT | Enables the clock for the pin interrupt block. 0 = Disable; 1 = Enable. | 0 |
| 19 | GINT | Enables the clock for the grouped pin interrupt block. 0 = Disable; 1 = Enable. | 0 |
| 20 | DMA | Enables the clock for the DMA controller. 0 = Disable; 1 = Enable. | 0 |
| 21 | CRC | Enables the clock for the CRC engine. 0 = Disable; 1 = Enable. | 0 |
| 22 | WWDT | Enables the clock for the Watchdog Timer. 0 = Disable; 1 = Enable. | 0 |
| 23 | RTC | Enables the clock for the RTC. 0 = Disable; 1 = Enable. | 0 |
| 25:24 | - | Reserved. Read value is undefined, only zero should be written. | 0 |
| 26 | MAILBOX | Enables the clock for the Mailbox. 0 = Disable; 1 = Enable. Present on LPC54102 devices | 0 |
| 27 | ADC0 | Enables the clock for the ADC0 register interface. 0 = Disable; 1 = Enable. | 0 |
| 31:28 | - | Reserved. Read value is undefined, only zero should be written. | 0 |

4.5.23 AHB Clock Control register 1

The AHBCLKCTRL1 register enables the clocks to individual peripheral blocks.

Table 52. AHB Clock Control register 1 (AHBCLKCTRL1, address 0x4000 00C4) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 0 | MRT | Enables the clock for the Multi-Rate Timer. 0 = Disable; 1 = Enable. | 0 |
| 1 | RIT | Enables the clock for the repetitive interrupt timer. 0 = Disable; 1 = Enable. | 0 |
| 2 | SCT0 | Enables the clock for SCT0. 0 = Disable; 1 = Enable. | 0 |
| 8:3 | - | Reserved. Read value is undefined, only zero should be written. | - |
| 9 | FIFO | Enables the clock for system FIFOs. 0 = Disable; 1 = Enable. | 0 |
| 10 | UTICK | Enables the clock for the Micro-tick Timer. 0 = Disable; 1 = Enable. | 0 |
| 21:11 | - | Reserved. Read value is undefined, only zero should be written. | 0 |
| 22 | CT32B2 | Enables the clock for CT32B 2. 0 = Disable; 1 = Enable. | 0 |
| 25:23 | - | Reserved. Read value is undefined, only zero should be written. | - |
| 26 | CT32B3 | Enables the clock for CT32B 3. 0 = Disable; 1 = Enable. | 0 |
| 27 | CT32B4 | Enables the clock for CT32B 4. 0 = Disable; 1 = Enable. | 0 |
| 31:28 | - | Reserved. Read value is undefined, only zero should be written. | - |

4.5.24 AHB Clock Control Set register 0

Writing a 1 to a bit position in AHBCLKCTRLSET0 sets the corresponding position in AHBCLKCTRL0. This is a write-only register. For bit assignments, see [Table 51](#).

Table 53. Clock control set register 0 (AHBCLKCTRLSET0, address 0x4000 00C8) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|--|-------------|
| 31:0 | CLK_SET0 | Writing ones to this register sets the corresponding bit or bits in the AHBCLKCTRL0 register, if they are implemented. Bits that do not correspond to defined bits in AHBCLKCTRL0 are reserved and only zeroes should be written to them. | - |

4.5.25 AHB Clock Control Set register 1

Writing a 1 to a bit position in AHBCLKCTRLSET1 sets the corresponding position in AHBCLKCTRL1. This is a write-only register. For bit assignments, see [Table 52](#).

Table 54. Clock control set register 1 (AHBCLKCTRLSET1, address 0x4000 00CC) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|--|-------------|
| 31:0 | CLK_SET1 | Writing ones to this register sets the corresponding bit or bits in the AHBCLKCTRL1 register, if they are implemented. Bits that do not correspond to defined bits in AHBCLKCTRL1 are reserved and only zeroes should be written to them. | - |

4.5.26 AHB Clock Control Clear register 0

Writing a 1 to a bit position in AHBCLKCTRLCLR0 clears the corresponding position in AHBCLKCTRL0. This is a write-only register. For bit assignments, see [Table 51](#).

Table 55. Clock control clear register 0 (AHBCLKCTRLCLR0, address 0x4000 00D0) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|--|-------------|
| 31:0 | CLK_CLR0 | Writing ones to this register clears the corresponding bit or bits in the AHBCLKCTRL0 register, if they are implemented. Bits that do not correspond to defined bits in AHBCLKCTRL0 are reserved and only zeroes should be written to them. | - |

4.5.27 AHB Clock Control Clear register 1

Writing a 1 to a bit position in AHBCLKCTRLCLR1 clears the corresponding position in AHBCLKCTRL1. This is a write-only register. For bit assignments, see [Table 52](#).

Table 56. Clock control clear register 1 (AHBCLKCTRLCLR1, address 0x4000 00D4) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|--|-------------|
| 31:0 | CLK_CLR1 | Writing ones to this register clears the corresponding bit or bits in the AHBCLKCTRL1 register, if they are implemented. Bits that do not correspond to defined bits in AHBCLKCTRL1 are reserved and only zeroes should be written to them. | - |

4.5.28 SYSTICK clock divider register

This register configures the SYSTICK peripheral clock. The SYSTICK timer clock can be shut down by setting the DIV field to zero.

Table 57. SYSTICK clock divider (SYSTICKCLKDIV, address 0x4000 00E0) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 7:0 | DIV | SYSTICK clock divider value. 0: Disable SYSTICK timer clock. 1: Divide by 1. to 255: Divide by 255. | 0 |
| 31:8 | - | Reserved. Read value is undefined, only zero should be written. | - |

4.5.29 System clock divider register

This register controls how the main clock is divided to provide the system clock to the CPU, AHB bus, and memories. The system clock can be shut down completely by setting the DIV field to zero.

Table 58. System clock divider register (AHBCLKDIV, address 0x4000 0100) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|--|-------------|
| 7:0 | DIV | System AHB clock divider value. 0: System clock disabled. 1: Divide by 1. to 255: Divide by 255. | 0x01 |
| 31:8 | - | Reserved. Read value is undefined, only zero should be written. | - |

4.5.30 ADC clock source divider register

This register divides the clock to the ADC. The clock can be shut down by setting the DIV bits to 0x0.

Table 59. ADC clock source divider (ADCCLKDIV, address 0x4000 0108) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 7:0 | DIV | ADC clock divider value. 0: Disable ADC clock. 1: Divide by 1. to 255: Divide by 255. | 0 |
| 31:8 | - | Reserved. Read value is undefined, only zero should be written. | - |

4.5.31 CLKOUT clock divider register

This register determines the divider value for the clock signal on the CLKOUT pin.

Table 60. CLKOUT clock divider register (CLKOUTDIV, address 0x4000 010C) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 7:0 | DIV | CLKOUT clock divider value. 0: Disable CLKOUT clock divider. 1: Divide by 1. to 255: Divide by 255. | 0 |
| 31:8 | - | Reserved. Read value is undefined, only zero should be written. | - |

4.5.32 Frequency measure function control register

This register starts the frequency measurement function and stores the result in the CAPVAL field. The target frequency can be calculated as follows with the frequencies given in MHz:

$$F_{\text{target}} = (\text{CAPVAL} - 2) \times F_{\text{reference}} / 2^{14}$$

Select the target and reference frequencies using the

Table 61. Frequency measure function control register (FREQMCTRL, address 0x4000 0120) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 13:0 | CAPVAL | Stores the capture result which is used to calculate the frequency of the target clock. This field is read-only. | 0 |
| 30:14 | - | Reserved. Read value is undefined, only zero should be written. | - |
| 31 | PROG | Set this bit to one to initiate a frequency measurement cycle. Hardware clears this bit when the measurement cycle has completed and there is valid capture data in the CAPVAL field (bits 13:0). | 0 |

See [Section 4.2.3 “Measure the frequency of a clock signal”](#), [Section 4.6.5 “Frequency measure function”](#), [Section 9.6.4 “Frequency measure function reference clock select register”](#), and [Section 9.6.5 “Frequency measure function target clock select register”](#) for more on this function.

4.5.33 Flash configuration register

Depending on the system clock frequency, access to the flash memory can be configured with various access times by writing to the FLASHCFG register. It is recommended to use the set_voltage Power API (see [Section 31.4.2](#)) to configure device operation in order to achieve lower power operation. However, flash timing can also be set up by user software as shown in the table.

Enabling buffering, acceleration, and prefetch will substantially improve performance. Buffering saves power by allowing previously accessed information to be reused without a flash read. Acceleration saves power by reducing CPU stalls. Prefetch typically has a small power cost due to some flash reads being performed that ultimately are not needed

Remark: Improper setting of this register may result in incorrect operation of the flash memory. Do not change the flash access time when using the power API in low-power mode.

Table 62. Flash configuration register (FLASHCFG, main syscon: offset 0x400) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|----------|-------|--|-------------|
| 1:0 | FETCHCFG | | Instruction fetch configuration. This field determines how flash accelerator buffers are used for instruction fetches. | 0x2 |
| | | 00 | Instruction fetches from flash are not buffered. Every fetch request from the CPU results in a read of the flash memory. This setting may use significantly more power than when buffering is enabled. | |
| | | 01 | One buffer is used for all instruction fetches. | |
| | | 10 | All buffers may be used for instruction fetches. | |
| | | 11 | Reserved setting, do not use. | |
| 3:2 | DATACFG | | Data read configuration. This field determines how flash accelerator buffers are used for data accesses. | 0x2 |
| | | 00 | Data accesses from flash are not buffered. Every data access from the CPU results in a read of the flash memory. | |
| | | 01 | One buffer is used for all data accesses. | |
| | | 10 | All buffers may be used for data accesses. | |
| | | 11 | Reserved setting, do not use. | |
| 4 | ACCEL | | Acceleration enable. | 1 |
| | | 0 | Flash acceleration is disabled. Every flash read (including those fulfilled from a buffer) takes FLASHTIM + 1 system clocks. This allows more determinism at a cost of performance. | |
| | | 1 | Flash acceleration is enabled. Performance is enhanced, dependent on other FLASHCFG settings. | |
| 5 | PREFEN | | Prefetch enable. | 0 |
| | | 0 | No instruction prefetch is performed. | |
| | | 1 | If the FETCHCFG field is not 0, the next flash line following the current execution address is automatically prefetched if it is not already buffered. | |
| 6 | PREFOVR | | Prefetch override. This bit only applies when PREFEN = 1 and a buffered instruction is completing for which the next flash line is not already buffered or being prefetched. | 0 |
| | | 0 | Any previously initiated prefetch will be completed. | |
| | | 1 | Any previously initiated prefetch will be aborted, and the next flash line following the current execution address will be prefetched if not already buffered. | |

Table 62. Flash configuration register (FLASHCFG, main syscon: offset 0x400) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|----------|--------|--|-------------|
| 11:7 | - | - | Reserved | - |
| 15:12 | FLASHTIM | | Flash memory access time. The number of system clocks used for flash accesses is equal to FLASHTIM +1. | 0x0 |
| | | 0x0 | 1 system clock flash access time (for system clock rates up to 12 MHz). | |
| | | 0x1 | 2 system clocks flash access time (for system clock rates up to 30 MHz). | |
| | | 0x2 | 3 system clocks flash access time (for system clock rates up to 60 MHz). | |
| | | 0x3 | 4 system clocks flash access time (for system clock rates up to 85 MHz). | |
| | | 0x4 | 5 system clocks flash access time (for system clock rates up to 100 MHz). | |
| | | others | "Value" + 1 system clocks flash access time. | |
| 31:16 | - | - | Reserved | - |

4.5.34 FIFO control register

This register is used to enable the System FIFO to provide DMA requests for individual peripheral FIFOs. This replaces the specific peripheral DMA request.

Table 63. FIFO control register (FIFOCTRL, address 0x4000 0148) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------------|--------------------------------|-------------|
| 0 | U0TXFIFOEN | USART0 transmitter FIFO enable | 0 |
| 1 | U1TXFIFOEN | USART1 transmitter FIFO enable | 0 |
| 2 | U2TXFIFOEN | USART2 transmitter FIFO enable | 0 |
| 3 | U3TXFIFOEN | USART3 transmitter FIFO enable | 0 |
| 4 | SPI0TXFIFOEN | SPI0 transmitter FIFO enable | 0 |
| 5 | SPI1TXFIFOEN | SPI1 transmitter FIFO enable | 0 |
| 6 | - | Reserved | - |
| 7 | - | Reserved | - |
| 8 | U0RXFIFOEN | USART0 receiver FIFO enable | 0 |
| 9 | U1RXFIFOEN | USART1 receiver FIFO enable | 0 |
| 10 | U2RXFIFOEN | USART2 receiver FIFO enable | 0 |
| 11 | U3RXFIFOEN | USART3 receiver FIFO enable | 0 |
| 12 | SPI0RXFIFOEN | SPI0 receiver FIFO enable | 0 |
| 13 | SPI1RXFIFOEN | SPI1 receiver FIFO enable | 0 |
| 31:14 | - | Reserved | - |

4.5.35 IRC control register

This register is used to trim the on-chip 12 MHz oscillator. The trim value is factory-preset and written by the boot code on start-up.

Table 64. IRC control register (IRCCTRL, address 0x4000 0184) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|-------------|---|
| 7:0 | TRIM | Trim value | Initially 0x80. Boot code will alter to a device-specific value. Users should not write to this register. |
| 31:8 | - | Reserved | - |

4.5.36 RTC oscillator control register

This register enables the 32 kHz output of the RTC oscillator. This clock can be used to create the main clock when the PLL input or output is selected as the clock source to the main clock.

Table 65. RTC oscillator control register (RTCOSCCTRL, address 0x4000 0190) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|--------------------------|-------------|
| 0 | EN | | RTC 32 kHz clock enable. | 1 |
| | | 0 | Disabled. RTC clock off. | |
| | | 1 | Enabled. RTC clock on. | |
| 31:1 | - | - | Reserved | 0 |

4.5.37 PLL registers

The PLL provides a wide range of frequencies and can potentially be used for many on-chip functions. the PLL can be used with or without a spread spectrum clock generator. See [Section 4.6.4 “PLL functional description”](#) for additional details of PLL operation.

4.5.37.1 System PLL control register

The SYSPLLCTRL register provides most of the control over basic selections of PLL modes and operating details.

Table 66. System PLL control register (SYSPLLCTRL, address 0x4000 01B0 bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|-------------------|-------|---|-------------|
| 3:0 | SELR | | Bandwidth select R value | |
| 9:4 | SELI | | Bandwidth select I value | |
| 14:10 | SELP | | Bandwidth select P value | |
| 15 | BYPASS | | PLL bypass control | 1 |
| | | 0 | Disabled. PLL CCO is used to create the PLL output. | |
| 16 | BYPASS CCODIV2 | | Bypass feedback clock divide by 2. | 0 |
| | | 0 | Divide by 2. The CCO feedback clock is divided by 2 in addition to the programmed M divide. | |
| 17 | UPLIMOFF | | Disable upper frequency limiter. For spread spectrum mode: SEL_EXT = 0, BANDSEL = 0, and UPLIMOFF = 1. | 0 |
| | | 0 | Normal mode. | |
| 18 | BANDSEL | | PLL filter control. Set this bit to one when the spread spectrum controller is disabled or at low frequencies. For spread spectrum mode: SEL_EXT = 0, BANDSEL = 0, and UPLIMOFF = 1. | 0 |
| | | 0 | SSCG control. The PLL filter uses the parameters derived from the spread spectrum controller. | |
| 19 | DIRECTI | | PLL0 direct input enable | 0 |
| | | 0 | Disabled. The PLL input divider (N divider) output is used to drive the PLL CCO. | |
| 20 | DIRECTO | | PLL0 direct output enable | 0 |
| | | 0 | Disabled. The PLL output divider (P divider) is used to create the PLL output. | |
| 31:21 | - | | Reserved. Read value is undefined, only zero should be written. | - |
| | | 1 | Enabled. The PLL output divider (P divider) is bypassed, the PLL CCO output is used as the PLL output. | |

4.5.37.2 System PLL status register

The read-only PLL0_STAT register provides the PLL lock status

Remark: The lock status does not reliably indicate the PLL status for the following two configurations: spread-spectrum mode or fractional enabled or low input clock frequencies such as 32 kHz. In these cases, refer to the PLL lock times listed in the specific device data sheet to obtain appropriate wait times for the PLL to lock.

Table 67. System PLL status register (SYSPLLSTAT, address 0x4000 01B4) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---------------------|-------------|
| 0 | LOCK | PLL0 lock indicator | 0 |
| 31:1 | - | Reserved | - |

4.5.37.3 System PLL N-divider register

Remark: The PLL N-divider register does not use the direct binary representation of N divide value directly. Instead, it uses an encoded version NDEC.

Remark: While the PLL0 output is in use, do not change the NDEC value. Changing the NDEC value changes the FCCO frequency and can cause the system to fail.

- The valid range for N is 1 to 2⁸. This value is encoded into a 10-bit NDEC value. The relationship can be expressed through the following pseudo-code:

```

N_max=0x00000100, x=0x00000080;
switch (N) {
    case 0: x = 0xFFFFFFFF;
    case 1: x = 0x00000302;
    case 2: x = 0x00000202;

    default: for (i = N; i <= N_max; i++)
                x = (((x ^ (x>>2) ^ (x>>3) ^ (x>>4)) & 1) << 7) |
                ((x>>1) & 0x7F); }
NENC[9:0] = x;
    
```

Table 68. System PLL N-divider register (SYSPLLNDEC, address 0x4000 01B8) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 9:0 | NDEC | Decoded N-divider coefficient value | 0 |
| 10 | NREQ | NDEC reload request. When a 1 is written to this bit, the NDEC value is loaded into the PLL. Must be cleared by software for any subsequent load, or the PLL can be powered down and back up via the PDEN_SYS_PLL bit in the PDRUNCFG register if the NDEC value is changed. | 0 |
| 31:11 | - | Reserved. Read value is undefined, only zero should be written. | - |

4.5.37.4 System PLL P-divider register

Remark: The PLL P-divider register does not use the direct binary representation of P divide value directly. Instead, it uses an encoded version PDEC.

Remark: While the PLL0 output is in use, do not change the PDEC value. Changing the PDEC value changes the PLL output frequency and can cause the system to fail.

- The valid range for P is from 1 to 2⁵. This value is encoded into a 7-bit PDEC value. The relationship can be expressed through the following pseudo-code:

```
P_max=0x20, x=0x10;
switch (P) {
    case 0: x = 0xFFFFFFFF;
    case 1: x = 0x00000062;
    case 2: x = 0x00000042;

    default: for (i = P; i <= P_max; i++)
                x = (((x ^ (x>>2)) & 1) << 4) | ((x>>1) & 0xF); }
PDEC[6:0] = x;
```

Table 69. System PLL P-divider register (SYSPLLDEC, address 0x4000 01BC) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|--|-------------|
| 6:0 | PDEC | Decoded P-divider coefficient value | 0 |
| 7 | PREQ | PDEC reload request. When a 1 is written to this bit, the PDEC value is loaded into the PLL. Must be cleared by software for any subsequent load, or the PLL can be powered down and back up via the PDEN_SYS_PLL bit in the PDRUNCFG register if the PDEC value is changed. | 0 |
| 31:8 | - | Reserved. Read value is undefined, only zero should be written. | - |

4.5.37.5 Spread spectrum control with PLL0

The spread spectrum functionality can be used to provide a spread spectrum clock, which can decrease electromagnetic interference (EMI).

The Spread Spectrum Clock Generator can be used in several ways:

- It can encode M-divider values between 1 and 255 to produce the MDEC value used directly by the PLL, saving the need for executing encoding algorithm code, or hard-coding predetermined values into an application.
- It can provide a fractional rate feature to the PLL.
- It can be set up to automatically alter the PLL CCO frequency on an ongoing basis to decrease electromagnetic interference (EMI).

If the spread spectrum mode is enabled, choose N to ensure 2 MHz < Fin/N < 4 MHz. Spread spectrum mode cannot be used when Fin = 32 kHz.

When the modulation (MR) is set to zero, the PLL becomes a fractional PLL.

4.5.37.5.1 System PLL spread spectrum control register 0

Table 70. System PLL spread spectrum control register 0 (SYSPLLSSCTRL0, address 0x4000 01C0) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|---------|-------|--|-------------|
| 16:0 | MDEC | | Decoded M-divider coefficient value | 0 |
| 17 | MREQ | | MDEC reload request. When a 1 is written to this bit, the MDEC value is loaded into the PLL. Must be cleared by software for any subsequent load, or the PLL can be powered down and back up via the PDEN_SYS_PLL bit in the PDRUNCFG register if the MDEC value is changed. | 0 |
| 18 | SEL_EXT | | Select spread spectrum mode. Selects the source of the feedback divider value. For normal mode, this must be the value from the MDEC field in this register. For spread spectrum mode: SEL_EXT = 0, BANDSEL = 0, and UPLIMOFF = 1. | 0 |
| | | 0 | The PLL feedback divider value comes from the spread spectrum controller. | |
| | | 1 | The PLL feedback divider value comes from the MDEC field in this register. | |
| 31:19 | - | | Reserved. Read value is undefined, only zero should be written. | - |

PLL0 M-divider register: The PLL0 M-divider value (MDEC) can be set directly if the PLL0 is not used with the spread spectrum clock generator (SSCG). If the SSCG is enabled via the SEL_EXT bit, then the SSCG sets the MDEC value.

Remark: MDEC does not use the direct binary representations of M directly. Instead, it uses an encoded version of M. The valid range for M is 1 to 2¹⁵. This value is encoded into a 17-bit MDEC value.

The relationship between M and MDEC is expressed via the following pseudo-code.

```

M_max=0x00008000, x=0x00004000;
switch (M) {
    case 0: x = 0xFFFFFFFF;
    case 1: x = 0x00018003;
    case 2: x = 0x00010003;

    default: for (i = M; i <= M_max; i++)
                x = (((x ^ (x>>1)) & 1) << 14) | ((x>>1) & 0x3FFF); }
MDEC[16:0] = x;
    
```

The values for SELP, SELI, and SELR depend on the value for M as expressed by the following pseudo-code:

```

if (M < 60) then
    SELP = (M>>1) + 1
else
    SELP = 31;
if (M > 16384) then
    SELI = 1
else if (M > 8192) then
    SELI = 2
else if (M > 2048) then
    SELI = 4
else if (M >= 501) then
    SELI = 8
    
```

```

else if (M >=60) then
    SELI = 4*(1024/(M+9))
else
    SELI = (M & 0x3C) + 4; /* & denotes bitwise AND */
SELR = 0;

```

Remark: The values for SELP, SELI, and SELR are generated by the encoding block when the spread spectrum clock generator is enabled and need not be programmed explicitly.

Remark: While the PLL0 output is in use, do not change the MDEC value. Changing the MDEC value changes the FCCO frequency and can cause the system to fail.

4.5.37.5.2 System PLL spread spectrum control register 1

Table 71. System PLL spread spectrum control register 1 (SYSPLLSSCTRL1, address 0x4000 01C4) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|--|-------------|
| 18:0 | MD | | M- divider value with fraction. MD[18:11]: integer portion of the feedback divider value. MD[10:0]: fractional portion of the feedback divider value. | 0 |
| 19 | MDREQ | | MD reload request. When a 1 is written to this bit, the MD value is loaded into the PLL. This bit is cleared when the load is complete. | 0 |
| 22:20 | MF | | Programmable modulation frequency $f_m = F_{ref}/N_{ss}$ with $F_{ref} = F_{in}/N$ 0b000 => $N_{ss} = 512$ ($f_m \approx 3.9 - 7.8$ kHz) 0b001 => $N_{ss} \approx 384$ ($f_m \approx 5.2 - 10.4$ kHz) 0b010 => $N_{ss} = 256$ ($f_m \approx 7.8 - 15.6$ kHz) 0b011 => $N_{ss} = 128$ ($f_m \approx 15.6 - 31.3$ kHz) 0b100 => $N_{ss} = 64$ ($f_m \approx 32.3 - 64.5$ kHz) 0b101 => $N_{ss} = 32$ ($f_m \approx 62.5 - 125$ kHz) 0b110 => $N_{ss} \approx 24$ ($f_m \approx 83.3 - 166.6$ kHz) 0b111 => $N_{ss} = 16$ ($f_m \approx 125 - 250$ kHz) | 0 |
| 25:23 | MR | | Programmable frequency modulation depth $\delta f_{modpk-pk} = F_{ref} \times k / F_{cco} = k / MD_{dec}$ 0 = no spread 0b000 => $k = 0$ (no spread spectrum) 0b001 => $k \approx 1$ 0b010 => $k \approx 1.5$ 0b011 => $k \approx 2$ 0b100 => $k \approx 3$ 0b101 => $k \approx 4$ 0b110 => $k \approx 6$ 0b111 => $k \approx 8$ | 0 |
| 27:26 | MC | | Modulation waveform control 0 = no compensation Compensation for low pass filtering of the PLL to get a triangular modulation at the output of the PLL, giving a flat frequency spectrum. 0b00 => no compensation 0b10 => recommended setting 0b11 => max. compensation | 0 |

Table 71. System PLL spread spectrum control register 1 (SYSPLLSSCTRL1, address 0x4000 01C4) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|---|-------------|
| 28 | PD | | Power down. | 1 |
| | | 0 | Enabled. Spread spectrum controller is enabled | |
| | | 1 | Disabled. Spread spectrum controller is disabled | |
| 29 | DITHER | | Select modulation frequency. | 0 |
| | | 0 | Fixed. Fixed modulation frequency. | |
| | | 1 | Dither. Randomly dither between two modulation frequencies. | |
| 31:30 | - | | Reserved. Read value is undefined, only zero should be written. | - |

4.5.38 Power Configuration register

The PDRUNCFG register controls the power to the various analog blocks.

Remark: for safety, this register should not be written. Changing the contents of PDRUNCFG should be accomplished by writing to PDRUNCFGSET and/or PDRUNCFGCLR. This prevents inadvertent changes to unintended bits. **Reserved bits must not be changed by user software.**

Regarding bits 13 through 16, see [Section 2.1.1](#) for details of SRAM configuration.

Table 72. Power Configuration register (PDRUNCFG, address 0x4000 0210) bit description

| Bit | Symbol | Description | Reset value |
|-------|---------------|--|-------------|
| 2:0 | - | . | - |
| 3 | PDEN_IRC_OSC | IRC oscillator output. 0 = Powered; 1 = Powered down. | 0 |
| 4 | PDEN_IRC | IRC oscillator. 0 = Powered; 1 = Powered down. | 0 |
| 5 | PDEN_FLASH | Flash memory. 0 = Powered; 1 = Powered down. | 0 |
| 6 | - | Reserved. | - |
| 7 | PDEN_BOD_RST | Brown-out Detect reset. 0 = Powered; 1 = Powered down. | 0 |
| 8 | PDEN_BOD_INTR | Brown-out Detect interrupt. 0 = Powered; 1 = Powered down. | 1 |
| 9 | - | Reserved. | - |
| 10 | PDEN_ADC0 | ADC0. 0 = Powered; 1 = Powered down. | 1 |
| 12:11 | - | Reserved. | - |
| 13 | PDEN_SRAM0A | First 8 KB of SRAM0. 0 = Powered; 1 = Powered down. | 0 |
| 14 | PDEN_SRAM0B | Remaining portion of SRAM0. 0 = Powered; 1 = Powered down. | 0 |
| 15 | PDEN_SRAM1 | SRAM1. 0 = Powered; 1 = Powered down. | 0 |
| 16 | PDEN_SRAM2 | SRAM2 (undedicated 8 KB RAM). 0 = Powered; 1 = Powered down. | 0 |
| 17 | PDEN_ROM | ROM. 0 = Powered; 1 = Powered down. | 0 |
| 18 | - | Reserved. | - |
| 19 | PDEN_VDDA | Vdda to the ADC, must be enabled for the ADC to work. Also see bit 23. 0 = Powered; 1 = Powered down. | 1 |
| 20 | PDEN_WDT_OSC | Watchdog oscillator. 0 = Powered; 1 = Powered down. | 1 |
| 21 | - | Reserved. | - |
| 22 | PDEN_SYS_PLL | PLL0. 0 = Powered; 1 = Powered down. | 1 |
| 23 | PDEN_VREFP | Vrefp to the ADC, must be enabled for the ADC to work. Also see bit 19. 0 = Powered; 1 = Powered down. | 1 |
| 24 | PDEN_32K_OSC | 32 kHz RTC oscillator. 0 = Powered; 1 = Powered down. | 0 |
| 31:25 | - | Reserved. | - |

4.5.39 Power configuration set register

Writing a 1 to a bit position in PDRUNCFGSET sets the corresponding position in PDRUNCFG. This is a write-only register. For bit assignments, see [Table 72](#).

Table 73. Power configuration set register (PDRUNCFGSET, address 0x4000 0214) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|--|-------------|
| 31:0 | PD_SET | Writing ones to this register sets the corresponding bit or bits in the PDRUNCFG register, if they are implemented. Bits that do not correspond to defined bits in PDRUNCFG are reserved and only zeroes should be written to them. | - |

4.5.40 Power configuration clear register

Writing a 1 to a bit position in PDRUNCFGCLR clears the corresponding position in PDRUNCFG. This is a write-only register. For bit assignments, see [Table 72](#).

Table 74. Power configuration clear register (PDRUNCFGCLR, address 0x4000 0218) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|--|-------------|
| 31:0 | PD_CLR | Writing ones to this register clears the corresponding bit or bits in the PDRUNCFG register, if they are implemented. Bits that do not correspond to defined bits in PDRUNCFG are reserved and only zeroes should be written to them. | - |

4.5.41 Start enable register 0

The STARTER0 and STARTER1 registers enable an interrupt for wake-up from deep-sleep and power-down modes.

Some interrupts are typically used in sleep mode only and will not occur during deep-sleep or power-down modes because relevant clocks are stopped. However, it is possible to enable those clocks (significantly increasing power consumption in the reduced power mode), making these wake-ups possible.

The pattern match feature of the pin interrupt requires a clock in order to operate, and will not wake up the device from reduced power modes beyond Sleep mode.

Whether peripheral interrupts can occur during deep-sleep or power-down modes depends on the peripheral, its configuration, and system setup.

Remark: Also enable the corresponding interrupts in the NVIC. See [Table 4 “Interrupt Set-Enable Register 0 register”](#).

Table 75. Start enable register 0 (STARTER0, address 0x4000 0240) bit description

| Bit | Symbol | Description | Reset value |
|-----|--------|--|-------------|
| 0 | WWDT | WWDT interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. | 0 |
| 1 | BOD | BOD interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. | 0 |
| 2 | - | Reserved. Read value is undefined, only zero should be written. | - |
| 3 | DMA | DMA wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Typically used in sleep mode only since the peripheral clock must be running for it to function. | 0 |
| 4 | GINT0 | Group interrupt 0 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. | 0 |
| 5 | PINT0 | GPIO pin interrupt 0 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match. | 0 |
| 6 | PINT1 | GPIO pin interrupt 1 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match. | 0 |
| 7 | PINT2 | GPIO pin interrupt 2 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match. | 0 |
| 8 | PINT3 | GPIO pin interrupt 3 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match. | 0 |
| 9 | UTICK | Micro-tick Timer wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. | 0 |
| 10 | MRT | Multi-Rate Timer wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Typically used in sleep mode only since the peripheral clock must be running for it to function. | 0 |
| 11 | CT32B0 | Standard counter/timer CT32B0 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Typically used in sleep mode only. | 0 |
| 12 | CT32B1 | Standard counter/timer CT32B1 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Typically used in sleep mode only since the peripheral clock must be running for it to function. | 0 |
| 13 | CT32B2 | Standard counter/timer CT32B2 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Typically used in sleep mode only since the peripheral clock must be running for it to function. | 0 |
| 14 | CT32B3 | Standard counter/timer CT32B3 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Typically used in sleep mode only since the peripheral clock must be running for it to function. | 0 |

Table 75. Start enable register 0 (STARTER0, address 0x4000 0240) bit description ...continued

| Bit | Symbol | Description | Reset value |
|-----|------------|---|-------------|
| 15 | CT32B4 | Standard counter/timer CT32B4 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Typically used in sleep mode only since the peripheral clock must be running for it to function. | 0 |
| 16 | SCT0 | SCT0 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Typically used in sleep mode only since the peripheral clock must be running for it to function. | 0 |
| 17 | USART0 | USART0 interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Peripheral interrupt. | 0 |
| 18 | USART1 | USART1 interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Peripheral interrupt. | 0 |
| 19 | USART2 | USART2 interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Peripheral interrupt. | 0 |
| 20 | USART3 | USART2 interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Peripheral interrupt. | 0 |
| 21 | I2C0 | I2C0 interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Peripheral interrupt. | 0 |
| 22 | I2C1 | I2C1 interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Peripheral interrupt. | 0 |
| 23 | I2C2 | I2C2 interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Peripheral interrupt. | 0 |
| 24 | SPI0 | SPI0 interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Peripheral interrupt. | 0 |
| 25 | SPI1 | SPI1 interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Peripheral interrupt. | 0 |
| 26 | ADC0_SEQA | ADC0 sequence A interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Typically used in sleep mode only since the peripheral clock must be running for it to function. | 0 |
| 27 | ADC0_SEQB | ADC0 sequence B interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Typically used in sleep mode only since the peripheral clock must be running for it to function. | 0 |
| 28 | ADC0_THCMP | ADC0 threshold and error interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Typically used in sleep mode only since the peripheral clock must be running for it to function. | 0 |
| 29 | RTC | RTC interrupt alarm and wake-up timer. 0 = Wake-up disabled. 1 = Wake-up enabled. | 0 |
| 30 | - | Reserved. Read value is undefined, only zero should be written. | - |
| 31 | MAILBOX | Mailbox interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. At least one CPU must be running in order for a mailbox interrupt to occur. Present on LPC54102 devices. | 0 |

4.5.42 Start enable register 1

The STARTER1 register selects additional interrupts that may wake up the part from deep-sleep and power-down modes.

Some interrupts are typically used in sleep mode only and will not occur during deep-sleep or power-down modes because relevant clocks are stopped. However, it is possible to enable those clocks (significantly increasing power consumption in the reduced power mode), making these wake-ups possible.

The pattern match feature of the pin interrupt requires a clock in order to operate, and will not wake up the device from reduced power modes beyond Sleep mode.

Remark: Also enable the corresponding interrupts in the NVIC. See [Table 5 "Interrupt Set-Enable Register 1 register"](#).

Table 76. Start enable register 1 (STARTER1, address 0x4000 0244) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 0 | GINT1 | Group interrupt 0 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. | 0 |
| 1 | PINT4 | GPIO pin interrupt 4 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match. | 0 |
| 2 | PINT5 | GPIO pin interrupt 5 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match. | 0 |
| 3 | PINT6 | GPIO pin interrupt 6 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match. | 0 |
| 4 | PINT7 | GPIO pin interrupt 7 wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Not for pattern match. | 0 |
| 7:5 | - | Reserved. Read value is undefined, only zero should be written. | 0 |
| 8 | RIT | Repetitive Interrupt Timer interrupt wake-up. 0 = Wake-up disabled. 1 = Wake-up enabled. Typically used in sleep mode only since the peripheral clock must be running for it to function. | 0 |
| 31:15 | - | Reserved. Read value is undefined, only zero should be written. | - |

4.5.43 Start enable set register 0

Writing a 1 to a bit position in STARTERSET0 sets the corresponding position in STARTER0. This is a write-only register. For bit assignments, see [Table 75](#).

Table 77. Start enable set register 0 (STARTERSET0, address 0x4000 0248) bit description

| Bit | Symbol | Description | Reset value |
|------|------------|--|-------------|
| 31:0 | START_SET0 | Writing ones to this register sets the corresponding bit or bits in the STARTER0 register, if they are implemented. Bits that do not correspond to defined bits in STARTER0 are reserved and only zeroes should be written to them. | - |

4.5.44 Start enable set register 1

Writing a 1 to a bit position in STARTERSET1 sets the corresponding position in STARTER1. This is a write-only register. For bit assignments, see [Table 76](#).

Table 78. Start enable set register 1 (STARTERSET1, address 0x4000 024C) bit description

| Bit | Symbol | Description | Reset value |
|------|------------|--|-------------|
| 31:0 | START_SET1 | Writing ones to this register sets the corresponding bit or bits in the STARTER1 register, if they are implemented. Bits that do not correspond to defined bits in STARTER1 are reserved and only zeroes should be written to them. | - |

4.5.45 Start enable clear register 0

Writing a 1 to a bit position in STARTERCLR0 clears the corresponding position in STARTER0. This is a write-only register. For bit assignments, see [Table 75](#).

Table 79. Start enable clear register 0 (STARTERCLR0, address 0x4000 0250) bit description

| Bit | Symbol | Description | Reset value |
|------|------------|--|-------------|
| 31:0 | START_CLR0 | Writing ones to this register clears the corresponding bit or bits in the STARTER0 register, if they are implemented. Bits that do not correspond to defined bits in STARTER0 are reserved and only zeroes should be written to them. | - |

4.5.46 Start enable clear register 1

Writing a 1 to a bit position in STARTERCLR1 clears the corresponding position in STARTER1. This is a write-only register. For bit assignments, see [Table 76](#).

Table 80. Start enable clear register 1 (STARTERCLR1, address 0x4000 0254) bit description

| Bit | Symbol | Description | Reset value |
|------|------------|--|-------------|
| 31:0 | START_CLR1 | Writing ones to this register clears the corresponding bit or bits in the STARTER1 register, if they are implemented. Bits that do not correspond to defined bits in STARTER1 are reserved and only zeroes should be written to them. | - |

4.5.47 Dual-CPU related registers

These registers control usage aspects of the two CPUs in an LPC54102 device. They are not used in an LPC54101 device that only provide a single CPU.

4.5.47.1 CPU Control register

The CPUCTRL register provides control for the 2 CPUs. Note that the Cortex-M4 is factory set to be the master. The master CPU cannot be reset or have its clock disabled via this register. Only the master CPU can use the Power APIs to cause the device to enter reduced power modes.

If the clock to the slave CPU is to be disabled at some point in the application for power savings, that CPU should have entered its own sleep mode prior to that point. This avoids incomplete operations in the slave CPU.

Table 81. CPU Control register (CPUCTRL, address 0x4000 0300) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|-----------|-------|--|-------------|
| 0 | MASTERCPU | | Indicates which CPU is considered the master. This is factory set assign the Cortex-M4 as the master. | 1 |
| | | | The master CPU cannot have its clock turned off via the related CMnCLKEN bit or be reset via the related CMxRSTEN in this register. | |
| | | | The slave CPU wakes up briefly following device reset, then goes back to sleep until activated by the master CPU. | |
| | | 0 | M0+. Cortex-M0+ is the master CPU. | |
| | | 1 | M4. Cortex-M4 is the master CPU. | |
| 1 | - | - | Reserved. Read value is undefined, only zero should be written. | - |
| 2 | CM4CLKEN | | Cortex-M4 clock enable. | 1 |
| | | 0 | Disabled. The Cortex-M4 clock is not enabled. | |
| | | 1 | Enabled. The Cortex-M4 clock is enabled. | |
| 3 | CM0CLKEN | | Cortex-M0+ clock enable. | 1 |
| | | 0 | Disabled. The Cortex-M0+ clock is not enabled. | |
| | | 1 | Enabled. The Cortex-M0+ clock is enabled. | |
| 4 | CM4RSTEN | | Cortex-M4 reset. | 0 |
| | | 0 | Disabled. The Cortex-M4 is not being reset. | |
| | | 1 | Enabled. The Cortex-M4 is being reset. | |
| 5 | CM0RSTEN | | Cortex-M0+ reset. | 0 |
| | | 0 | Disabled. The Cortex-M0+ is not being reset. | |
| | | 1 | Enabled. The Cortex-M0+ is being reset. | |
| 6 | POWERCPU | | Identifies the owner of reduced power mode control: which CPU can cause the device to enter Deep Sleep, Power-down, and Deep Power-down modes. | 1 |
| | | 0 | M0+. Cortex-M0+ is the owner of reduced power mode control. | |
| | | 1 | M4. Cortex-M4 is the owner of reduced power mode control. | |
| 14:7 | - | - | Reserved. Read value is undefined, only zero should be written. | - |
| 15 | - | - | Must be written as a 1. | - |
| 31:16 | - | - | Must be written as 0xC0C4 for the write to have an effect. | - |

4.5.47.2 Coprocessor Boot register

CPBOOT can be used in an application that uses both CPUs in order to send the slave processor (the CPU not selected as the master by the MASTERCPU bit in the CPUCTRL register) to an appropriate boot address that is different than the master CPU.

Table 82. Coprocessor Boot register (CPBOOT, address 0x4000 0304) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|-------------------------------|-------------|
| 31:0 | BOOTADDR | Slave processor boot address. | 0 |

4.5.47.3 Coprocessor Stack register

CPSTACK can be used in an application that uses both CPUs in order to set up the stack for the slave processor (the CPU not selected as the master by the MASTERCPU bit in the CPUCTRL register) to an appropriate address that is different than the master CPU.

Table 83. Coprocessor Stack register (CPSTACK, address 0x4000 0308) bit description

| Bit | Symbol | Description | Reset value |
|------|-----------|--------------------------------|-------------|
| 31:0 | STACKADDR | Slave processor stack address. | 0 |

4.5.47.4 Coprocessor Status register

CPU_STAT provides some status for dual CPUs. This register can be read by software at run time, or with a debugger.

Table 84. Coprocessor Status register (CPSTAT, address 0x4000 030C) bit description

| Bit | Symbol | Description | Reset value |
|------|-------------|---|-------------|
| 0 | CM4SLEEPING | When 1, the Cortex-M4 CPU is sleeping. | 0 |
| 1 | CM0SLEEPING | When 1, the Cortex-M0+ CPU is sleeping. | 0 |
| 2 | CM4LOCKUP | When 1, the Cortex-M4 CPU is in lockup. | 0 |
| 3 | CM4LOCKUP | When 1, the Cortex-M0+ CPU is in lockup. | 0 |
| 31:4 | - | Reserved. Read value is undefined, only zero should be written. | - |

4.5.48 JTAG ID code register

This register contains the JTAG ID code.

Table 85. JTAG ID code register (JTAGIDCODE, address 0x4000 03F4) bit description

| Bit | Symbol | Description | Value |
|------|--------|---------------|-------------|
| 31:0 | JTAGID | JTAG ID code. | 0x1FEC E02B |

4.5.49 Device ID0 register

This register contains the part ID. The part ID can also be obtained using the ISP or IAP ReadPartID commands. See [Table 475](#) and [Table 488](#).

Table 86. Device ID0 register (DEVICE_ID0, address 0x4000 03F8) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|-------------|----------------|
| 31:0 | PARTID | Part ID | part dependent |

Table 87. Device ID0 register values

| Part number | Part ID |
|--------------|-------------|
| LPC54101J256 | 0x8845 4101 |
| LPC54101J512 | 0x8885 4101 |
| LPC54102J256 | 0x8845 4102 |
| LPC54102J512 | 0x8885 4102 |

4.5.50 Device ID1 register

This register contains the boot ROM and die revisions.

Table 88. Device ID1 register (DEVICE_ID1, address 0x4000 03FC) bit description

| Bit | Symbol | Description | Value |
|------|--------|-------------|----------------|
| 31:0 | REVID | Revision. | part dependent |

4.5.51 Asynchronous peripheral reset control register

The ASYNCPRESETCTRL register allows software to reset specific peripherals attached to the async APB bridge. Writing a zero to any assigned bit in this register clears the reset and allows the specified peripheral to operate. Writing a one asserts the reset.

Table 89. Asynchronous peripheral reset control register (ASYNCPRESETCTRL, address 0x4008 0000) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 0 | - | Reserved | - |
| 1 | USART0 | USART0 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 2 | USART1 | USART1 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 3 | USART2 | USART2 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 4 | USART3 | USART3 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 5 | I2C0 | I2C0 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | - |
| 6 | I2C1 | I2C1 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 7 | I2C2 | I2C2 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 8 | - | Reserved | - |
| 9 | SPI0 | SPI0 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 10 | SPI1 | SPI1 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 12:11 | - | Reserved | - |
| 13 | CT32B0 | Standard counter/timer CT32B0 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 14 | CT32B1 | Standard counter/timer CT32B1 reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 15 | FRG0 | FRG reset control. 0 = Clear reset to this function. 1 = Assert reset to this function. | 0 |
| 31:16 | - | Reserved | - |

4.5.52 Asynchronous peripheral reset control set register

Writing a 1 to a bit position in ASYNCPRESETCTRLSET sets the corresponding position in ASYNCPRESETCTRL. This is a write-only register. For bit assignments, see [Table 89](#).

Table 90. Asynchronous peripheral reset control set register (ASYNCPRESETCTRLSET, address 0x4008 0004) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|--|-------------|
| 31:0 | ARST_SET | Writing ones to this register sets the corresponding bit or bits in the ASYNCPRESETCTRL register, if they are implemented. Bits that do not correspond to defined bits in ASYNCPRESETCTRL are reserved and only zeroes should be written to them. | - |

4.5.53 Asynchronous peripheral reset control clear register

Writing a 1 to a bit position in ASYNCPRESETCTRLCLR clears the corresponding position in PRESETCTRL0. This is a write-only register. For bit assignments, see [Table 89](#).

Table 91. Asynchronous peripheral reset control clear register (ASYNCPRESETCTRLCLR, address 0x4008 0008) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|--|-------------|
| 31:0 | ARST_CLR | Writing ones to this register clears the corresponding bit or bits in the ASYNCPRESETCTRL register, if they are implemented. Bits that do not correspond to defined bits in ASYNCPRESETCTRL are reserved and only zeroes should be written to them. | - |

4.5.54 Asynchronous APB clock control register

This register controls how the clock selected for the asynchronous APB peripherals is divided to provide the clock to the asynchronous peripherals. The clock will be stopped if the DIV field is set to zero.

Table 92. Asynchronous APB clock control register (ASYNCAPBCLKCTRL, address 0x4008 0010) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 0 | - | Reserved. Read value is undefined, only zero should be written. | - |
| 1 | USART0 | Controls the clock for USART0. 0 = Disable; 1 = Enable. | 0 |
| 2 | USART1 | Controls the clock for USART1. 0 = Disable; 1 = Enable. | 0 |
| 3 | USART2 | Controls the clock for USART2. 0 = Disable; 1 = Enable. | 0 |
| 4 | USART3 | Controls the clock for USART3. 0 = Disable; 1 = Enable. | 0 |
| 5 | I2C0 | Controls the clock for I2C0. 0 = Disable; 1 = Enable. | |
| 6 | I2C1 | Controls the clock for I2C1. 0 = Disable; 1 = Enable. | 0 |
| 7 | I2C2 | Controls the clock for I2C2. 0 = Disable; 1 = Enable. | 0 |
| 8 | - | Reserved. Read value is undefined, only zero should be written. | - |
| 9 | SPI0 | Controls the clock for SPI0. 0 = Disable; 1 = Enable. | 0 |
| 10 | SPI1 | Controls the clock for SPI1. 0 = Disable; 1 = Enable. | 0 |
| 12:11 | - | Reserved. Read value is undefined, only zero should be written. | - |
| 13 | CT32B0 | Controls the clock for CT32B0. 0 = Disable; 1 = Enable. | 0 |
| 14 | CT32B1 | Controls the clock for CT32B1. 0 = Disable; 1 = Enable. | 0 |
| 15 | FRG0 | Controls the clock for the Fractional Rate Generator used with the USARTs. 0 = Disable; 1 = Enable. | 0 |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | - |

4.5.55 Asynchronous APB clock control set register

Writing a 1 to a bit position in ASYNCAPBCLKCTRLSET sets the corresponding position in ASYNCAPBCLKCTRL. This is a write-only register. For bit assignments, see [Table 89](#).

Table 93. Asynchronous APB clock control set register (ASYNCAPBCLKCTRLSET, address 0x4008 0014) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|--|-------------|
| 31:0 | ACLK_SET | Writing ones to this register sets the corresponding bit or bits in the ASYNCAPBCLKCTRL register, if they are implemented. Bits that do not correspond to defined bits in ASYNCPRESETCTRL are reserved and only zeroes should be written to them. | - |

4.5.56 Asynchronous APB clock control clear register

Writing a 1 to a bit position in ASYNCAPBCLKCTRLCLR clears the corresponding position in ASYNCAPBCLKCTRL. This is a write-only register. For bit assignments, see [Table 89](#).

Table 94. Asynchronous APB clock control clear register (ASYNCAPBCLKCTRLCLR, address 0x4008 0018) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|--|-------------|
| 31:0 | ACLK_CLR | Writing ones to this register clears the corresponding bit or bits in the ASYNCAPBCLKCTRL register, if they are implemented. Bits that do not correspond to defined bits in ASYNCAPBCLKCTRL are reserved and only zeroes should be written to them. | - |

4.5.57 Asynchronous clock source select register A

This register selects a potential clock for the asynchronous APB peripherals from among several clock sources. The clock selected becomes one of the inputs to the asynchronous clock source select register B (see [Table 96](#)), which selects the final clock source for the asynchronous APB clock.

Remark: Note that this selection is internally synchronized: the clock being switched from and the clock being switched to must both be running and have occurred in specific states before the selection actually changes.

Table 95. Asynchronous clock source select register A (ASYNCAPBCLKSELA, address 0x4008 0020) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---|-------------|
| 1:0 | SEL | | Clock source for asynchronous clock source selector A | 0 |
| | | 0x0 | IRC Oscillator | |
| | | 0x1 | Watchdog oscillator | |
| | | 0x2 | Reserved | |
| | | 0x3 | Reserved | |
| 31:2 | - | - | Reserved | - |

4.5.58 Asynchronous clock source select register B

This register selects the clock source for the asynchronous APB clock.

Remark: Note that this selection is internally synchronized: the clock being switched from and the clock being switched to must both be running and have occurred in specific states before the selection actually changes.

Table 96. Asynchronous clock source select register B (ASYNCAPBCLKSELB, address 0x4008 0024) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|--|-------------|
| 1:0 | SEL | | Clock source for asynchronous clock source selector B. | 0 |
| | | 0x0 | Main clock | |
| | | 0x1 | CLKIN | |
| | | 0x2 | System PLL output. | |
| | | 0x3 | ASYNCAPBCLKSELA. Clock selected by the ASYNCAPBCLKSELA register. | |
| 31:2 | - | - | Reserved | - |

4.5.59 Asynchronous APB clock divider register

This register controls how the asynchronous APB clock is divided before use by peripherals. The clock can be shut down completely by setting the DIV field to zero.

Table 97. Asynchronous APB clock divider register (ASYNCLKDIV, address 0x4008 0028) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 7:0 | DIV | Asynchronous APB clock divider value. 0: Clock disabled. 1: Divide by 1. to 255: Divide by 255. | 0x01 |
| 31:8 | - | Reserved | - |

4.5.60 USART fractional baud rate generator register

All USART peripherals share a common clock (see [Figure 3](#)), which can be adjusted by a fractional divider. This register sets the MULT and DIV values for the fractional rate generator. The output rate is:

$$\text{USART clock} = \text{async bridge clock rate} / (1 + \text{MULT} / \text{DIV})$$

The async bridge clock rate is the USART clock configured as selected via ASYNCAPBCLKSELA ([Section 4.5.57](#)) and ASYNCAPBCLKSELB ([Section 4.5.58](#)), and divided as defined by the ASYNCLKDIV register ([Section 4.5.59](#)).

Remark: In order to use the fractional baud rate generator, 0xFF must first be written to the DIV value to yield a denominator value of 256. All other values are not supported.

See also [Section 22.3.1 “Configure the USART clock and baud rate”](#) and [Section 22.7.1 “Clocking and baud rates”](#)

Table 98. USART fractional baud rate generator register (FRGCTRL, address 0x4008 0030) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 7:0 | DIV | Denominator of the fractional divider. DIV is equal to the programmed value +1. Always set to 0xFF to use with the fractional baud rate generator. | 0xFF |
| 15:8 | MULT | Numerator of the fractional divider. MULT is equal to the programmed value. | 0 |
| 31:16 | - | Reserved | - |

4.5.61 BOD control register

The BOD control register selects four separate threshold values for sending a BOD interrupt to the NVIC and for forced reset. Reset and interrupt threshold values listed in [Table 99](#) are typical values. More details can be found in specific device data sheets.

Both the BOD interrupt and the BOD reset can wake-up the chip from Sleep, Deep-sleep, and Power-down modes if enabled. See [Chapter 31 “LPC5410x Power profiles/Power control API”](#).

Table 99. BOD control register (BODCTRL, address 0x4002 C044) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|------------|-------|---|-------------|
| 1:0 | BODRSTLEV | | BOD reset level | 0 |
| | | 0x0 | Level 0: 1.5 V | |
| | | 0x1 | Level 1: 1.85 V | |
| | | 0x2 | Level 2: 2.0 V | |
| | | 0x3 | Level 3: 2.3 V | |
| 2 | BODRSTENA | | BOD reset enable | 0 |
| | | 0 | Disable reset function. | |
| | | 1 | Enable reset function. | |
| 4:3 | BODINTLEV | | BOD interrupt level | 0 |
| | | 0x0 | Level 0: 2.05 V | |
| | | 0x1 | Level 1: 2.45 V | |
| | | 0x2 | Level 2: 2.75 V | |
| | | 0x3 | Level 3: 3.05 V | |
| 5 | BODINTENA | | BOD interrupt enable | 0 |
| | | 0 | Disable interrupt function. | |
| | | 1 | Enable interrupt function. | |
| 6 | BODRSTSTAT | | BOD reset status. When 1, a BOD reset has occurred. Cleared by writing 1 to this bit. | 0 |
| 7 | BODINTSTAT | | BOD interrupt status. When 1, a BOD interrupt has occurred. Cleared by writing 1 to this bit. | 0 |
| 31:8 | - | - | Reserved | - |

4.6 Functional description

4.6.1 Reset

Reset has the following sources:

- The $\overline{\text{RESET}}$ pin.
- Watchdog reset.
- Power-On Reset (POR).
- Brown Out Detect (BOD).
- ARM software reset.
- ISP-AP debug reset.

Assertion of the POR or the BOD reset, once the operating voltage attains a usable level, starts the IRC. After the IRC-start-up time (maximum of 6 μs on power-up), the IRC provides a stable clock output. The reset remains asserted until the external Reset is released, the oscillator is running, and the flash controller has completed its initialization.

On the assertion of any reset source (ARM software reset, POR, BOD reset, External reset, and Watchdog reset), the following processes are initiated:

1. The IRC is enabled or starts up if not running.
2. The flash wake-up timer starts. This takes approximately 250 ms or less.
3. The boot code in the ROM starts. The boot code performs the boot tasks and may jump to the flash.

When the internal Reset is removed, the processor begins executing at address 0, which is initially the Reset vector mapped from the boot block. At that point, all of the processor and peripheral registers have been initialized to predetermined values.

4.6.2 Start-up behavior

See [Figure 4](#) for the start-up timing after reset. The IRC is the default clock at Reset and provides a clean system clock shortly after the supply pins reach operating voltage.

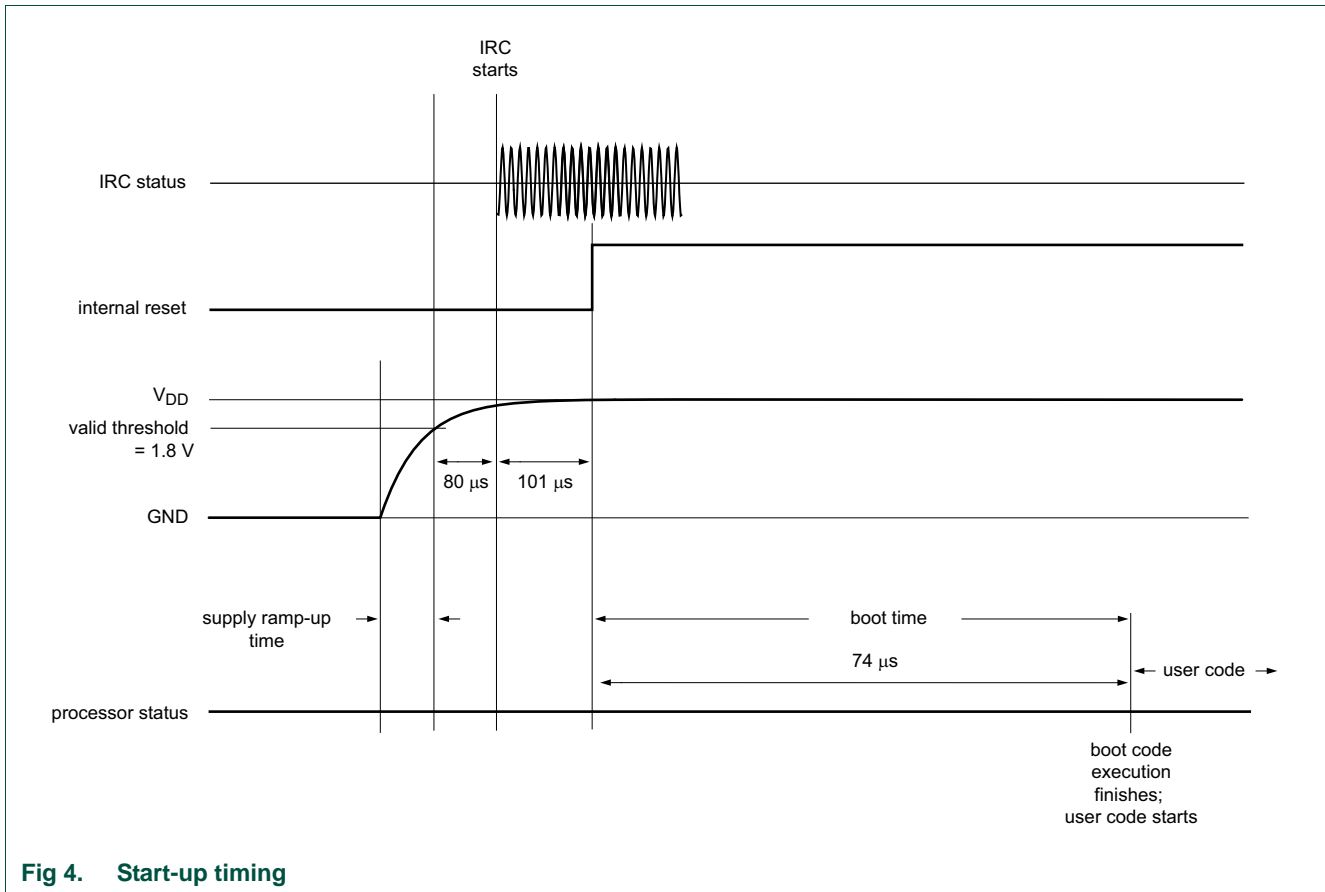


Fig 4. Start-up timing

4.6.3 Brown-out detection

The part includes up to four levels for monitoring the voltage on the V_{DD} pin. If this voltage falls below one of the selected levels, the BOD asserts an interrupt signal to the NVIC or issues a reset, depending on the value of the BODRSTENA bit in the BOD control register (Table 99).

The interrupt signal can be enabled for interrupt in the Interrupt Enable Register in the NVIC (see Table 4) in order to cause a CPU interrupt; if not, software can monitor the signal by reading a dedicated status register.

If the BOD interrupt is enabled in the STARTER0 register and in the NVIC, the BOD interrupt can wake up the chip from reduced power modes, not including deep power-down.

If the BOD reset is enabled, the forced BOD reset can wake up the chip from reduced power modes, not including deep power-down.

4.6.4 PLL functional description

The PLL is typically used to create a frequency that is higher than other on-chip clock sources, and used to operate the PCU and/or other on-chip functions. It may also be used to obtain a specific clock that is otherwise not available. For example, a clock with a frequency of any integer MHz (e.g. the 12 MHz IRC) can be divided down to 1 MHz, then multiplied up to any other integer MHz (e.g. 13, 14, 15, etc.).

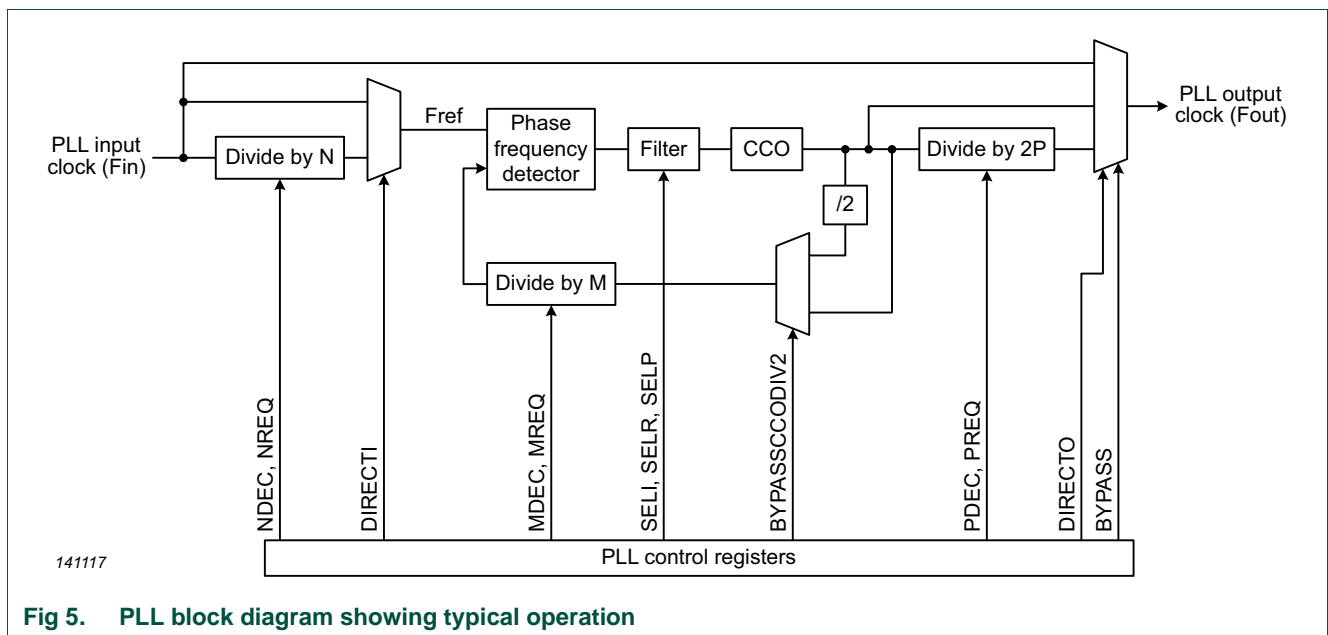


Fig 5. PLL block diagram showing typical operation

4.6.4.1 PLL Features

- Input frequency: Limited to on-chip sources, including the 32 kHz RTC clock and 12 MHz IRC, or up to 24 MHz from the CLKIN pin.
- CCO frequency: 75 MHz to 150 MHz.

- Output clock range: 1.2 MHz to 150 MHz. Note that the upper frequency limit of the PLL exceeds the upper frequency limit of this device.
- Programmable dividers:
 - Pre-divider. Divide by N, where $N = 1$ to 256
 - Feedback-divider. Divide by M or $2 \times M$ (where $M = 1$ to 32768)
 - Post-divider. Divide by 1 or $2 \times P$, where $P = 1$ to 32
- Lock detector.
- Power-down mode.
- Fractional divider mode.
- Spread Spectrum mode.

4.6.4.2 PLL description

A number of sources may be used as an input to the PLL, see [Figure 3](#). The PLL input, in the range of 32 kHz to 24 MHz, may initially be divided down by a value "N", which may be in the range of 1 to 256. This input division provides a greater number of possibilities in providing a wide range of output frequencies from the same input frequency.

Following the PLL input divider is the PLL multiplier. The multiplier can multiply the input divider output through the use of a Current Controlled Oscillator (CCO) by a value "M", in the range of 1 through 32,768. The resulting frequency must be in the range of 75 MHz to 150 MHz. The multiplier works by dividing the CCO output by the value of M, then using a phase-frequency detector to compare the divided CCO output to the multiplier input. The error value is filtered and used to adjust the CCO frequency.

There are additional dividers at the PLL output to bring the frequency down to what is needed for the CPU, USB, and other peripherals. The PLL output dividers are described in the Clock Dividers section following the PLL description. A block diagram of the PLL is shown in [Figure 5](#).

All of the dividers use an encoded value, not the binary divide value. The `set_pll` API (see [Section 31.4.1](#)) adjusts the value for the main feedback divider (the M divider), but does not accept pre- and post-divider values. See [Section 4.6.4.3](#) and [Section 4.6.4.5](#) for information on how to obtain divider values.

4.6.4.2.1 Lock detector

The lock detector measures the phase difference between the rising edges of the input and feedback clocks. Only when this difference is smaller than the so called "lock criterion" for more than eight consecutive input clock periods, the lock output switches from low to high. A single too large phase difference immediately resets the counter and causes the lock signal to drop (if it was high). Requiring eight phase measurements in a row to be below a certain figure ensures that the lock detector will not indicate lock until both the phase and frequency of the input and feedback clocks are very well aligned. This effectively prevents false lock indications, and thus ensures a glitch free lock signal.

The PLL lock indicator is not dependable when F_{ref} is below 100 kHz or above 20 MHz.

In fractional mode and spread spectrum mode, the PLL will generally not lock, software should use a time interval to insure the PLL will be stable. See [Section 4.6.4.5.1](#).

4.6.4.2.2 Power-down

To reduce the power consumption when the PLL clock is not needed, a PLL Power-down mode has been incorporated. This mode is enabled by setting the SYSPLL_PD bit to one in the power configuration register PDRUNCFG (Section 4.5.38). In this mode, the internal current reference will be turned off, the oscillator and the phase-frequency detector will be stopped and the dividers will enter a reset state. While in PLL Power-down mode, the lock output will be low to indicate that the PLL is not in lock. When the PLL Power-down mode is terminated by setting the SYSPLL_PD bit to zero, the PLL will resume its normal operation and will make the lock signal high once it has regained lock on the input clock.

4.6.4.3 Operating modes

The PLL includes several main operating modes, and a power-down mode. These are summarized in Table 100 and detailed in the following sections.

Table 100. PLL operating mode summary

| Mode | PDEN_SYS_PLL bit in PDRUNCFG | Bits in SYSPLLCTRL: | | | SEL_EXT bit in SYSYPLLSSCTRL0 | PD bit in SYSYPLLSSCTRL1 |
|--------------------|------------------------------|---------------------|----------|---------|-------------------------------|--------------------------|
| | | BYPASS | UPLIMOFF | BANDSEL | | |
| Normal | 0 | 0 | 0 | 1 | 1 | 1 |
| Fractional divider | 0 | 0 | 1 | 0 | 0 | 0 |
| Spread spectrum | 0 | 0 | 1 | 0 | 0 | 0 |
| Power-down | 1 | x [1] | x | x | x | 1 |

[1] Use 1 if the PLL output is used even though the PLL is not altering the frequency.

4.6.4.3.1 Normal modes

Typical operation of the PLL includes an optional pre-divide of the PLL input, followed by a frequency multiplication, and finally an optional post-divide to produce the PLL output.

Notations used in the frequency equations:

- Fin = the input to the PLL.
- Fout = the output of the PLL.
- Fref = the PLL reference frequency, the input to the phase frequency detector.
- N = optional pre-divider value.
- M = feedback divider value, which represents the multiplier for the PLL. Note that an additional divide-by-2 may optionally be included in the divider path.
- P = optional post-divider value. An additional divide-by-2 is included in the post-divider path.

A block diagram of the PLL as used in normal modes is shown in Figure 3.

In all variations of normal mode, the following requirements must be met:

- 75 MHz ≤ Fcco ≤ 150 MHz
- 4 kHz ≤ Fin / N ≤ 24 MHz

Normal mode with optional pre-divide

In the equations, use $N = 1$ when the pre-divider is not used:

When the extra divide by 2 **is** in the feedback divider path ($BYPASSCCODIV2 = 0$):

$$F_{out} = F_{cco} = 2 \times M \times F_{in} / N$$

When the extra divide by 2 **is not** in the feedback divider path ($BYPASSCCODIV2 = 1$):

$$F_{out} = F_{cco} = M \times F_{in} / N$$

Normal mode with post-divide and optional pre-divide

In the equations, use $N = 1$ when the pre-divider is not used:

When the extra divide by 2 **is** in the feedback divider path ($BYPASSCCODIV2 = 0$). Use $N = 1$ when the pre-divider is not used:

$$F_{out} = F_{cco} / (2 \times P) = M \times F_{in} / (N \times P)$$

When the extra divide by 2 **is not** in the feedback divider path ($BYPASSCCODIV2 = 1$):

$$F_{out} = F_{cco} / (2 \times P) = M \times F_{in} / (N \times 2 \times P)$$

4.6.4.3.2 Fractional divider mode

The PLL includes an fractional divide mode. The fractional mode uses an integer divide value and that value plus 1 in a ratio determined by the fractional part of the divide value in order to obtain an average rate that is a fractional multiple of the PLL reference frequency. The `SEL_EXT` bit in the `SYSPLLSSCTRL0` register determines whether the fractional divider is used (`SEL_EXT = 0`) or bypassed (`SEL_EXT = 1`). In the first case, the `MD` value from the `SYSPLLSSCTRL1` register is used to generate the feedback divider values. In the latter case, the `MDEC` value from the `SYSPLLSSCTRL0` register is used directly to control the feedback divider.

When the fractional divider is active, the spread spectrum controller block generates divider values M and $M+1$ in the correct proportion so that the average CCO frequency is represented by the specified fraction. the average CCO frequency is:

$$F_{cco} = 2 * (MD[18:11] + MD[10:0] * 2^{-11}) * F_{ref}$$

The overall effect of the PLL otherwise the same as normal modes. A block diagram of the PLL as used in fractional mode is shown in [Figure 6](#).

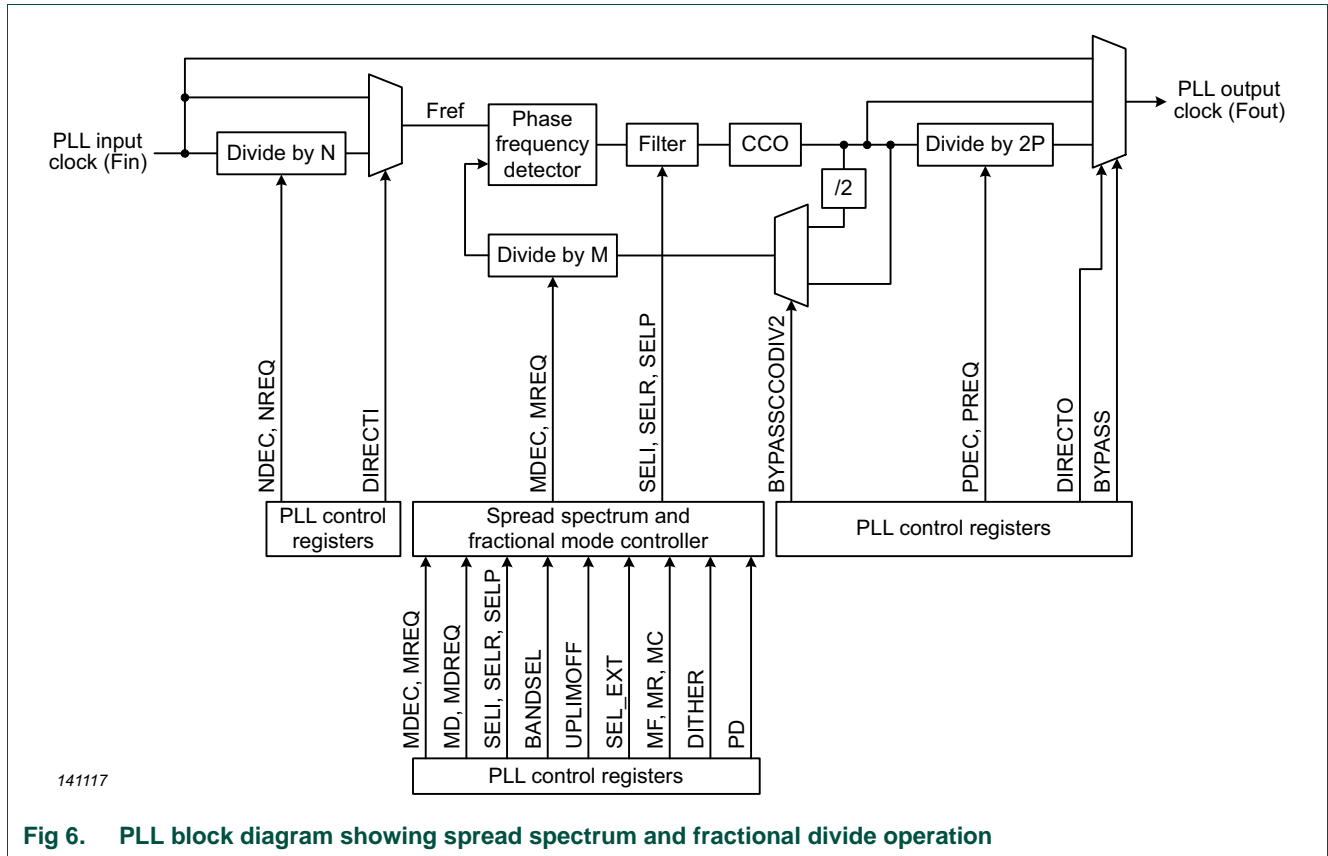


Fig 6. PLL block diagram showing spread spectrum and fractional divide operation

4.6.4.3.3 Spread Spectrum mode

The spread spectrum mode allows the PLL to change frequency automatically in a programmable manner.

A block diagram of the PLL as used in fractional mode is shown in [Figure 6](#).

4.6.4.3.4 Power-down mode

If the PLL is not used, or if there are cases where it is turned off in a running application, power can be saved by putting the PLL in power-down mode. Before this is done, the CPU and any peripherals that are not meant to be stopped as well must be running from some other clock source.

4.6.4.4 PLL Related registers

The PLL is controlled by registers described elsewhere in this chapter, summarized below.

Table 101. System PLL status register (SYSPLLSTAT, address 0x4000 01B4) bit description

| Register | Description | Reference |
|------------|---------------|----------------------------------|
| SYSPLLCTRL | PLL control | Section 4.5.37.1 |
| SYSPLLSTAT | PLL status | Section 4.5.37.2 |
| SYSPLLNDEC | PLL N divider | Section 4.5.37.3 |

Table 101. System PLL status register (SYSPLLSTAT, address 0x4000 01B4) bit description

| Register | Description | Reference |
|--------------|-------------------------------|------------------------------------|
| SYSPLLPDEC | PLL P divider | Section 4.5.37.4 |
| SYSPLLSCTRL0 | PLL spread spectrum control 0 | Section 4.5.37.5.1 |
| SYSPLLSCTRL1 | PLL spread spectrum control 1 | Section 4.5.37.5.2 |

4.6.4.5 PLL usage

As previously noted, the PLL divider settings used in the PLL registers are not simple binary values, they are encoded as shown in the PLL register descriptions. The divider values and their encoding can be found by calculation using the information in this document. For simple PLL usage with no pre- or post-divide, the `set_pll` API can be used (see [Section 31.4.1](#)). Also, a PLL setting calculator can be found on the NXP website. The latter two possibilities are recommended in order to avoid PLL setup issues.

4.6.4.5.1 Procedure for determining PLL settings

In general, PLL configuration values may be found as follows:

1. Identify a desired PLL output frequency. This may depend on a specific interface frequency needed or be based on expected CPU performance requirements, and may be limited by system power availability.
2. Determine which clock source to use as the PLL input. This can be influenced by power or accuracy required, or by the potential to obtain the desired PLL output frequency.
3. Identify PLL settings to obtain the desired output from the selected input. The F_{cco} frequency must be either the actual desired output frequency, or the desired output frequency times $2 \times P$, where P is from 2 to 32. The F_{cco} frequency must also be a multiple of the PLL reference frequency, which is either the PLL input, or the PLL input divided by N , where N is from 2 to 256.
4. There may be several ways to obtain the same PLL output frequency. PLL power depends on F_{cco} (a lower frequency uses less power) and the divider used. Bypassing the input and/or output divider saves power.
5. Check that the selected settings meet all of the PLL requirements:
 - F_{in} is in the range of 32 kHz to 24 MHz.
 - F_{cco} is in the range of 75 MHz to 150 MHz.
 - F_{out} is in the range of 1.2 MHz to 150 MHz.
 - The pre-divider is either bypassed, or N is in the range of 2 to 256.
 - The post-divider is either bypassed, or P is in the range of 2 to 32.
 - M is in the range of 3 to 32,768.

Also note that PLL startup time becomes longer as F_{ref} drops below 500 kHz. At 500 kHz and above, startup time is up to 500 microseconds. Below 500 kHz, startup time can be estimated as $200 / F_{ref}$, or up to 6.1 milliseconds for $F_{ref} = 32$ kHz. PLL accuracy and jitter is better with higher values of F_{ref} .

4.6.4.5.2 PLL setup sequence

The following sequence should be followed to initialize and connect the PLL:

1. Make sure that the PLL output is disconnected from any downstream functions. If the PLL was previously being used to clock the CPU, and the CPU Clock Divider is being used, it may be set to speed up operation while the PLL is disconnected.
2. Select a PLL input clock source. See [Section 4.5.21 “System PLL clock source select register”](#).
3. Set up the PLL dividers and mode settings. See [Section 4.5.37 “PLL registers”](#).
4. Wait for the PLL output to stabilize. The value of the PLI lock may not be stable when the PLL reference frequency (FREF, the frequency of REFCLK, which is equal to the PLL input frequency divided by the pre-divider value) is less than 100 kHz or greater than 20 MHz. In these cases, the PLL may be assumed to be stable after a start-up time has passed. This time is 500 μ s when Fref is 500 kHz or greater and 200 / Fref seconds when FREF is less than 500 kHz.
5. If the PLL will be used to clock the CPU, change the CPU Clock Divider setting for operation with the PLL, if needed. This must be done before connecting the PLL.
6. Connect the PLL to whichever downstream function is will be used with. The structure of the clock dividers may be seen on the right of [Figure 3 “Clock generation”](#).

4.6.5 Frequency measure function

The Frequency Measure circuit is based on two 14-bit counters, one clocked by the reference clock and one by the target clock. Synchronization between the clocks is performed at the start and end of each count sequence.

A measurement cycle is initiated by software setting a control/status bit in the FREQMECTRL register ([Table 61](#)). The software can then poll this same measurement-in-progress bit which will be cleared by hardware when the measurement operation is completed.

The measurement cycle terminates when the reference counter rolls-over. At that point the state of the target counter is loaded into a capture field in the FREQMEAS register, and the measure-in-progress bit is cleared. Software can read this capture value and apply to it a specific calculation which will return the precise frequency of the target clock in MHz.

See [Section 4.2.3 “Measure the frequency of a clock signal”](#), [Section 4.5.32 “Frequency measure function control register”](#), [Section 9.6.4 “Frequency measure function reference clock select register”](#), and [Section 9.6.5 “Frequency measure function target clock select register”](#) for more on this function.

4.6.5.1 Accuracy

The frequency measurement function can measure the frequency of any on-chip (or off-chip) clock (referred to as the target clock) to a high degree of accuracy using another on-chip clock of known frequency as a reference.

The following constraints apply:

- The frequency of the reference clock must be (somewhat) greater than the frequency of the target clock.
- The system clock used to access the frequency measure function register must also be greater than the frequency of the target clock.

The frequency measurement function circuit is able to measure the target frequency with an error of less than 0.1%, provided the reference frequency is precisely known.

Uncertainty in the reference clock (for example the +/- 1% accuracy of the IRC) will add to the measurement error of the target clock. In general, though, this additional error is less than the uncertainty of the reference clock.

There can also be a modest loss of accuracy if the reference frequency exceeds the target frequency by a very large margin (25x or more). Accuracy is not a simple function of the magnitude of the frequency difference, however. Nearly identical frequency combinations, still with a spread of about 43x, result in errors of less than 0.05%.

If the target and reference clocks are different by more than a factor of approximately 500, then the accuracy decreases to +/- 4%.

5.1 Introduction

This chapter provides an overview of power related information about LPC5410x devices. These devices include a variety of adjustable regulators, power switches, and clock switches to allow fine tuning power usage to match requirements at different performance levels and reduced power modes. All devices include an on-chip API in the boot ROM to adjust power consumption in reduced power modes, and provide entry to those modes. See [Chapter 31](#).

To turn analog components on or off in active and sleep modes, use the PDRUNCFG register (see [Table 72](#)). In deep-sleep and power-down modes, the power profile API controls which analog peripherals remain powered up (see [Section 31.4.3 “power_mode_configure”](#)). There is no register implemented to turn analog peripherals on or off for deep-sleep mode or power-down mode.

5.2 General description

Power to the part is supplied via two power domains. The main power domain is powered by VDD and supplies power to the core, peripheral, memories, inputs and outputs via an on-chip regulator.

A second, always-on power domain is also powered by Vdd, and includes the RTC and wakeup timer. This domain always has power as long as sufficient voltage is supplied to Vdd.

Power use is controlled by settings in register within the SYSCON block, regulator settings controlled via a Power API, and the operating mode of a CPU. The ROM based power configuration API configures the part for each reduced power mode. The following modes are supported in order from highest to lowest power consumption:

1. Active mode: The part is in active mode after a Power-On Reset (POR) and when it is fully powered and operational after booting.

2. Sleep mode:

The sleep mode affects the relevant CPU only. The clock to the core is shut off. Peripherals and memories are active and operational.

3. Deep-sleep and power-down modes:

The Deep-sleep and power-down modes affect the entire system. In both modes, the clock to all CPUs is shut down and the peripherals receive no internal clocks. All SRAM and registers maintain their internal states. Entry to these modes can only be accomplished by the master CPU in an LPC54102 device.

Through the power profiles API, selected peripherals can be left running for safe operation of the part (WWDT and BOD) or for monitoring analog inputs (comparators and internal voltage reference and temperature sensor via one of the comparators).

The differences between Deep-sleep mode and Power-down modes are the following:

- a. In Deep-sleep mode, the flash is in stand-by mode to minimize wake-up time, and the IRC is turned off to save power.
 - b. In Power-down mode, the flash is also powered down to conserve power at the expense of a somewhat longer wake-up time.
4. Deep power-down mode:
- For maximal power savings, the entire system (CPUs and all peripherals) is shut down except for the PMU and the RTC. On wake-up, the part reboots. Entry to Deep power-down mode can only be accomplished by the master CPU in an LPC54102 device.

Table 102. Peripheral configuration in reduced power modes

| Peripheral | Sleep mode | Deep-sleep mode | Power-down mode | Deep power-down mode |
|---------------------------|-----------------------|--|--|-----------------------|
| IRC | Software configurable | Off | Off | Off |
| Flash | Software configurable | On | Off | Off |
| BOD | Software configurable | Software configurable | Software configurable | Off |
| PLL | Software configurable | Off | Off | Off |
| Watchdog osc and WWDT | Software configurable | Software configurable | Software configurable | Off |
| USART | Software configurable | Off; but can create a wake-up interrupt in synchronous slave mode or 32 kHz clock mode | Off; but can create a wake-up interrupt in synchronous slave mode or 32 kHz clock mode | Off |
| SPI | Software configurable | Off; but can create a wake-up interrupt in slave mode | Off; but can create a wake-up interrupt in slave mode | Off |
| I2C | Software configurable | Off; but can create a wake-up interrupt in slave mode | Off; but can create a wake-up interrupt in slave mode | Off |
| Other digital peripherals | Software configurable | Off | Off | Off |
| Analog peripherals | Software configurable | Software configurable | Software configurable | Off |
| RTC oscillator | Software configurable | Software configurable | Software configurable | Software configurable |

5.2.1 Wake-up process

The part always wakes up to the active mode. To wake up from the reduced power modes, you must configure the wake-up source. Each reduced power mode supports its own wake-up sources and needs to be configured accordingly as shown in [Table 103](#).

Table 103. Wake-up sources for reduced power modes

| Power mode | Wake-up source | Conditions |
|---------------------------|---|--|
| Sleep | Any interrupt | Enable interrupt in NVIC. |
| Deep-sleep and Power-down | Pin interrupts | Enable pin interrupts in NVIC and STARTER0 and/or STARTER1 registers. |
| | BOD interrupt | <ul style="list-style-type: none"> Enable interrupt in NVIC and STARTER0 registers. Enable interrupt in BODCTRL register. Configure the BOD to keep running in this mode with the power API. |
| | BOD reset | Enable reset in BODCTRL register. |
| | Watchdog interrupt | <ul style="list-style-type: none"> Enable the watchdog oscillator in the PDRUNCFG register. Enable the watchdog interrupt in NVIC and STARTER0 registers. Enable the watchdog in the WWDT MOD register and feed. Enable interrupt in WWDT MOD register. Configure the WDOOSC to keep running in this mode with the power API. |
| | Watchdog reset | <ul style="list-style-type: none"> Enable the watchdog oscillator in the PDRUNCFG register. Enable the watchdog and watchdog reset in the WWDT MOD register and feed. |
| | Reset pin | Always available. |
| | RTC 1 Hz alarm timer | <ul style="list-style-type: none"> Enable the RTC 1 Hz oscillator in the RTCOSCCTRL register. Enable the RTC bus clock in the AHBCLKCTRL0 register. Start RTC alarm timer by writing a time-out value to the RTC COUNT register. Enable the RTCALARM interrupt in the STARTER0 register. |
| | RTC 1 kHz timer time-out and alarm | <ul style="list-style-type: none"> Enable the RTC 1 Hz oscillator and the RTC 1 kHz oscillator in the RTC CTRL register. Start RTC 1 kHz timer by writing a time-out value to the RTC WAKE register. Enable the RTCWAKE interrupt in the STARTER0 register. |
| | Micro-tick timer (specifically intended ultra-low power wake-up from Power-down mode) | <ul style="list-style-type: none"> Enable the watchdog oscillator in the PDRUNCFG register. Enable the Micro-tick timer clock by writing to the AHBCLKCTRL1 register. Start the Micro-tick timer by writing UTICK CTRL register. Enable the Micro-tick timer interrupt in the STARTER0 register. |
| | I2C interrupt | Interrupt from I2C in slave mode. See Chapter 24 “LPC5410x I2C-bus interfaces (I2C0/1/2)” . |
| | SPI interrupt | Interrupt from SPI in slave mode. See Chapter 23 “LPC5410x Serial Peripheral Interfaces (SPI0/1)” . |
| USART interrupt | Interrupt from USART in slave or 32 kHz mode. See Chapter 22 “LPC5410x USARTs (USART0/1/2/3)” . | |
| Deep power-down | RTC 1 Hz alarm timer | <ul style="list-style-type: none"> Enable the RTC 1 Hz oscillator in the RTC CTRL register. Start RTC alarm timer by writing a time-out value to the RTC COUNT register. |
| | RTC 1 kHz timer time-out and alarm | <ul style="list-style-type: none"> Enable the RTC 1 Hz oscillator and the RTC 1 kHz oscillator in the RTCOSCCTRL register. Enable the RTC bus clock in the AHBCLKCTRL0 register. Start RTC 1 kHz timer by writing a time-out value to the RTC WAKE register. |
| | Reset pin | Always available. |

5.3 Functional description

5.3.1 Power management

The LPC5410x support a variety of power control features. In Active mode, when the chip is running, power and clocks to selected peripherals can be optimized for power consumption. In addition, there are four special modes of processor power reduction with different peripherals running: Sleep mode, Deep-sleep mode, Power-down mode, and Deep power-down mode, activated by the power mode configure API (see [Section 31.4.3 “power_mode_configure”](#)).

Remark: The Debug mode is not supported in Sleep, Deep-sleep, Power-down, or Deep power-down modes.

5.3.2 Active mode

In Active mode, the CPU, memories, and peripherals are clocked by the AHB/CPU clock.

The chip is in Active mode after reset and the default power configuration is determined by the reset values of the PDRUNCFG, AHBCLKCTRL0, and AHBCLKCTRL1 registers. The power configuration can be changed during run time.

5.3.2.1 Power configuration in Active mode

Power consumption in Active mode is determined by the following configuration choices:

- The AHBCLKCTRL registers control which memories and peripherals are running ([Section 4.5.22 “AHB Clock Control register 0”](#) and [Section 4.5.23 “AHB Clock Control register 1”](#)). Generally speaking, in order to save power, functions that are not needed by the application should be turned off. If specific times are known when certain functions will not be needed, they can be turned off temporarily and turned back on when they will be needed.
- The power to various analog blocks (RAMs, PLL, oscillators, the BOD circuit, and the flash block) can be controlled individually through the PDRUNCFG register ([Table 72](#)). As with clock controls, these blocks should generally be tuned off if not needed by the application. If turned off, time will be needed before these blocks can be used again after being turned on.
- The clock source for the system clock can be selected from the IRC (default), the system oscillator, the 32 kHz oscillator, or the watchdog oscillator (see [Figure 3](#) and related registers).
- The system clock frequency can be selected (see [Section 4.6.4 “PLL functional description”](#) and other clocking related sections). You can find optimal settings for setting the system PLL by using the set_pll routine in the power API ([Section 31.4.1 “set_pll”](#)). Generally speaking, everything uses less power at lower frequencies, so running the CPU and other device features at a frequency sufficient for the application (plus some margin) will save power. If the PLL is not needed, it should be turned off to save power. Also, running the PLL at a lower CCO frequency saves power.
- Several peripherals use individual peripheral clocks with their own clock dividers. The peripheral clocks can be shut down through the corresponding clock divider registers if the root clock is still needed for another function.

- The power API provides an easy way to optimize power consumption depending on CPU load and performance requirements. See [Chapter 31 “LPC5410x Power profiles/Power control API”](#).

5.3.3 Sleep mode

In Sleep mode, the system clock to the CPU is stopped and execution of instructions is suspended until either a reset or an interrupt occurs.

Peripheral functions, if selected to be clocked in the AHBCLKCTRL registers, continue operation during Sleep mode and may generate interrupts to cause the processor to resume execution. Sleep mode eliminates dynamic power used by the processor itself, memory systems and related controllers, and internal buses. The processor state and registers, peripheral registers, and internal SRAM values are maintained, and the logic levels of the pins remain static.

As in active mode, the power API provides an easy way to optimize power consumption depending on CPU load and performance requirements in sleep mode. See [Section 31.4.3 “power_mode_configure”](#).

5.3.3.1 Power configuration in Sleep mode

Power consumption in Sleep mode is configured by the same settings as in Active mode:

- The clock remains running.
- The system clock frequency remains the same as in Active mode, but the processor is not clocked.
- Analog and digital peripherals are powered and selected as in Active mode through the PDRUNCFG, AHBCLKCTRL0, AHBCLKCTRL1 registers.

5.3.3.2 Programming Sleep mode

The following steps must be performed to enter Sleep mode:

1. In the NVIC, enable all interrupts that are needed to wake up the part.
2. Call power API: `pPWRD->power_mode_configure(SLEEP, peripheral);`
Remark: The `peripheral` parameter is don't care.
3. Execute the Wait-For-Interrupt (WFI) instruction.

5.3.3.3 Wake-up from Sleep mode

Sleep mode is exited automatically when an interrupt enabled by the NVIC arrives at the processor or a reset occurs. After wake-up caused by an interrupt, the device returns to its original power configuration defined by the contents of the PDRUNCFG and the AHBCLKCTRL registers. If a reset occurs, the microcontroller enters the default configuration in Active mode.

5.3.4 Deep-sleep mode

In Deep-sleep mode, the system clock to the processor is disabled as in Sleep mode. All analog blocks are powered down by default but can be selected to keep running through the power API if needed as wake-up sources. The main clock, and therefore all peripheral clocks, are disabled. The IRC is disabled. The flash is in stand-by mode.

Deep-sleep mode eliminates all power used by analog peripherals and all dynamic power used by the processor itself, memory systems and related controllers, and internal buses. The processor state and registers, peripheral registers, and internal SRAM values are maintained, and the logic levels of the pins remain static.

5.3.4.1 Power configuration in Deep-sleep mode

Power consumption in Deep-sleep mode is determined primarily by which analog wake-up sources remain enabled. Serial peripherals and pin interrupts configured to wake up the contribute to the power consumption only to the extent that they are clocked by external sources. All wake-up events (other than reset) must be enabled in the STARTER registers and in the NVIC. In addition, any related analog block (e.g. the RTC oscillator or the watchdog oscillator) must be explicitly enabled through the power API function `power_mode_configure()` for wake-up. See [Table 103](#).

5.3.4.2 Programming Deep-sleep mode

The following steps must be performed to enter Deep-sleep mode:

1. Select wake-up sources and enable all selected wake-up events in the STARTER registers ([Table 75](#) and [Table 76](#)) and in the NVIC.
2. Select the IRC as the main clock. See [Table 45](#) and [Table 46](#).
3. Call the power API with the parameter `peripheral` set to enable the analog peripherals the serve as wake-up sources (see [Table 458 “power_mode_configure routine”](#)):

```
pPWRD->power_mode_configure(DEEP_SLEEP, peripheral);
```
4. Execute the WFI instruction.

5.3.4.3 Wake-up from Deep-sleep mode

The part can wake up from Deep-sleep mode in the following ways:

- Using a signal on one of the eight pin interrupts selected in [Section 9.6.1 “Pin interrupt select registers”](#). Each pin interrupt must also be enabled in the STARTER0 register ([Table 75](#)) and in the NVIC.
- Using an interrupt from a block such as the watchdog interrupt or RTC interrupt, when enabled during the reduced power mode via the power API. Also enable the wake-up sources in the STARTER registers ([Table 75](#) and [Table 76](#)) and the NVIC.
- Using a reset from the $\overline{\text{RESET}}$ pin, or the BOD or WWDT (if enabled in the power API).
- Using a wake-up signal from any of the serial peripherals that are operating in Deep-sleep mode. Also enable the wake-up sources in the STARTER registers ([Table 75](#) and [Table 76](#)) and the NVIC.
- GPIO group interrupt signal. The interrupt must also be enabled in the STARTER1 register ([Table 76](#)) and in the NVIC.
- RTC alarm signal or wake-up signal. See [Chapter 17](#). Interrupts must also be enabled in the STARTER1 register ([Table 76](#)) and in the NVIC.

5.3.5 Power-down mode

In Power-down mode, the system clock to the processor is disabled as in Sleep mode. All analog blocks are powered down by default but can be selected to keep running if needed for waking up the part. The main clock and all peripheral clocks are disabled except for the clock to the watchdog timer if the watchdog oscillator is selected. The flash is powered down, decreasing power consumption compared to Deep-sleep mode.

Power-down mode can eliminate all power used by analog peripherals and all dynamic power used by the processor itself, memory systems and related controllers, and internal buses. The processor state and registers, peripheral registers, and internal SRAM values are maintained, and the logic levels of the pins remain static. Wake-up times are longer compared to the Deep-sleep mode.

5.3.5.1 Power configuration in Power-down mode

Power consumption in power-down mode is determined by which analog wake-up sources are enabled. Serial peripherals and pin interrupts configured to wake up the part do not contribute significantly to the power consumption. All wake-up events (other than reset) must be enabled in the STARTER registers and in the NVIC. In addition, any related analog block (e.g. the RTC oscillator or the watchdog oscillator) must be explicitly enabled through the power API function `power_mode_configure()` for wake-up. See [Table 103](#).

5.3.5.2 Programming Power-down mode

The following steps must be performed to enter Power-down mode:

1. Select wake-up sources and enable all related wake-up events in the STARTER registers ([Table 75](#) and [Table 76](#)) and in the NVIC.
2. Select the IRC as the main clock. See [Table 45](#) and [Table 46](#).
3. Call the power API with the peripheral parameter set to enable the analog wake-up sources: `pPWRD->power_mode_configure(POWER_DOWN, peripheral);`
4. Execute the WFI instruction.

5.3.5.3 Wake-up from Power-down mode

The part can wake up from Power-down mode in the following ways:

- Using a signal on one of the eight pin interrupts selected in [Section 9.6.1 “Pin interrupt select registers”](#). Each pin interrupt must also be enabled in the STARTER0 register ([Table 75](#)) and in the NVIC.
- Using an interrupt from a block such as the watchdog timer, RTC, or Micro-tick timer, when enabled during the reduced power mode via the power API. Also enable the wake-up sources in the STARTER registers ([Table 75](#) and [Table 76](#)) and the NVIC.
- Using a reset from the $\overline{\text{RESET}}$ pin, or the BOD or WWDT (if enabled in the power API).
- Using a wake-up signal from any of the serial peripherals. Also enable the wake-up sources in the STARTER registers ([Table 75](#) and [Table 76](#)) and the NVIC.
- GPIO group interrupt signal. Interrupt must also be enabled in the STARTER1 register ([Table 76](#)) and in the NVIC.
- RTC alarm signal or wake-up signal. See [Chapter 17](#). Interrupts must also be enabled in the STARTER1 register ([Table 76](#)) and in the NVIC.

5.3.6 Deep power-down mode

In Deep power-down mode, power and clocks are shut off to the entire chip with the exception of the RTC.

During Deep power-down mode, the contents of the SRAM and registers are not retained. All functional pins are tri-stated in Deep power-down mode.

5.3.6.1 Power configuration in Deep power-down mode

Deep power-down mode has no configuration options. All clocks, the core, and all peripherals are powered down. Only the RTC is powered, as long as power is supplied to the device.

5.3.6.2 Wakeup from Deep power-down mode:

Wakeup from Deep power-down can be accomplished via the reset pin or the RTC.

5.3.6.3 Programming Deep power-down mode using the RTC for wake-up:

The following steps must be performed to enter Deep power-down mode when using the RTC for waking up:

1. Set up the RTC high resolution timer. Write to the RTC VAL register. This starts the high res timer if enabled. Another option is to use the 1 Hz alarm timer.
2. Call the power API: `pPWRD->power_mode_configure(DEEP_POWER_DOWN, peripheral);`
Remark: The `peripheral` parameter is don't care.
3. Execute the WFI instruction.

5.3.6.4 Wake-up from Deep power-down mode using the RTC:

The part goes through the entire reset process when the RTC times out:

- The PMU will turn on the on-chip voltage regulator. When the core voltage reaches the power-on-reset (POR) trip point, a system reset will be triggered and the chip boots.
- All registers will be in their reset state.

6.1 Features

- 64 KB on-chip boot ROM
- Contains the boot loader with In-System Programming (ISP) facility and the following APIs:
 - In-Application Programming (IAP) of flash memory
 - Power profiles for optimizing power consumption and system performance and for controlling low power modes

6.2 Pin description

The parts support ISP via USART0. The ISP mode is determined by the state of the P0_31 pin at boot time:

Table 104. ISP modes

| Boot mode | P0_31 | Description |
|-----------|-------|---|
| No ISP | HIGH | ISP bypassed. Part attempts to boot from flash. |
| USART0 | LOW | Part enters ISP via USART0. |

6.3 General description

The boot loader controls initial operation after reset and also provides the means to program the flash memory. This could be initial programming of a blank device, erasure and re-programming of a previously programmed device, or programming of the flash memory by the application program in a running system.

The boot loader code is executed every time the part is powered on or reset (see [Figure 7](#)). The loader can execute the ISP command handler or the user application code.

The boot loader version can be read by ISP/IAP calls (see [Table 477](#) or [Table 489](#)).

Assuming that power supply pins are at their nominal levels when the rising edge on RESET pin is generated, it may take up to 3 ms before the boot pins are sampled and the decision whether to continue with user code or ISP handler is made. If the boot pins are sampled LOW and the watchdog overflow flag is set, the external hardware request to start the ISP command handler is ignored. If there is no request for the ISP command handler execution, a search is made for a valid user program. If a valid user program is found then the execution control is transferred to it. If a valid user program is not found, the auto-baud routine is invoked.

See [Chapter 32 “LPC5410x Flash API”](#) for ISP and IAP commands.

6.3.1 Boot process flowchart

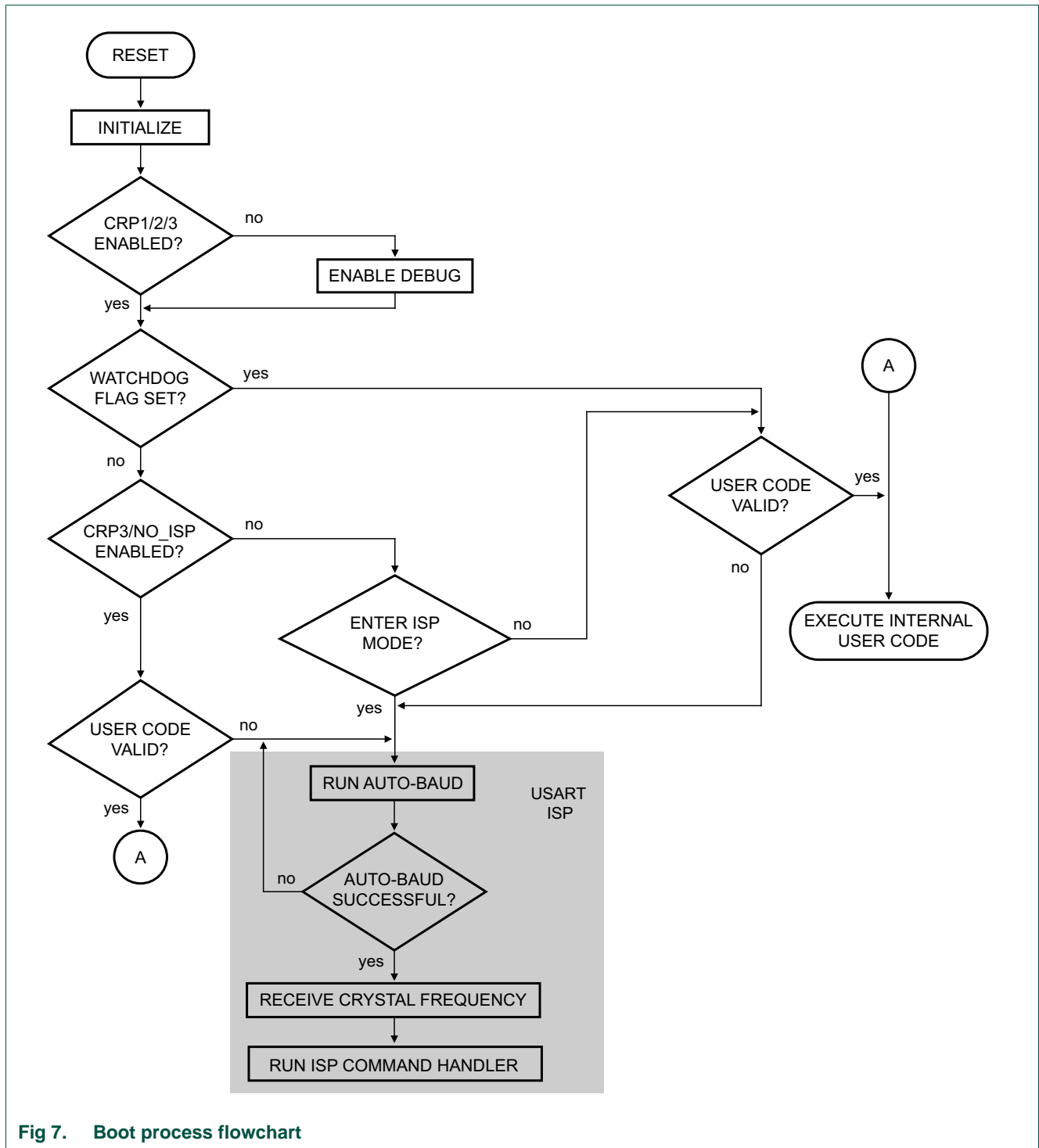


Fig 7. Boot process flowchart

7.1 Pin description

Table 105. Pin description

| Symbol | WLCSP49 | LQFP64 | Reset state [1] | Type [6] | Description |
|--------|---------|--------|---------------------------------|--------------------------|--|
| PIO0_0 | A6 | 31 | [2] | PU | I/O PIO0_0 — General-purpose digital input/output pin. Remark: In ISP mode, this pin is the UART0 RXD function. |
| | | | | | I U0_RXD — Receiver input for USART0. |
| | | | | | I/O SPI0_SSEL0 — Slave Select 0 for SPI0. |
| | | | | | I CT32B0_CAP0 — 32-bit CT32B0 capture input 0. |
| | | | | | I R — Reserved. |
| | | | | | O SCT0_OUT3 — SCT0 output 3. PWM output 3. |
| PIO0_1 | B6 | 32 | [2] | PU | I/O PIO0_1 — General-purpose digital input/output pin. Remark: In ISP mode, this pin is the UART0 TXD function. |
| | | | | | O U0_TXD — Transmitter output for USART0. |
| | | | | | I/O SPI0_SSEL1 — Slave Select 1 for SPI0. |
| | | | | | I CT32B0_CAP1 — 32-bit CT32B0 capture input 1. |
| | | | | | I R — Reserved. |
| | | | | | O SCT0_OUT1 — SCT0 output 1. PWM output 1. |
| PIO0_2 | - | 36 | [2] | PU | I/O PIO0_2 — General-purpose digital input/output pin. |
| | | | | | I U0_CTS — Clear To Send input for USART0. |
| | | | | | I R — Reserved. |
| | | | | | I CT32B2_CAP1 — 32-bit CT32B2 capture input 1. |
| | | | | | I R — Reserved. |
| PIO0_3 | - | 37 | [2] | PU | I/O PIO0_3 — General-purpose digital input/output pin. |
| | | | | | O U0_RTS — Request To Send output for USART0. |
| | | | | | I R — Reserved. |
| | | | | | O CT32B1_MAT3 — 32-bit CT32B1 match output 3. |
| | | | | | I R — Reserved. |
| PIO0_4 | C7 | 38 | [2] | PU | I/O PIO0_4 — General-purpose digital input/output pin. |
| | | | | | I/O U0_SCLK — USART0 clock in synchronous USART mode. |
| | | | | | I/O SPI0_SSEL2 — Slave Select 2 for SPI0. |
| | | | | | I CT32B0_CAP2 — 32-bit CT32B0 capture input 2. |
| | | | | | I R — Reserved. |

Table 105. Pin description ...continued

| Symbol | WLCSP49 | LQFP64 | Reset state [1] | Type [6] | Description |
|---------|---------|--------|---------------------------------|--------------------------|--|
| PIO0_5 | C6 | 39 | [2] | PU | I/O PIO0_5 — General-purpose digital input/output pin. |
| | | | | | I U1_RXD — Receiver input for USART1. |
| | | | | | O SCT0_OUT6 — SCT0 output 6. PWM output 6. |
| | | | | | O CT32B0_MAT0 — 32-bit CT32B0 match output 0. |
| | | | | | I R — Reserved. |
| PIO0_6 | D7 | 40 | [2] | PU | I/O PIO0_6 — General-purpose digital input/output pin. |
| | | | | | O U1_TXD — Transmitter output for USART1. |
| | | | | | I R — Reserved. |
| | | | | | O CT32B0_MAT1 — 32-bit CT32B0 match output 1. |
| | | | | | I R — Reserved. |
| PIO0_7 | D6 | 41 | [2] | PU | I/O PIO0_7 — General-purpose digital input/output pin. |
| | | | | | I/O U1_SCLK — USART1 clock in synchronous USART mode. |
| | | | | | O SCT0_OUT0 — SCT0 output 0. PWM output 0. |
| | | | | | O CT32B0_MAT2 — 32-bit CT32B0 match output 2. |
| | | | | | I R — Reserved. |
| | | | | | I CT32B0_CAP2 — 32-bit CT32B0 capture input 2. |
| PIO0_8 | D5 | 43 | [2] | PU | I/O PIO0_8 — General-purpose digital input/output pin. |
| | | | | | I U2_RXD — Receiver input for USART2. |
| | | | | | O SCT0_OUT1 — SCT0 output 1. PWM output 1. |
| | | | | | O CT32B0_MAT3 — 32-bit CT32B0 match output 3. |
| | | | | | I R — Reserved. |
| PIO0_9 | E7 | 44 | [2] | PU | I/O PIO0_9 — General-purpose digital input/output pin. |
| | | | | | O U2_TXD — Transmitter output for USART2. |
| | | | | | O SCT0_OUT2 — SCT0 output 2. PWM output 2. |
| | | | | | I CT32B3_CAP0 — 32-bit CT32B3 capture input 0. |
| | | | | | I R — Reserved. |
| | | | | | I/O SPI0_SSEL0 — Slave Select 0 for SPI0. |
| PIO0_10 | E6 | 45 | [2] | PU | I/O PIO0_10 — General-purpose digital input/output pin. |
| | | | | | I/O U2_SCLK — USART2 clock in synchronous USART mode. |
| | | | | | O SCT0_OUT3 — SCT0 output 3. PWM output 3. |
| | | | | | O CT32B3_MAT0 — 32-bit CT32B3 match output 0. |
| | | | | | I R — Reserved. |
| PIO0_11 | E5 | 46 | [2] | PU | I/O PIO0_11 — General-purpose digital input/output pin. |
| | | | | | I/O SPI0_SCK — Serial clock for SPI0. |
| | | | | | I U1_RXD — Receiver input for USART1. |
| | | | | | O CT32B2_MAT1 — 32-bit CT32B2 match output 1. |
| | | | | | I R — Reserved. |

Table 105. Pin description ...continued

| Symbol | WLCSP49 | LQFP64 | | Reset state [1] | Type [6] | Description |
|-------------------|---------|--------|-----|-----------------|----------|---|
| PIO0_12 | F7 | 47 | [2] | PU | I/O | PIO0_12 — General-purpose digital input/output pin. |
| | | | | | I/O | SPI0_MOSI — Master Out Slave in for SPI0. |
| | | | | | O | U1_TXD — Transmitter output for USART1. |
| | | | | | O | CT32B2_MAT3 — 32-bit CT32B2 match output 3. |
| | | | | | I | R — Reserved. |
| PIO0_13 | G7 | 48 | [2] | PU | I/O | PIO0_13 — General-purpose digital input/output pin. |
| | | | | | I/O | SPI0_MISO — Master In Slave Out for SPI0. |
| | | | | | O | SCT0_OUT4 — SCT0 output 4. PWM output 4. |
| | | | | | O | CT32B2_MAT0 — 32-bit CT32B2 match output 0. |
| | | | | | I | R — Reserved. |
| PIO0_14/ TCK | F6 | 49 | [2] | PU | I/O | PIO0_14 — General-purpose digital input/output pin. In boundary scan mode: TCK (Test Clock). |
| | | | | | I/O | SPI0_SSEL0 — Slave Select 0 for SPI0. |
| | | | | | O | SCT0_OUT5 — SCT0 output 5. PWM output 5. |
| | | | | | O | CT32B2_MAT1 — 32-bit CT32B2 match output 1. |
| | | | | | I | R — Reserved. |
| PIO0_15 | G6 | 50 | [2] | PU | I/O | PIO0_15 — General-purpose digital input/output pin. In boundary scan mode: TDO (Test Data Out). |
| | | | | | I/O | SPI0_SSEL1 — Slave Select 1 for SPI0. |
| | | | | | I/O | SWO — Serial wire trace output. |
| | | | | | O | CT32B2_MAT2 — 32-bit CT32B2 match output 2. |
| | | | | | I | R — Reserved. |
| SWCLK/ PIO0_16 | F5 | 52 | [2] | PU | I/O | PIO0_16 — General-purpose digital input/output pin. After booting, this pin is connected to the SWCLK. |
| | | | | | I/O | SPI0_SSEL2 — Slave Select 2 for SPI0. |
| | | | | | I | U1_CTS — Clear To Send input for USART1. |
| | | | | | O | CT32B3_MAT1 — 32-bit CT32B3 match output 1. |
| | | | | | I | R — Reserved. |
| | | | | | I/O | SWCLK — Serial Wire Clock. JTAG Test Clock. This is the default function after booting. |
| SWDIO/ PIO0_17 | G5 | 53 | [2] | PU | I/O | PIO0_17 — General-purpose digital input/output pin. After booting, this pin is connected to SWDIO. |
| | | | | | I/O | SPI0_SSEL3 — Slave Select 3 for SPI0. |
| | | | | | O | U1_RTS — Request To Send output for USART1. |
| | | | | | O | CT32B3_MAT2 — 32-bit CT32B3 match output 2. |
| | | | | | I | R — Reserved. |
| | | | | | I/O | SWDIO — Serial Wire Debug I/O. This is the default function after booting. |

Table 105. Pin description ...continued

| Symbol | WLCSP49 | LQFP64 | | Reset state [1] | Type [6] | Description |
|------------------|---------|--------|-----|-----------------|----------|--|
| PIO0_18/ TRST | G4 | 58 | [2] | PU | I/O | PIO0_18 — General-purpose digital input/output pin. In boundary scan mode: $\overline{\text{TRST}}$ (Test Reset). |
| | | | | | O | U3_TXD — Transmitter output for USART3. |
| | | | | | O | SCT0_OUT0 — SCT0 output 0. PWM output 0. |
| | | | | | O | CT32B0_MAT0 — 32-bit CT32B0 match output 0. |
| | | | | | I | R — Reserved. |
| PIO0_19/ TDI | G3 | 59 | [2] | PU | I/O | PIO0_19 — General-purpose digital input/output pin. In boundary scan mode: TDI (Test Data In). |
| | | | | | I/O | U3_SCLK — USART3 clock in synchronous USART mode. |
| | | | | | O | SCT0_OUT1 — SCT0 output 1. PWM output 1. |
| | | | | | O | CT32B0_MAT1 — 32-bit CT32B0 match output 1. |
| | | | | | I | R — Reserved. |
| PIO0_20/ TMS | F3 | 60 | [2] | PU | I/O | PIO0_20 — General-purpose digital input/output pin. In boundary scan mode: TMS (Test Mode Select). |
| | | | | | I | U3_RXD — Receiver input for USART3. |
| | | | | | I/O | U0_SCLK — USART0 clock in synchronous USART mode. |
| | | | | | I | CT32B3_CAP0 — 32-bit CT32B3 capture input 0. |
| | | | | | I | R — Reserved. |
| PIO0_21 | E3 | 61 | [2] | PU | I/O | PIO0_21 — General-purpose digital input/output pin. |
| | | | | | O | CLKOUT — Clock output pin. |
| | | | | | O | U0_TXD — Transmitter output for USART0. |
| | | | | | O | CT32B3_MAT0 — 32-bit CT32B3 match output 0. |
| | | | | | I | R — Reserved. |
| PIO0_22 | G2 | 63 | [2] | PU | I/O | PIO0_22 — General-purpose digital input/output pin. |
| | | | | | I | CLKIN — Clock input. |
| | | | | | I | U0_RXD — Receiver input for USART0. |
| | | | | | O | CT32B3_MAT3 — 32-bit CT32B3 match output 3. |
| | | | | | I | R — Reserved. |
| PIO0_23 | F2 | 1 | [3] | Z | I/O | PIO0_23 — General-purpose digital input/output pin. |
| | | | | | I/O | I2C0_SCL — I ² C0 clock input/output. |
| | | | | | I | R — Reserved. |
| | | | | | I | CT32B0_CAP0 — 32-bit CT32B0 capture input 0. |
| | | | | | I | R — Reserved. |

Table 105. Pin description ...continued

| Symbol | WLCSP49 | LQFP64 | Reset state [1] | Type [6] | Description | |
|--------------------|---------|--------|-----------------|----------|-------------|---|
| PIO0_24 | F1 | 2 | [3] | Z | I/O | PIO0_24 — General-purpose digital input/output pin. |
| | | | | | I/O | I2C0_SDA — I ² C0 data input/output. |
| | | | | | I | R — Reserved. |
| | | | | | I | CT32B0_CAP1 — 32-bit CT32B0 capture input 1. |
| | | | | | I | R — Reserved. |
| | | | | | O | CT32B0_MAT0 — 32-bit CT32B0 match output 0. |
| PIO0_25 | E2 | 3 | [3] | Z | I/O | PIO0_25 — General-purpose digital input/output pin. |
| | | | | | I/O | I2C1_SCL — I ² C1 clock input/output. |
| | | | | | I | U1_CTS — Clear To Send input for USART1. |
| | | | | | I | CT32B0_CAP2 — 32-bit CT32B0 capture input 2. |
| | | | | | I | R — Reserved. |
| | | | | | I | CT32B1_CAP1 — 32-bit CT32B1 capture input 1. |
| PIO0_26 | E1 | 4 | [3] | Z | I/O | PIO0_26 — General-purpose digital input/output pin. |
| | | | | | I/O | I2C1_SDA — I ² C1 data input/output. |
| | | | | | I | R — Reserved. |
| | | | | | I | CT32B0_CAP3 — 32-bit CT32B0 capture input 3. |
| | | | | | I | R — Reserved. |
| PIO0_27 | D2 | 5 | [3] | Z | I/O | PIO0_27 — General-purpose digital input/output pin. |
| | | | | | I/O | I2C2_SCL — I ² C2 clock input/output. |
| | | | | | I | R — Reserved. |
| | | | | | I | CT32B2_CAP0 — 32-bit CT32B2 capture input 0. |
| | | | | | I | R — Reserved. |
| PIO0_28 | D1 | 6 | [3] | Z | I/O | PIO0_28 — General-purpose digital input/output pin. |
| | | | | | I/O | I2C2_SDA — I ² C2 data input/output. |
| | | | | | I | R — Reserved. |
| | | | | | O | CT32B2_MAT0 — 32-bit CT32B2 match output 0. |
| | | | | | I | R — Reserved. |
| PIO0_29/A DC0_0 | D3 | 11 | [4] | PU | I/O; AI | PIO0_29/ADC0_0 — General-purpose digital input/output pin (default). ADC input channel 0 if the DIGIMODE bit is set to 0 in the IOCON register for this pin. |
| | | | | | - | R — Reserved. |
| | | | | | O | SCT0_OUT2 — SCT0 output 2. |
| | | | | | O | CT32B0_MAT3 — 32-bit CT32B0 match output 3. |
| | | | | | I | R — Reserved. |
| | | | | | I | CT32B0_CAP1 — 32-bit CT32B0 capture input 1. |
| | | | | | O | CT32B0_MAT1 — 32-bit CT32B0 match output 1. |
| | | | | | | |

Table 105. Pin description ...continued

| Symbol | WLCSP49 | LQFP64 | Reset state [1] | Type [6] | Description | |
|--------------------|---------|--------|-----------------|----------|---|---|
| PIO0_30/ ADC0_1 | C1 | 12 | [4] | PU | I/O; AI PIO0_30/ADC0_1 — General-purpose digital input/output pin (default). ADC input channel 1 if the DIGIMODE bit is set to 0 in the IOCON register for this pin. | |
| | | | | | - | R — Reserved. |
| | | | | | O | SCT0_OUT3 — SCT0 output 3. |
| | | | | | O | CT32B0_MAT2 — 32-bit CT32B0 match output 2. |
| | | | | | I | R — Reserved. |
| | | | | | I | CT32B0_CAP2 — 32-bit CT32B0 capture input 2. |
| PIO0_31/ ADC0_2 | C2 | 13 | [4] | PU | I/O; AI PIO0_31/ADC0_2 — General-purpose digital input/output pin (default). ADC input channel 2 if the DIGIMODE bit is set to 0 in the IOCON register for this pin. Remark: This pin is also used to force In-System Programming mode after device reset. See the Boot Process chapter for details. | |
| | | | | | - | R — Reserved. |
| | | | | | I | U2_CTS — Clear To Send input for USART2. |
| | | | | | I | CT32B2_CAP2 — 32-bit CT32B2 capture input 2. |
| | | | | | I | R — Reserved. |
| | | | | | I | CT32B0_CAP3 — 32-bit CT32B0 capture input 3. |
| PIO1_0/ ADC0_3 | C3 | 14 | [4] | PU | I/O; AI PIO1_0/ADC0_3 — General-purpose digital input/output pin (default). ADC input channel 3 if the DIGIMODE bit is set to 0 in the IOCON register for this pin. | |
| | | | | | - | R — Reserved. |
| | | | | | O | U2_RTS — Request To Send output for USART2. |
| | | | | | O | CT32B3_MAT1 — 32-bit CT32B3 match output 1. |
| | | | | | I | R — Reserved. |
| | | | | | I | CT32B0_CAP0 — 32-bit CT32B0 capture input 0. |
| PIO1_1/ ADC0_4 | B1 | 15 | [4] | PU | I/O; AI PIO1_1/ADC0_4 — General-purpose digital input/output pin (default). ADC input channel 4 if the DIGIMODE bit is set to 0 in the IOCON register for this pin. | |
| | | | | | - | R — Reserved. |
| | | | | | I/O | SWO — Serial wire trace output. |
| | | | | | O | SCT0_OUT4 — SCT0 output 4. |
| PIO1_2/ ADC0_5 | A1 | 16 | [4] | PU | I/O; AI PIO1_2/ADC0_5 — General-purpose digital input/output pin (default). ADC input channel 5 if the DIGIMODE bit is set to 0 in the IOCON register for this pin. | |
| | | | | | - | R — Reserved. |
| | | | | | I/O | SPI1_SSEL3 — Slave Select 3 for SPI1. |
| | | | | | O | SCT0_OUT5 — SCT0 output 5. |

Table 105. Pin description ...continued

| Symbol | WLCSP49 | LQFP64 | Reset state [4] | Type [6] | Description | |
|-------------------|---------|--------|-----------------|----------|--|---|
| PIO1_3/ ADC0_6 | B2 | 17 | [4] | PU | I/O; AI PIO1_3/ADC0_6 — General-purpose digital input/output pin (default). ADC input channel 6 if the DIGIMODE bit is set to 0 in the IOCON register for this pin. | |
| | | | | | - | R — Reserved. |
| | | | | | I/O | SPI1_SSEL2 — Slave Select 2 for SPI1. |
| | | | | | O | SCT0_OUT6 — SCT0 output 6. |
| | | | | | I | R — Reserved. |
| | | | | | I/O | SPI0_SCK — Serial clock for SPI0. |
| | | | | | I | CT32B0_CAP1 — 32-bit CT32B0 capture input 1. |
| PIO1_4/ ADC0_7 | A2 | 18 | [4] | PU | I/O; AI PIO1_4/ADC0_7 — General-purpose digital input/output pin (default). ADC input channel 7 if the DIGIMODE bit is set to 0 in the IOCON register for this pin. | |
| | | | | | - | R — Reserved. |
| | | | | | I/O | SPI1_SSEL1 — Slave Select 1 for SPI1. |
| | | | | | O | SCT0_OUT7 — SCT0 output 7. |
| | | | | | I | R — Reserved. |
| | | | | | I/O | SPI0_MISO — Master In Slave Out for SPI0. |
| | | | | | O | CT32B0_MAT1 — 32-bit CT32B0 match output 1. |
| PIO1_5/ ADC0_8 | B3 | 19 | [4] | PU | I/O; AI PIO1_5/ADC0_8 — General-purpose digital input/output pin (default). ADC input channel 8 if the DIGIMODE bit is set to 0 in the IOCON register for this pin. | |
| | | | | | - | R — Reserved. |
| | | | | | I/O | SPI1_SSEL0 — Slave Select 0 for SPI1. |
| | | | | | I | CT32B1_CAP0 — 32-bit CT32B1 capture input 0. |
| | | | | | I | R — Reserved. |
| | | | | | O | CT32B1_MAT3 — 32-bit CT32B1 match output 3. |
| | | | | | I | R — Reserved. |
| PIO1_6/ ADC0_9 | A5 | 26 | [4] | PU | I/O; AI PIO1_6/ADC0_9 — General-purpose digital input/output pin (default). ADC input channel 9 if the DIGIMODE bit is set to 0 in the IOCON register for this pin. | |
| | | | | | - | R — Reserved. |
| | | | | | I/O | SPI1_SCK — Serial clock for SPI1. |
| | | | | | I | CT32B1_CAP2 — 32-bit CT32B1 capture input 2. |
| | | | | | - | R — Reserved. |
| | | | | | O | CT32B1_MAT2 — 32-bit CT32B1 match output 2. |
| | | | | | I | R — Reserved. |

Table 105. Pin description ...continued

| Symbol | WLCSP49 | LQFP64 | Reset state [1] | Type [6] | Description | |
|--------------------|---------|--------|-----------------|----------|--|---|
| PIO1_7/ ADC0_10 | B5 | 27 | [4] | PU | I/O; AI PIO1_7/ADC0_10 — General-purpose digital input/output pin (default). ADC input channel 10 if the DIGIMODE bit is set to 0 in the IOCON register for this pin. | |
| | | | | | - | R — Reserved. |
| | | | | | I/O | SPI1_MOSI — Master Out Slave in for SPI1. |
| | | | | | O | CT32B1_MAT2 — 32-bit CT32B1 match output 2. |
| | | | | | - | R — Reserved. |
| | | | | | I | CT32B1_CAP2 — 32-bit CT32B1 capture input 2. |
| | | | | | I | R — Reserved. |
| PIO1_8/ ADC0_11 | C5 | 28 | [4] | PU | I/O; AI PIO1_8/ADC0_11 — General-purpose digital input/output pin (default). ADC input channel 11 if the DIGIMODE bit is set to 0 in the IOCON register for this pin. | |
| | | | | | - | R — Reserved. |
| | | | | | I/O | SPI1_MISO — Master In Slave Out for SPI1. |
| | | | | | O | CT32B1_MAT3 — 32-bit CT32B1 match output 3. |
| | | | | | I | R — Reserved. |
| | | | | | I | CT32B1_CAP3 — 32-bit CT32B1 capture input 3. |
| | | | | | I | R — Reserved. |
| PIO1_9 | - | 29 | [2] | PU | I/O PIO1_9 — General-purpose digital input/output pin. | |
| | | | | | I | R — Reserved. |
| | | | | | I/O | SPI0_MOSI — Master Out Slave In for SPI0. |
| | | | | | I | CT32B0_CAP2 — 32-bit CT32B0 capture input 2. |
| PIO1_10 | - | 30 | [2] | PU | I/O PIO1_10 — General-purpose digital input/output pin. | |
| | | | | | I | R — Reserved. |
| | | | | | O | U1_TXD — Transmitter output for USART1. |
| | | | | | O | SCT0_OUT4 — SCT0 output 4. |
| PIO1_11 | - | 42 | [2] | PU | I/O PIO1_11 — General-purpose digital input/output pin. | |
| | | | | | I | R — Reserved. |
| | | | | | O | U1_RTS — Request To Send output for USART1. |
| | | | | | I | CT32B1_CAP0 — 32-bit CT32B1 capture input 0. |
| PIO1_12 | - | 51 | [2] | PU | I/O PIO1_12 — General-purpose digital input/output pin. | |
| | | | | | I | R — Reserved. |
| | | | | | I | U3_RXD — Receiver input for USART3. |
| | | | | | O | CT32B1_MAT0 — 32-bit CT32B1 match output 0. |
| | | | | | I/O | SPI1_SCK — Serial clock for SPI1. |
| PIO1_13 | - | 54 | [2] | PU | I/O PIO1_13 — General-purpose digital input/output pin. | |
| | | | | | I | R — Reserved. |
| | | | | | O | U3_TXD — Transmitter output for USART3. |
| | | | | | O | CT32B1_MAT1 — 32-bit CT32B1 match output 1. |
| | | | | | I/O | SPI1_MOSI — Master Out Slave In for SPI1. |

Table 105. Pin description ...continued

| Symbol | WLCSP49 | LQFP64 | Reset state [1] | Type [6] | Description | |
|--------------|---------|---------------|-----------------|----------|-------------|---|
| PIO1_14 | - | 57 | [2] | PU | I/O | PIO1_14 — General-purpose digital input/output pin. |
| | | | | | I | R — Reserved. |
| | | | | | I | U2_RXD — Receiver input for USART2. |
| | | | | | O | SCT0_OUT7 — SCT0 output 7. |
| | | | | | I/O | SPI1_MISO — Master In Slave Out for SPI1. |
| PIO1_15 | - | 62 | [2] | PU | I/O | PIO1_15 — General-purpose digital input/output pin. |
| | | | | | I | R — Reserved. |
| | | | | | O | SCT0_OUT5 — SCT0 output 5. |
| | | | | | I | CT32B1_CAP3 — 32-bit CT32B1 capture input 3. |
| | | | | | I/O | SPI1_SSEL0 — Slave Select 0 for SPI1. |
| PIO1_16 | - | 7 | [2] | PU | I/O | PIO1_16 — General-purpose digital input/output pin. |
| | | | | | I | R — Reserved. |
| | | | | | O | CT32B0_MAT0 — 32-bit CT32B0 match output 0. |
| | | | | | I | CT32B0_CAP0 — 32-bit CT32B0 capture input 0. |
| | | | | | I/O | SPI1_SSEL1 — Slave Select 1 for SPI1. |
| PIO1_17 | - | 10 | [2] | PU | I/O | PIO1_17 — General-purpose digital input/output pin. |
| RESET | G1 | 64 | [5] | PU | I | External reset input: A LOW on this pin resets the device, causing I/O ports and peripherals to take on their default states, and processor execution to begin at address 0. Wakes up the part from Deep power-down mode. |
| RTCXIN | A7 | 33 | - | - | - | RTC oscillator input. |
| RTCXOUT | B7 | 35 | - | - | - | RTC oscillator output. |
| VREFP | B4 | 22 | - | - | - | ADC positive reference voltage. |
| VREFN | - | 21 | - | - | - | ADC negative reference voltage. |
| VDDA | A4 | 23 | - | - | - | Analog 3.3 V supply voltage. |
| VDD | C4, F4 | 8, 24, 56, 34 | - | - | - | Single 1.62 V to 3.6 V power supply powers internal digital functions and I/Os. |
| VSS | D4, E4 | 9, 25, 55 | - | - | - | Ground. |
| VSSA | A3 | 20 | - | - | - | Analog ground. |

[1] PU = input mode, pull-up enabled (pull-up resistor pulls up pin to V_{DD}).
Z = high impedance; pull-up or pull-down disabled.
Reset state reflects the pin state at reset without boot code operation.

[2] 5 V tolerant pad with programmable glitch filter (5 V tolerant if V_{DD} present; if V_{DD} not present, do not exceed 3.3 V); provides digital I/O functions with TTL levels and hysteresis; normal drive strength. Pulse width of spikes or glitches suppressed by input filter is from 3 ns to 16 ns (simulated value).

[3] Specialized I2C pads.

[4] Digital I/O pad with analog functionality.

- [5] Reset pad. 5V tolerant pad with glitch filter with hysteresis. Pulse width of spikes or glitches suppressed by input filter is from 3 ns to 20 ns (simulated value).
- [6] I = Input; AI = Analog Input; O = Output.

8.1 How to read this chapter

The IOCON block is included on all LPC5410x parts. Registers for pins that are not available on a specific package are reserved.

Table 106. Available pins and configuration registers

| Package | Total GPIOs | GPIO Port 0 | GPIO Port 1 |
|-----------------------------------|-------------|-------------------------------------|-------------------|
| 64-pin device with RTC oscillator | 50 | PIO0_0 to PIO0_31 | PIO1_0 to PIO1_17 |
| 49-pin device with RTC oscillator | 39 | PIO0_0 to PIO0_1, PIO0_4 to POIO_31 | PIO1_0 to PIO1_8 |

8.2 Features

The following electrical properties are configurable for standard port pins:

- Pull-up/pull-down resistor
- Open-drain mode
- Inverted function

Pins PIO0_23 through PIO0_28 are true open-drain pins that can be configured for different I²C-bus speeds.

8.3 Basic configuration

Enable the clock to the IOCON in the AHBCLKCTRL0 register ([Table 51](#)). Once the pins are configured, the IOCON clock can be disabled in order to conserve power.

8.4 General description

8.4.1 Pin configuration

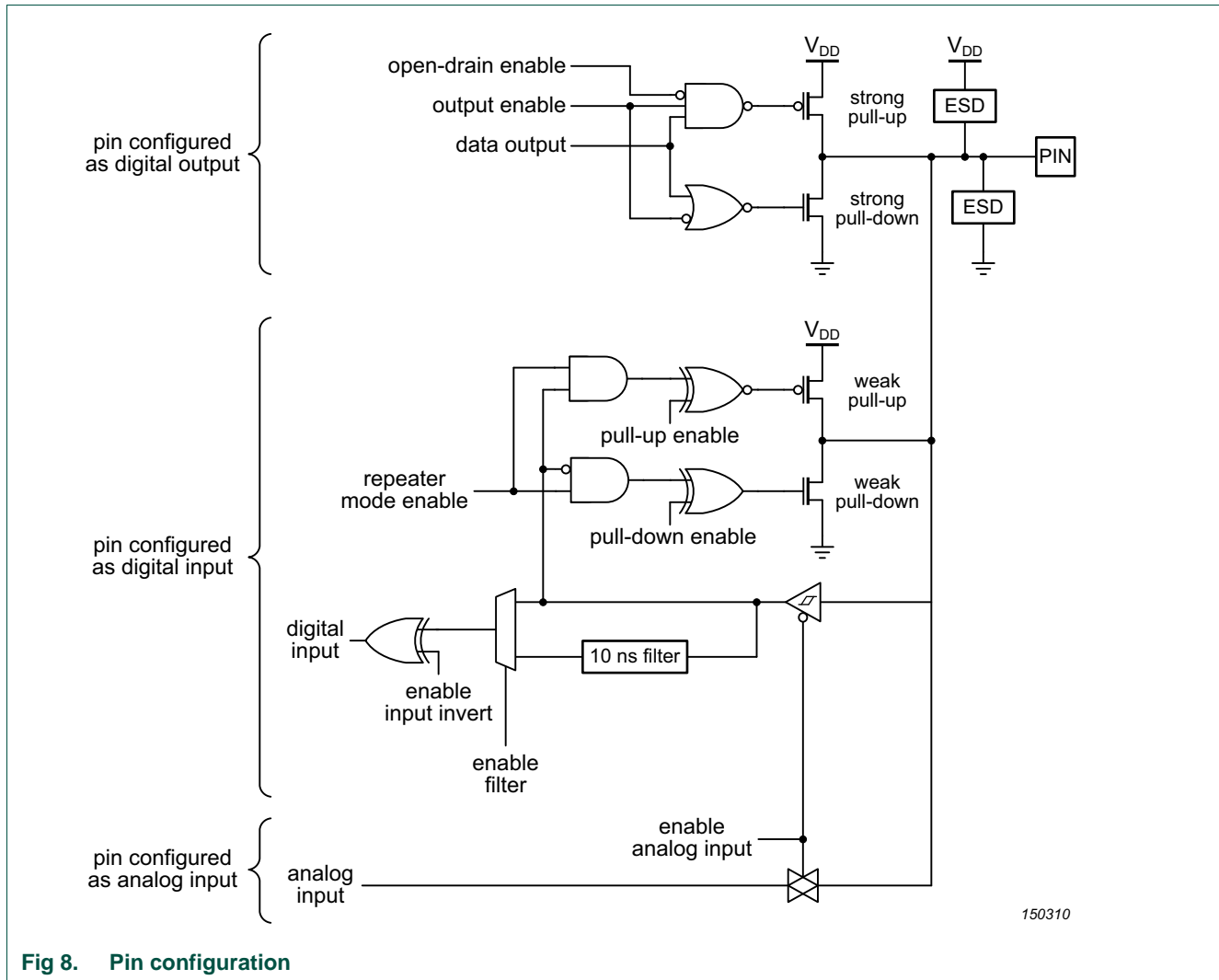


Fig 8. Pin configuration

8.4.2 IOCON registers

The IOCON registers control the functions of device pins. Each GPIO pin has a dedicated control register to select its function and characteristics. Each pin has a unique set of functional capabilities. Not all pin characteristics are selectable on all pins. For instance, pins that have an I²C function can be configured for different I²C-bus modes, while pins that have an analog alternate function have an analog mode can be selected. Details of the IOCON registers are in [Section 8.4.2](#). The following sections describe specific characteristics of pins.

Multiple connections

Since a particular peripheral function may be allowed on more than one pin, it is possible to configure more than one pin to perform the same function. If a peripheral output function is configured to appear on more than one pin, it will in fact be routed to those

pins. If a peripheral input function is defined as coming from more than one source, the values will be logically combined, possibly resulting in incorrect peripheral operation. Therefore care should be taken to avoid this situation.

8.4.2.1 Pin function

The FUNC bits in the IOCON registers can be set to GPIO (typically value 000) or to a special function. For pins set to GPIO, the DIR registers determine whether the pin is configured as an input or output (see [Section 10.5.3](#)). For any special function, the pin direction is controlled automatically depending on the function. The FIONDIR registers have no effect for special functions.

8.4.2.2 Pin mode

The MODE bits in the IOCON register allow the selection of on-chip pull-up or pull-down resistors for each pin or select the repeater mode.

The possible on-chip resistor configurations are pull-up enabled, pull-down enabled, or no pull-up/pull-down. The default value is pull-up enabled.

The repeater mode enables the pull-up resistor if the pin is high and enables the pull-down resistor if the pin is low. This causes the pin to retain its last known state if it is configured as an input and is not driven externally. Such state retention is not applicable to the Deep Power-down mode. Repeater mode may typically be used to prevent a pin from floating (and potentially using significant power if it floats to an indeterminate state) if it is temporarily not driven.

8.4.2.3 Hysteresis

The input buffer for digital functions has built-in hysteresis. See the appropriate specific device data sheet for quantitative details.

8.4.2.4 Invert pin

This option is included to avoid having to include an external inverter on an input that is meant to be the opposite polarity of the external signal.

8.4.2.5 Analog/digital mode

When not in digital mode (DIGIMODE = 0) a pin is in analog mode, some digital pin functions are disabled and any analog pin functions are enabled. In digital mode (DIGIMODE = 1), any analog pin functions are disabled and digital pin functions are enabled. This protects the analog input from voltages outside the range of the analog power supply and reference that may sometimes be present on digital pins, since they are typically 5V tolerant. All pin types include this control, even if they do not support any analog functions.

In order to use a pin that has an ADC input option for that purpose, select GPIO (FUNC field = 0) and disable the digital pin function (DIGIMODE = 0).

In analog mode, the MODE field should be "Inactive" (00); the INVERT, FILTEROFF, and OD settings have no effect. For an unconnected pin that has an analog function, keep the DIGIMODE bit set to 1 (digital mode), and pull-up or pull-down mode selected in the MODE field.

8.4.2.6 Input filter

Some pins include a filter that can be selectively disabled by setting the FILTEROFF bit. The filter suppresses input pulses smaller than about 10 ns.

8.4.2.7 Output slew rate

The SLEW bits of digital outputs that do not need to switch state very quickly should be set to “standard”. This setting allows multiple outputs to switch simultaneously without noticeably degrading the power/ground distribution of the device, and has only a small effect on signal transition time. This is particularly important if analog accuracy is significant to the application. See the relevant specific device data sheet for more details.

8.4.2.8 I²C modes

Pins that support I²C with specialized pad electronics (P0[23] through P0[28]) have additional configuration bits. These have multiple configurations to support I²C variants. These are not hard-wired so that the pins can be more easily used for non-I²C functions. See [Table 113](#) for recommended mode settings.

For non-I²C operation, these pins remain open-drain and can only drive low, regardless of how I2CSLEW and I2CDRIVE are set. They would typically be used with an external pull-up resistor if they are used as outputs unless they are used only to sink current. Leave I2CSLEW = 1, I2CDRIVE = 0, and I2CFILTER = 0 to maximize compatibility with other GPIO pins.

8.4.2.9 Open-Drain Mode

When output is selected, either by selecting a special function in the FUNC field, or by selecting the GPIO function for a pin having a 1 in the related bit of that port's DIR register, a 1 in the OD bit selects open-drain operation, that is, a 1 disables the high-drive transistor. This option has no effect on the primary I²C pins. Note that the properties of a pin in this simulated open-drain mode are somewhat different than those of a true open drain output.

8.5 Register description

Each port pin PIOm_n has one IOCON register assigned to control the pin's electrical characteristics.

Table 107. Register overview: I/O configuration (base address 0x4001 C000)

| Name | Access | Offset | Description | Reset value ^[1] | Pin type | Section |
|--------------|--------|---------------|---|------------------------------------|----------|-----------------------|
| PIO0_[0:1] | R/W | [0x000:0x004] | Digital I/O control for port 0 pins PIO0_0 to PIO0_1 | 0x0190 | D | 8.5.1 |
| PIO0_[4:22] | R/W | [0x010:0x058] | Digital I/O control for port 0 pins PIO4 to PIO0_22. | PIO0_16/17: 0x0195, others: 0x0190 | D | 8.5.1 |
| PIO0_[23:28] | R/W | [0x05C:0x070] | Digital I/O control for port 0 pins PIO0_23 to PIO0_28. These pins support I ² C with true open-drain, drive and filtering for modes up to Fast-mode Plus. | 0x01A0 | I | 8.5.2 |
| PIO0_[29:31] | R/W | [0x074:0x07C] | Digital I/O control for port 0 pins PIO0_29 to PIO0_31. These pins include an ADC input. | 0x0190 | A | 8.5.3 |
| PIO1_[0:8] | R/W | [0x080:0x0A0] | Digital I/O control for port 1 pins PIO0_0 to PIO0_8. These pins include an ADC input. | 0x0190 | A | 8.5.3 |
| PIO1_[9:17] | R/W | [0x0A4:0x0C4] | Digital I/O control for port 1 pins PIO1_9 to PIO1_17. | 0x0190 | D | 8.5.4 |

[1] Reset Value reflects the data stored in defined bits only. Reserved bits assumed to be 0.

8.5.1 Type D IOCON registers (PIO0)

This IOCON table applies to port pins P0[0 to 2] and P0[4 to 22]. Other pins include ADC or I²C functions that alter the contents of the related IOCON registers.

Remark: The FUNC field for P0[16] and P0[17] resets to 0b101 (0x5), selecting the Serial Wire Debug function by default.

Table 108. Address map PIO0_[0:22] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| IOCON | 0x4001 C000 | [0x000:0x058] | 0x4 | 23 |

Table 109. Type D IOCON registers (PIO0_[0:22], address offsets [0x000:0x058]) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|-----------|--|--|----------------------------|
| 2:0 | FUNC | | Selects pin function. | PIO0_16/17: 5 others: 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 10 |
| | | 0x0 | Inactive. Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down. Pull-down resistor enabled. | |
| | | 0x2 | Pull-up. Pull-up resistor enabled. | |
| | 0x3 | Repeater. Repeater mode. | | |
| 5 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 6 | INVERT | | Input polarity. | 0 |
| | | 0 | Disabled. Input function is not inverted. | |
| | 1 | Enabled. Input is function inverted. | | |
| 7 | DIGIMODE | | Select Analog/Digital mode. | 1 |
| | | 0 | Analog mode. | |
| | 1 | Digital mode. | | |
| 8 | FILTEROFF | | Controls input glitch filter. | 1 |
| | | 0 | Filter enabled. Noise pulses below approximately 10 ns are filtered out. | |
| | 1 | Filter disabled. No input filtering is done | | |
| 9 | SLEW | | Driver slew rate. | 0 |
| | | 0 | Standard mode, output slew rate control is enabled. More outputs can be switched simultaneously. | |
| | 1 | Fast mode, slew rate control is disabled. Refer to the appropriate specific device data sheet for details. | | |
| 10 | OD | | Controls open-drain mode. | 0 |
| | | 0 | Normal. Normal push-pull output | |
| | 1 | Open-drain. Simulated open-drain output (high drive disabled) | | |
| 31:11 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

Table 110. Type D I/O Control registers: FUNC values and pin functions

| Register | Value of FUNC field in IOCON register | | | | | | | |
|----------|---------------------------------------|-------------|-------------|-------------|-----|-------------|-----|-----|
| | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| PIO0_0 | PIO0_0 | U0_RXD | SPI0_SSELN0 | CT32B0_CAP0 | | SCT0_OUT3 | | |
| PIO0_1 | PIO0_1 | U0_TXD | SPI0_SSELN1 | CT32B0_CAP1 | | SCT0_OUT1 | | |
| PIO0_2 | PIO0_2 | U0_CTS | | CT32B2_CAP1 | | | | |
| PIO0_3 | PIO0_3 | U0_RTS | | CT32B1_MAT3 | | | | |
| PIO0_4 | PIO0_4 | U0_SCLK | SPI0_SSELN2 | CT32B0_CAP2 | | | | |
| PIO0_5 | PIO0_5 | U1_RXD | SCT0_OUT6 | CT32B0_MAT0 | | | | |
| PIO0_6 | PIO0_6 | U1_TXD | | CT32B0_MAT1 | | | | |
| PIO0_7 | PIO0_7 | U1_SCLK | SCT0_OUT0 | CT32B0_MAT2 | | CT32B0_CAP2 | | |
| PIO0_8 | PIO0_8 | U2_RXD | SCT0_OUT1 | CT32B0_MAT3 | | | | |
| PIO0_9 | PIO0_9 | U2_TXD | SCT0_OUT2 | CT32B3_CAP0 | | SPI0_SSELN0 | | |
| PIO0_10 | PIO0_10 | U2_SCLK | SCT0_OUT3 | CT32B3_MAT0 | | | | |
| PIO0_11 | PIO0_11 | SPI0_SCK | U1_RXD | CT32B2_MAT1 | | | | |
| PIO0_12 | PIO0_12 | SPI0_MOSI | U1_TXD | CT32B2_MAT3 | | | | |
| PIO0_13 | PIO0_13 | SPI0_MISO | SCT0_OUT4 | CT32B2_MAT0 | | | | |
| PIO0_14 | PIO0_14 | SPI0_SSELN0 | SCT0_OUT5 | CT32B2_MAT1 | | | | |
| PIO0_15 | PIO0_15 | SPI0_SSELN1 | SWO | CT32B2_MAT2 | | | | |
| PIO0_16 | PIO0_16 | SPI0_SSELN2 | U1_CTS | CT32B3_MAT1 | | SWCLK | | |
| PIO0_17 | PIO0_17 | SPI0_SSELN3 | U1_RTS | CT32B3_MAT2 | | SWDIO | | |
| PIO0_18 | PIO0_18 | U3_TXD | SCT0_OUT0 | CT32B0_MAT0 | | | | |
| PIO0_19 | PIO0_19 | U3_SCLK | SCT0_OUT1 | CT32B0_MAT1 | | | | |
| PIO0_20 | PIO0_20 | U3_RXD | U0_SCLK | CT32B3_CAP0 | | | | |
| PIO0_21 | PIO0_21 | CLKOUT | U0_TXD | CT32B3_MAT0 | | | | |
| PIO0_22 | PIO0_22 | CLKIN | U0_RXD | CT32B3_MAT3 | | | | |

8.5.2 Type I IOCON registers (PIO0)

This IOCON table applies to pins P0[23 to 28]. See [Table 113](#) for recommended setting for I2C operation.

Table 111. Address map PIO0_[23:28] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| IOCON | 0x4001 C000 | [0x05C:0x070] | 0x4 | 6 |

Table 112. Type I IOCON registers (PIO0_[23:28], address offsets [0x05C:0x070]) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|-----------|-------|--|-------------|
| 2:0 | FUNC | | Selects pin function. | 0 |
| 4:3 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 5 | I2CSLEW | | Controls slew rate of I ² C pad. | 1 |
| | | 0 | I ² C mode. | |
| | | 1 | GPIO mode. | |
| 6 | INVERT | 1 | Input polarity. | 0 |
| | | 0 | Disabled. Input function is not inverted. | |
| | | 1 | Enabled. Input is function inverted. | |
| 7 | DIGIMODE | | Select Analog/Digital mode. | 1 |
| | | 0 | Analog mode. | |
| | | 1 | Digital mode. | |
| 8 | FILTEROFF | | Controls input glitch filter. | 1 |
| | | 0 | Filter enabled. Noise pulses below approximately 10 ns are filtered out | |
| | | 1 | Filter disabled. No input filtering is done | |
| 9 | I2CDRIVE | | Controls the current sink capability of the pin. | 0 |
| | | 0 | Low drive. Output drive sink is 4 mA. This is sufficient for standard and fast mode I ² C. | |
| | | 1 | High drive. Output drive sink is 20 mA. This is needed for Fast Mode Plus I ² C. Refer to the appropriate specific device data sheet for details. | |
| 10 | I2CFILTER | | Configures I ² C features for standard mode, fast mode, and Fast Mode Plus operation. | 0 |
| | | 0 | Enabled. I ² C 50 ns glitch filter enabled. | |
| | | 1 | Disabled. I ² C 50 ns glitch filter disabled. | |
| 31:11 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

Table 113. Suggested IOCON settings for I2C functions

| Mode | IOCON register bit | | | | | |
|---------------------------------------|--------------------|-------------|-----------------------|-----------------------|-----------|------------|
| | 10: I2CFILTER | 9: I2CDRIVE | 8: FILTEROFF | 7: DIGIMODE | 6: INVERT | 5: I2CSLEW |
| GPIO 4 mA drive | 0 | 0 | 0 [1] | 1 [2] | 0 | 1 |
| GPIO 20 mA drive | 0 | 1 | 0 [1] | 1 [2] | 0 | 1 |
| Fast / Standard mode I ² C | 0 | 0 | 1 | 1 | 0 | 0 |
| Fast Mode Plus I ² C | 1 | 1 | 1 | 1 | 0 | 0 |
| High Speed slave I ² C | 1 | 1 | 1 | 1 | 0 | 0 |

[1] The input filter may be turned by setting FILTEROFF off if it is not needed.

[2] The input may be turned off by clearing DIGIMODE if it is not needed.

Table 114. Type I I/O Control registers: FUNC values and pin functions

| Register | Value of FUNC field in IOCON register | | | | | | | |
|----------|---------------------------------------|----------|--------|-------------|-----|-------------|-----|-----|
| | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| PIO0_23 | PIO0_23 | I2C0_SCL | | CT32B0_CAP0 | | | | |
| PIO0_24 | PIO0_24 | I2C0_SDA | | CT32B0_CAP1 | | CT32B0_MAT0 | | |
| PIO0_25 | PIO0_25 | I2C1_SCL | U1_CTS | CT32B0_CAP2 | | CT32B1_CAP1 | | |
| PIO0_26 | PIO0_26 | I2C1_SDA | | CT32B0_CAP3 | | | | |
| PIO0_27 | PIO0_27 | I2C2_SCL | | CT32B2_CAP0 | | | | |
| PIO0_28 | PIO0_28 | I2C2_SDA | | CT32B2_MAT0 | | | | |

8.5.3 Type A IOCON registers (PIO0, PIO1)

This IOCON table applies to pins P0[29 to 31], P1[0 to 8].

Table 115. Address map PIO0_[29:31] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| IOCON | 0x4001 C000 | [0x074:0x07C] | 0x4 | 3 |

Table 116. Type A IOCON registers(PIO0_[29:31], address offsets [0x074:0x07C]) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|-----------|-------|---|-------------|
| 2:0 | FUNC | | Selects pin function. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 10 |
| | | 0x0 | Inactive. Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down. Pull-down resistor enabled. | |
| | | 0x2 | Pull-up. Pull-up resistor enabled. | |
| | | 0x3 | Repeater. Repeater mode. | |
| 5 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 6 | INVERT | | Input polarity. | 0 |
| | | 0 | Disabled. Input function is not inverted. | |
| | | 1 | Enabled. Input is function inverted. | |
| 7 | DIGIMODE | | Select Analog/Digital mode. | 1 |
| | | 0 | Analog mode. | |
| | | 1 | Digital mode. | |
| 8 | FILTEROFF | | Controls input glitch filter. | 1 |
| | | 0 | Filter enabled. Noise pulses below approximately 10 ns are filtered out | |
| | | 1 | Filter disabled. No input filtering is done | |
| 9 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

Table 116. Type A IOCON registers(PIO0_[29:31], address offsets [0x074:0x07C]) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|---|-------------|
| 10 | OD | | Controls open-drain mode. | 0 |
| | | 0 | Normal. Normal push-pull output | |
| | | 1 | Open-drain. Simulated open-drain output (high drive disabled) | |
| 31:11 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

Table 117. Address map PIO1_[0:8] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| IOCON | 0x4001 C000 | [0x080:0x0A0] | 0x4 | 9 |

Table 118. Type A IOCON registers(PIO1_[0:8], address offsets [0x080:0x0A0]) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|-----------|-------|---|-------------|
| 2:0 | FUNC | | Selects pin function. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 10 |
| | | 0x0 | Inactive. Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down. Pull-down resistor enabled. | |
| | | 0x2 | Pull-up. Pull-up resistor enabled. | |
| | | 0x3 | Repeater. Repeater mode. | |
| 5 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 6 | INVERT | | Input polarity. | 0 |
| | | 0 | Disabled. Input function is not inverted. | |
| | | 1 | Enabled. Input is function inverted. | |
| 7 | DIGIMODE | | Select Analog/Digital mode. | 1 |
| | | 0 | Analog mode. | |
| | | 1 | Digital mode. | |
| 8 | FILTEROFF | | Controls input glitch filter. | 1 |
| | | 0 | Filter enabled. Noise pulses below approximately 10 ns are filtered out | |
| | | 1 | Filter disabled. No input filtering is done | |
| 9 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 10 | OD | | Controls open-drain mode. | 0 |
| | | 0 | Normal. Normal push-pull output | |
| | | 1 | Open-drain. Simulated open-drain output (high drive disabled) | |
| 31:11 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

Table 119. Type A I/O Control registers: FUNC values and pin functions

| Register | Value of FUNC field in IOCON register | | | | | | | |
|----------|---------------------------------------|-----|-------------|-------------|-----|-------------|-------------|-----|
| | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| PIO0_29 | PIO0_29/ADC_0 | | SCT0_OUT2 | CT32B0_MAT3 | | CT32B0_CAP1 | CT32B0_MAT1 | |
| PIO0_30 | PIO0_30/ADC_1 | | SCT0_OUT3 | CT32B0_MAT2 | | CT32B0_CAP2 | | |
| PIO0_31 | PIO0_31/ADC_2 | | U2_CTS | CT32B2_CAP2 | | CT32B0_CAP3 | CT32B0_MAT3 | |
| PIO1_0 | PIO1_0/ADC_3 | | U2_RTS | CT32B3_MAT1 | | CT32B0_CAP0 | | |
| PIO1_1 | PIO1_1/ADC_4 | | SWO | SCT0_OUT4 | | | | |
| PIO1_2 | PIO1_2/ADC_5 | | SPI1_SSELN3 | SCT0_OUT5 | | | | |
| PIO1_3 | PIO1_3/ADC_6 | | SPI1_SSELN2 | SCT0_OUT6 | | SPI0_SCK | CT32B0_CAP1 | |
| PIO1_4 | PIO1_4/ADC_7 | | SPI1_SSELN1 | SCT0_OUT7 | | SPI0_MISO | CT32B0_MAT1 | |
| PIO1_5 | PIO1_5/ADC_8 | | SPI1_SSELN0 | CT32B1_CAP0 | | CT32B1_MAT3 | | |
| PIO1_6 | PIO1_6/ADC_9 | | SPI1_SCK | CT32B1_CAP2 | | CT32B1_MAT2 | | |
| PIO1_7 | PIO1_7/ADC_10 | | SPI1_MOSI | CT32B1_MAT2 | | CT32B1_CAP2 | | |
| PIO1_8 | PIO1_8/ADC_11 | | SPI1_MISO | CT32B1_MAT3 | | CT32B1_CAP3 | | |

[1] To enable an ADC input, select the GPIO function and disable the digital functions of the pin by clearing the DIGIMODE bit in the related IOCON register.

8.5.4 Type D IOCON registers (PIO1)

This IOCON table applies to port pins P1[9 to 17]. Other pins include ADC or I²C functions that alter the contents of the related IOCON registers.

Table 120. Address map PIO1_[9:17] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| IOCON | 0x4001 C000 | [0x0A4:0x0C4] | 0x4 | 9 |

Table 121. Type D IOCON registers (PIO1_[9:17], address offsets [0x0A4:0x0C4]) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|-----------|-------|--|-------------|
| 2:0 | FUNC | | Selects pin function. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 10 |
| | | 0x0 | Inactive. Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down. Pull-down resistor enabled. | |
| | | 0x2 | Pull-up. Pull-up resistor enabled. | |
| | | 0x3 | Repeater. Repeater mode. | |
| 5 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 6 | INVERT | | Input polarity. | 0 |
| | | 0 | Disabled. Input function is not inverted. | |
| | | 1 | Enabled. Input is function inverted. | |
| 7 | DIGIMODE | | Select Analog/Digital mode. | 1 |
| | | 0 | Analog mode. | |
| | | 1 | Digital mode. | |
| 8 | FILTEROFF | | Controls input glitch filter. | 1 |
| | | 0 | Filter enabled. Noise pulses below approximately 10 ns are filtered out | |
| | | 1 | Filter disabled. No input filtering is done | |
| 9 | SLEW | | Driver slew rate. | 0 |
| | | 0 | Standard mode, output slew rate control is enabled. More outputs can be switched simultaneously. | |
| | | 1 | Fast mode, slew rate control is disabled. Refer to the appropriate specific device data sheet for details. | |
| 10 | OD | | Controls open-drain mode. | 0 |
| | | 0 | Normal. Normal push-pull output | |
| | | 1 | Open-drain. Simulated open-drain output (high drive disabled) | |
| 31:11 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

Table 122. Type D I/O Control registers: FUNC values and pin functions

| Register | Value of FUNC field in IOCON register | | | | | | | |
|----------|---------------------------------------|-----|-------------|-------------|-------------|-----|-----|-----|
| | 000 | 001 | 010 | 011 | 100 | 101 | 110 | 111 |
| PIO1_9 | PIO1_9 | | SPI0_MOSI | CT32B0_CAP2 | | | | |
| PIO1_10 | PIO1_10 | | U1_TXD | SCT0_OUT4 | | | | |
| PIO1_11 | PIO1_11 | | U1_RTS | CT32B1_CAP0 | | | | |
| PIO1_12 | PIO1_12 | | U3_RXD | CT32B1_MAT0 | SPI1_SCK | | | |
| PIO1_13 | PIO1_13 | | U3_TXD | CT32B1_MAT1 | SPI1_MOSI | | | |
| PIO1_14 | PIO1_14 | | U2_RXD | SCT0_OUT7 | SPI1_MISO | | | |
| PIO1_15 | PIO1_15 | | SCT0_OUT5 | CT32B1_CAP3 | SPI1_SSELN0 | | | |
| PIO1_16 | PIO1_16 | | CT32B0_MAT0 | CT32B0_CAP0 | SPI1_SSELN1 | | | |
| PIO1_17 | PIO1_17 | | | | | | | |

9.1 How to read this chapter

Input multiplexing is available for all parts. Depending on the package, not all inputs from external pins may be available.

9.2 Features

- Configures the inputs to the pin interrupt block and pattern match engine.
- Configures the inputs to the DMA triggers.
- Configures the inputs to the frequency measure function. This function is controlled by the FREQMECTRL register in the SYSCON block.

9.3 Basic configuration

Once set up, no clocks are needed for the input multiplexer to function. The system clock is needed only to write to or read from the INPUT MUX registers. Once the input multiplexer is configured, disable the clock to the INPUT MUX block in the AHBCLKCTRL register.

9.4 Pin description

The input multiplexer has no dedicated pins. However, all digital pins of ports 0 and 1 can be selected as inputs to the pin interrupts. Multiplexer inputs from external pins work independently of any other function assigned to the pin as long as no analog function is enabled.

Table 123. INPUT MUX pin description

| Pins | Peripheral | Reference |
|----------------------------------|-------------------------|---------------------------|
| Any existing pin on port 0 or 1 | Pin interrupts 0 to 7 | Table 126 |
| PIO0_4, PIO0_20, PIO0_24, PIO1_4 | Frequency measure block | Table 131 |

9.5 General description

The inputs to the DMA triggers, to the eight pin interrupts, and to the frequency measure block are multiplexed to multiple input sources. The sources can be external pins, interrupts, or output signals of other peripherals.

The input multiplexing makes it possible to design event-driven processes without CPU intervention by connecting peripherals like the ADC.

The DMA can use trigger input multiplexing to sequence DMA transactions without the use of interrupt service routines.

9.5.1 Pin interrupt input multiplexing

The input mux for the pin interrupts and pattern match engine multiplexes all existing pins from ports 0 and 1.

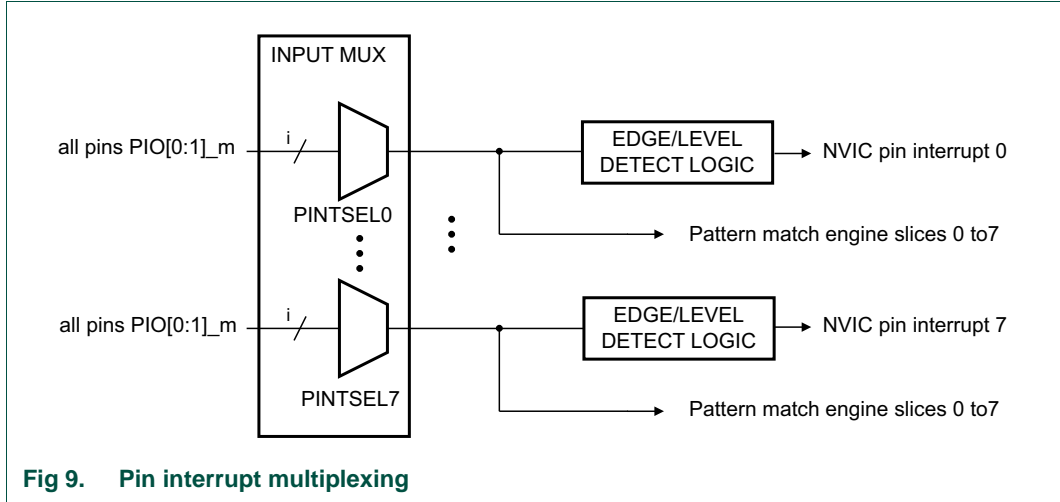


Fig 9. Pin interrupt multiplexing

9.5.2 DMA trigger input multiplexing

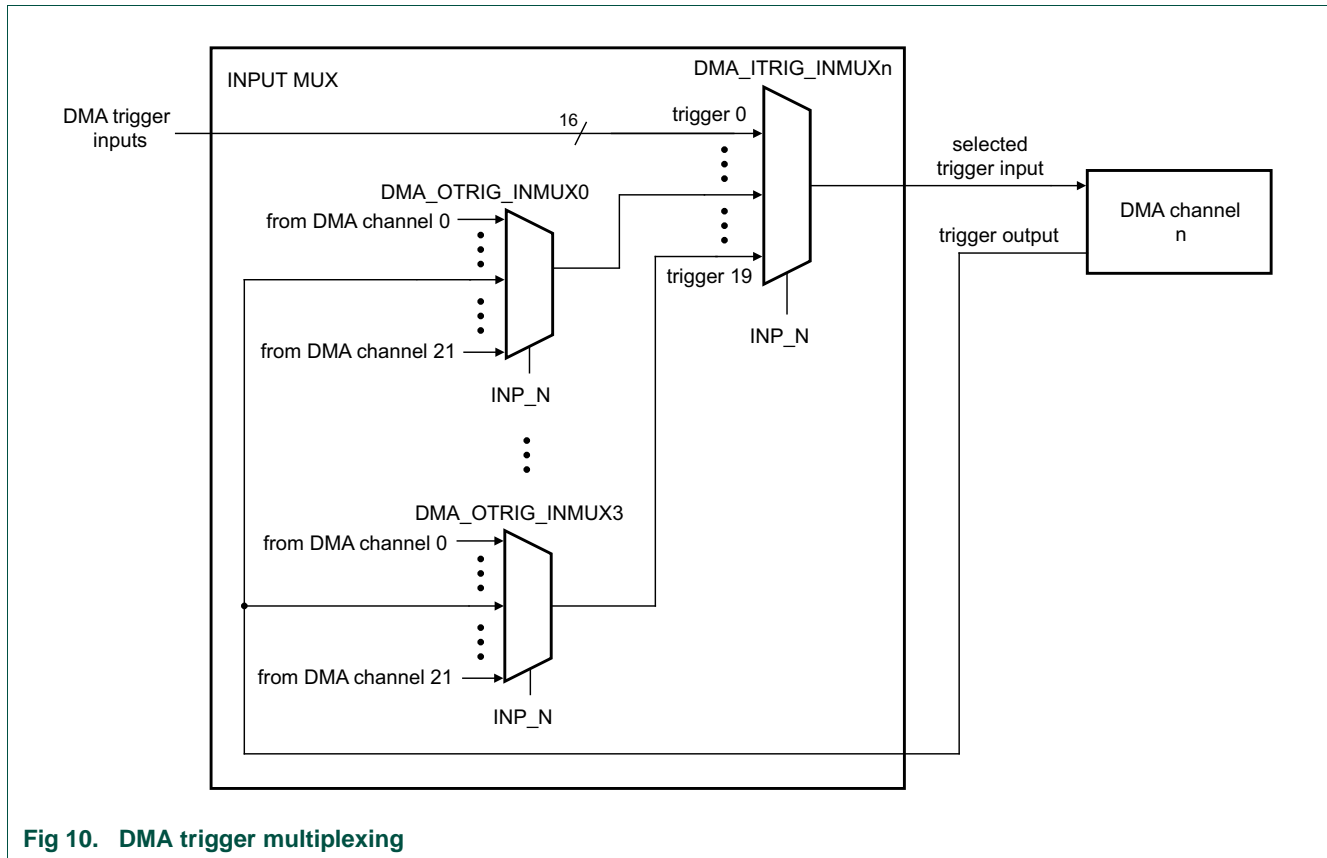


Fig 10. DMA trigger multiplexing

9.6 Register description

All input mux registers reside on word address boundaries. Details of the registers appear in the description of each function.

All address offsets not shown in [Table 124](#) are reserved and should not be written to.

Table 124. Register overview: Input multiplexing (base address 0x4005 0000)

| Name | Access | Offset | Description | Reset value | Reference |
|-------------------|--------|--------|---|-------------|---------------------------|
| PINTSEL0 | R/W | 0x0C0 | Pin interrupt select register 0 | 0x0 | Table 126 |
| PINTSEL1 | R/W | 0x0C4 | Pin interrupt select register 1 | 0x0 | Table 126 |
| PINTSEL2 | R/W | 0x0C8 | Pin interrupt select register 2 | 0x0 | Table 126 |
| PINTSEL3 | R/W | 0x0CC | Pin interrupt select register 3 | 0x0 | Table 126 |
| PINTSEL4 | R/W | 0x0D0 | Pin interrupt select register 4 | 0x0 | Table 126 |
| PINTSEL5 | R/W | 0x0D4 | Pin interrupt select register 5 | 0x0 | Table 126 |
| PINTSEL6 | R/W | 0x0D8 | Pin interrupt select register 6 | 0x0 | Table 126 |
| PINTSEL7 | R/W | 0x0DC | Pin interrupt select register 7 | 0x0 | Table 126 |
| DMA_ITRIG_INMUX0 | R/W | 0x0E0 | Trigger select register for DMA channel 0 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX1 | R/W | 0x0E4 | Trigger select register for DMA channel 1 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX2 | R/W | 0x0E8 | Trigger select register for DMA channel 2 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX3 | R/W | 0x0EC | Trigger select register for DMA channel 3 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX4 | R/W | 0x0F0 | Trigger select register for DMA channel 4 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX5 | R/W | 0x0F4 | Trigger select register for DMA channel 5 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX6 | R/W | 0x0F8 | Trigger select register for DMA channel 6 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX7 | R/W | 0x0FC | Trigger select register for DMA channel 7 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX8 | R/W | 0x100 | Trigger select register for DMA channel 8 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX9 | R/W | 0x104 | Trigger select register for DMA channel 9 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX10 | R/W | 0x108 | Trigger select register for DMA channel 10 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX11 | R/W | 0x10C | Trigger select register for DMA channel 11 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX12 | R/W | 0x110 | Trigger select register for DMA channel 12 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX13 | R/W | 0x114 | Trigger select register for DMA channel 13 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX14 | R/W | 0x118 | Trigger select register for DMA channel 14 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX15 | R/W | 0x11C | Trigger select register for DMA channel 15 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX16 | R/W | 0x120 | Trigger select register for DMA channel 16 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX17 | R/W | 0x124 | Trigger select register for DMA channel 17 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX18 | R/W | 0x128 | Trigger select register for DMA channel 18 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX19 | R/W | 0x12C | Trigger select register for DMA channel 19 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX20 | R/W | 0x130 | Trigger select register for DMA channel 20 | 0x1F | Table 128 |
| DMA_ITRIG_INMUX21 | R/W | 0x134 | Trigger select register for DMA channel 21 | 0x1F | Table 128 |
| DMA_OTRIG_INMUX0 | R/W | 0x140 | DMA output trigger selection to become DMA trigger 16 | 0x1F | Table 130 |
| DMA_OTRIG_INMUX1 | R/W | 0x144 | DMA output trigger selection to become DMA trigger 17 | 0x1F | Table 130 |
| DMA_OTRIG_INMUX2 | R/W | 0x148 | DMA output trigger selection to become DMA trigger 18 | 0x1F | Table 130 |

Table 124. Register overview: Input multiplexing (base address 0x4005 0000) ...continued

| Name | Access | Offset | Description | Reset value | Reference |
|------------------|--------|--------|--|-------------|---------------------------|
| DMA_OTRIG_INMUX3 | R/W | 0x14C | DMA output trigger selection to become DMA trigger 19 | 0x1F | Table 130 |
| FREQMEAS_REF | R/W | 0x160 | Clock selection for frequency measurement function reference clock | 0x1F | Table 131 |
| FREQMEAS_TARGET | R/W | 0x164 | Clock selection for frequency measurement function target clock | 0x1F | Table 132 |

9.6.1 Pin interrupt select registers

Each of these 8 registers selects one pin from among ports 0 and 1 as the source of a pin interrupt or as the input to the pattern match engine. To select a pin for any of the 8 pin interrupts or pattern match engine inputs, write the GPIO port pin number as 0 to 31 for pins PIO0_0 to PIO0_31 to the INTPIN bits. Port 1 pins correspond to pin numbers 32 to 63. For example, setting INTPIN to 0x5 in PINTSEL0 selects pin PIO0_5 for pin interrupt 0. To determine the GPIO port pin number for a given device package, see the pin description table in the data sheet.

Each of the pin interrupts must be enabled in the NVIC (see [Table 2](#)) before it becomes active.

To use the selected pins for pin interrupts or the pattern match engine, see [Section 11.5.2 “Pattern match engine”](#).

Table 125. Address map PINTSEL[0:7] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| INPUTMUX | 0x4005 0000 | [0x0C0:0x0DC] | 0x4 | 8 |

Table 126. Pin interrupt select registers (PINTSEL[0:7], address offsets [0x0C0:0x0DC]) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 7:0 | INTPIN | Pin number select for pin interrupt or pattern match engine input. (PIO0_0 to PIO1_31 correspond to numbers 0 to 63). | 0 |
| 31:8 | - | Reserved | - |

9.6.2 DMA trigger input mux registers 0 to 21

With the DMA trigger input mux registers, one trigger input can be selected for each of the DMA channels from the potential internal sources. By default, none of the triggers are selected.

Table 127. Address map DMA_ITRIG_INMUX[0:21] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| INPUTMUX | 0x4005 0000 | [0x0E0:0x134] | 0x4 | 22 |

Table 128. DMA trigger Input mux registers (DMA_ITRIG_INMUX[0:21], address offsets [0x0E0:0x134]) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 4:0 | INP | Trigger input number (decimal value) for DMA channel n (n = 0 to 21). 0 = ADC0 Sequence A interrupt 1 = ADC0 Sequence B interrupt 2 = SCT0 DMA request 0 3 = SCT0 DMA request 1 4 = Timer CT32B0 Match 0 5 = Timer CT32B0 Match 1 6 = Timer CT32B1 Match 0 7 = Timer CT32B2 Match 0 8 = Timer CT32B2 Match 1 9 = Timer CT32B3 Match 0 10 = Timer CT32B4 Match 0 11 = Timer CT32B4 Match 1 12 = Pin interrupt 0 13 = Pin interrupt 1 14 = Pin interrupt 2 15 = Pin interrupt 3 16 = DMA output trigger mux 0 17 = DMA output trigger mux 1 18 = DMA output trigger mux 2 19 = DMA output trigger mux 3 | 0x1F |
| 31:5 | - | Reserved. | - |

9.6.3 DMA output trigger feedback mux registers 0 to 3

This register provides a multiplexer for inputs 16 to 19 of each DMA trigger input mux register DMA_ITRIG_INMUX. These inputs can be selected from among the trigger outputs generated by the each DMA channel. By default, none of the triggers are selected.

Table 129. Address map DMA_OTRIG_INMUX[0:3] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| INPUTMUX | 0x4005 0000 | [0x140:0x14C] | 0x4 | 4 |

Table 130. DMA output trigger feedback mux registers (DMA_OTRIG_INMUX[0:3], address offset [0x140:0x14C]) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|--|-------------|
| 4:0 | INP | DMA trigger output number (decimal value) for DMA channel n (n = 0 to 19). | 0x1F |
| 31:5 | - | Reserved. | - |

9.6.4 Frequency measure function reference clock select register

This register selects a clock for the reference clock of the frequency measure function. By default, no clock is selected. See [Section 4.6.5 “Frequency measure function”](#), [Section 4.2.3 “Measure the frequency of a clock signal”](#), [Section 4.5.32 “Frequency measure function control register”](#), and [Section 9.6.5](#) below for more on this function.

Table 131. Frequency measure function frequency clock select register (FREQMEAS_REF, address 0x4005 0160) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|--|-------------|
| 4:0 | CLKIN | Clock source number (decimal value) for frequency measure function target clock: 0 = CLK_IN 1 = IRC oscillator 2 = Watchdog oscillator 3 = 32 kHz RTC oscillator 4 = Main clock (see Section 4.5.17) 5 = PIO0_4 6 = PIO0_20 7 = PIO0_24 8 = PIO1_4 | 0x1F |
| 31:5 | - | Reserved. | - |

9.6.5 Frequency measure function target clock select register

This register selects a clock for the target clock of the frequency measure function. By default, no clock is selected. See [Section 4.6.5 “Frequency measure function”](#), [Section 4.2.3 “Measure the frequency of a clock signal”](#), [Section 4.5.32 “Frequency measure function control register”](#), and [Section 9.6.4](#) above for more on this function.

Table 132. Frequency measure function target clock select register (FREQMEAS_TARGET, address 0x4005 0164) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|--|-------------|
| 4:0 | CLKIN | Clock source number (decimal value) for frequency measure function target clock: 0 = CLK_IN 1 = IRC oscillator 2 = Watchdog oscillator 3 = 32 kHz RTC oscillator 4 = Main clock (see Section 4.5.17) 5 = PIO0_4 6 = PIO0_20 7 = PIO0_24 8 = PIO1_4 | 0x1F |
| 31:5 | - | Reserved. | - |

10.1 How to read this chapter

GPIO registers support up to 32 pins on each port. Depending on the device and package type, a subset of those pins may be available, and the unused bits in GPIO registers are reserved (see [Table 133](#)).

Table 133. GPIO pins available

| Package | Total GPIOs | GPIO Port 0 | GPIO Port 1 |
|-----------------------------------|-------------|-------------------------------------|-------------------|
| 64-pin device with RTC oscillator | 48 | PIO0_0 to PIO0_1, PIO0_4 to POIO_31 | PIO1_0 to PIO1_17 |
| 49-pin device with RTC oscillator | 39 | PIO0_0 to PIO0_1, PIO0_4 to POIO_31 | PIO1_0 to PIO1_8 |

10.2 Basic configuration

For the GPIO port registers, enable the clock to each GPIO port in the AHBCLKCTRL0 register ([Table 51](#)).

10.3 Features

- GPIO pins can be configured as input or output by software.
- All GPIO pins default to inputs with interrupt disabled at reset.
- Pin registers allow pins to be sensed and set individually.

10.4 General description

The GPIO pins can be used in several ways to set pins as inputs or outputs and use the inputs as combinations of level and edge sensitive interrupts.

The GPIOs can be used as external interrupts together with the pin interrupt and group interrupt blocks, see [Chapter 11](#) and [Chapter 12](#).

The GPIO port registers configure each GPIO pin as input or output and read the state of each pin if the pin is configured as input or set the state of each pin if the pin is configured as output.

10.5 Register description

Note: In all GPIO registers, bits that are not shown are **reserved**.

GPIO port addresses can be read and written as bytes, halfwords, or words.

Remark: A reset value noted as “ext” in this table and subsequent tables indicates that the data read after reset depends on the state of the pin, which in turn may depend on an external source.

Table 134. Register overview: GPIO port (base address 0x1C00 0000)

| Name | Access | Offset | Description | Reset value | Width | Related section |
|---------|--------|-----------------|--|-------------|---------------|------------------------|
| B[0:49] | R/W | [0x0000:0x0031] | Byte pin registers for all port 0 and 1 GPIO pins. | ext | byte (8 bit) | 10.5.1 |
| W[0:49] | R/W | [0x1000:0x10C4] | Word pin registers for all port 0 and 1 GPIO pins. | ext | word (32 bit) | 10.5.2 |
| DIR0 | R/W | 0x2000 | Direction registers port 0 | 0 | word (32 bit) | 10.5.3 |
| DIR1 | R/W | 0x2004 | Direction registers port 1 | 0 | word (32 bit) | 10.5.3 |
| MASK0 | R/W | 0x2080 | Mask register port 0 | 0 | word (32 bit) | 10.5.4 |
| MASK1 | R/W | 0x2084 | Mask register port 1 | 0 | word (32 bit) | 10.5.4 |
| PIN0 | R/W | 0x2100 | Port pin register port 0 | ext | word (32 bit) | 10.5.5 |
| PIN1 | R/W | 0x2104 | Port pin register port 1 | ext | word (32 bit) | 10.5.5 |
| MPIN0 | R/W | 0x2180 | Masked port register port 0 | ext | word (32 bit) | 10.5.6 |
| MPIN1 | R/W | 0x2184 | Masked port register port 1 | ext | word (32 bit) | 10.5.6 |
| SET0 | R/W | 0x2200 | Write: Set register for port 0 Read: output bits for port 0 | 0 | word (32 bit) | 10.5.7 |
| SET1 | R/W | 0x2204 | Write: Set register for port 1 Read: output bits for port 1 | 0 | word (32 bit) | 10.5.7 |
| CLR0 | WO | 0x2280 | Clear port 0 | NA | word (32 bit) | 10.5.8 |
| CLR1 | WO | 0x2284 | Clear port 1 | NA | word (32 bit) | 10.5.8 |
| NOT0 | WO | 0x2300 | Toggle port 0 | NA | word (32 bit) | 10.5.9 |
| NOT1 | WO | 0x2304 | Toggle port 1 | NA | word (32 bit) | 10.5.9 |

10.5.1 GPIO port byte pin registers

Each GPIO pin has a byte register in this address range. Software typically reads and writes bytes to access individual pins, but can read or write halfwords to sense or set the state of two pins, and read or write words to sense or set the state of four pins.

Table 135. Address map B[0:49] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| GPIO | 0x1C00 0000 | [0x000:0x031] | 0x1 | 50 |

Table 136. GPIO port byte pin registers (B[0:49], address offset [0x0000:0x0031]) bit description

| Bit | Symbol | Description | Reset value | Access |
|-----|--------|--|-------------|--------|
| 0 | PBYTE | Read: state of the pin P _{IOm_n} , regardless of direction, masking, or alternate function, except that pins configured as analog I/O always read as 0. One register for each port pin. Supported pins depends on the specific device and package. Write: loads the pin's output bit. Remark: One register for each port pin. Supported pins depends on the specific device and package. | ext | R/W |
| 7:1 | | Reserved (0 on read, ignored on write) | 0 | - |

10.5.2 GPIO port word pin registers

Each GPIO pin has a word register in this address range. Any byte, halfword, or word read in this range will be all zeros if the pin is low or all ones if the pin is high, regardless of direction, masking, or alternate function, except that pins configured as analog I/O always read as zeros. Any write will clear the pin's output bit if the value written is all zeros, else it will set the pin's output bit.

Table 137. Address map W[0:49] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| GPIO | 0x1C00 0000 | [0x1000:0x10C4] | 0x4 | 50 |

Table 138. GPIO port word pin registers (W[0:49], address offsets [0x1000:0x10C4]) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|--------|--|-------------|--------|
| 31:0 | PWORD | Read 0: pin P _{IOm_n} is LOW. Write 0: clear output bit. Read 0xFFFF FFFF: pin P _{IOm_n} is HIGH. Write any value 0x0000 0001 to 0xFFFF FFFF: set output bit. Remark: Only 0 or 0xFFFF FFFF can be read. Writing any value other than 0 will set the output bit. One register for each port pin. Supported pins depends on the specific device and package. | ext | R/W |

10.5.3 GPIO port direction registers

Each GPIO port has one direction register for configuring the port pins as inputs or outputs.

Table 139. Address map DIR[0:1] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| GPIO | 0x1C00 0000 | [0x2000:0x2004] | 0x4 | 2 |

Table 140. GPIO direction port register (DIR[0:1], address offset [0x2000:0x2004]) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|--------|--|-------------|--------|
| 31:0 | DIRP | Selects pin direction for pin P _{IOm_n} . Supported pins depends on the specific device and package. 0 = input. 1 = output. | 0 | R/W |

10.5.4 GPIO port mask registers

These registers affect writing and reading the MPORT registers. Zeroes in these registers enable reading and writing; ones disable writing and result in zeros in corresponding positions when reading.

Table 141. Address map MASK[0:1] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| GPIO | 0x1C00 0000 | [0x2080:0x2084] | 0x4 | 2 |

Table 142. GPIO mask port register (MASK[0:1], address offset [0x2080:0x2084]) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|--------|--|-------------|--------|
| 31:0 | MASKP | Controls which bits corresponding to P _{IOm_n} are active in the MPORT register. Supported pins depends on the specific device and package. 0 = Read MPORT: pin state; write MPORT: load output bit. 1 = Read MPORT: 0; write MPORT: output bit not affected. | 0 | R/W |

10.5.5 GPIO port pin registers

Reading these registers returns the current state of the pins read, regardless of direction, masking, or alternate functions, except that pins configured as analog I/O always read as 0s. Writing these registers loads the output bits of the pins written to, regardless of the Mask register.

Table 143. Address map PIN[0:1] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| GPIO | 0x1C00 0000 | [0x2100:0x2104] | 0x4 | 2 |

Table 144. GPIO port pin register (PIN[0:1], address offset [0x2100:0x2104]) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|--------|--|-------------|--------|
| 31:0 | PORT | Reads pin states or loads output bits. Supported pins depends on the specific device and package. 0 = Read: pin is low; write: clear output bit. 1 = Read: pin is high; write: set output bit. | ext | R/W |

10.5.6 GPIO masked port pin registers

These registers are similar to the PORT registers, except that the value read is masked by ANDing with the inverted contents of the corresponding MASK register, and writing to one of these registers only affects output register bits that are enabled by zeros in the corresponding MASK register

Table 145. Address map MPIN[0:1] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| GPIO | 0x1C00 0000 | [0x2180:0x2184] | 0x4 | 2 |

Table 146. GPIO masked port pin register (MPIN[0:1], address offset [0x2180:0x2184]) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|---------|--|-------------|--------|
| 31:0 | MPORPTP | Masked port register. Supported pins depends on the specific device and package. 0 = Read: pin is LOW and/or the corresponding bit in the MASK register is 1; write: clear output bit if the corresponding bit in the MASK register is 0. 1 = Read: pin is HIGH and the corresponding bit in the MASK register is 0; write: set output bit if the corresponding bit in the MASK register is 0. | ext | R/W |

10.5.7 GPIO port set registers

Output bits can be set by writing ones to these registers, regardless of MASK registers. Reading from these register returns the port's output bits, regardless of pin directions.

Table 147. Address map SET[0:1] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| GPIO | 0x1C00 0000 | [0x2200:0x2204] | 0x4 | 2 |

Table 148. GPIO set port register (SET[0:1], address offset [0x2200:0x2204]) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|--------|---|-------------|--------|
| 31:0 | SETP | Read or set output bits. Supported pins depends on the specific device and package. 0 = Read: output bit; write: no operation. 1 = Read: output bit; write: set output bit. | 0 | R/W |

10.5.8 GPIO port clear registers

Output bits can be cleared by writing ones to these write-only registers, regardless of MASK registers.

Table 149. Address map CLR[0:1] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| GPIO | 0x1C00 0000 | [0x2280:0x2284] | 0x4 | 2 |

Table 150. GPIO clear port register (CLR[0:1], address offset [0x2280:0x2284]) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|--------|---|-------------|--------|
| 31:0 | CLRP | Clear output bits. Supported pins depends on the specific device and package. 0 = No operation. 1 = Clear output bit. | NA | WO |

10.5.9 GPIO port toggle registers

Output bits can be toggled/inverted/complemented by writing ones to these write-only registers, regardless of MASK registers.

Table 151. Address map NOT[0:1] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| GPIO | 0x1C00 0000 | [0x2300:0x2304] | 0x4 | 2 |

Table 152. GPIO toggle port register (NOT[0:1], address offset [0x2300:0x2304]) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|--------|---|-------------|--------|
| 31:0 | NOTP | Toggle output bits. Supported pins depends on the specific device and package. 0 = no operation. 1 = Toggle output bit. | NA | WO |

10.6 Functional description

10.6.1 Reading pin state

Software can read the state of all GPIO pins except those selected for analog input or output in the “I/O Configuration” logic. A pin does not have to be selected for GPIO in “I/O Configuration” in order to read its state. There are four ways to read pin state:

- The state of a single pin can be read with 7 high-order zeros from a Byte Pin register.
- The state of a single pin can be read in all bits of a byte, halfword, or word from a Word Pin register.
- The state of multiple pins in a port can be read as a byte, halfword, or word from a PORT register.
- The state of a selected subset of the pins in a port can be read from a Masked Port (MPORT) register. Pins having a 1 in the port’s Mask register will read as 0 from its MPORT register.

10.6.2 GPIO output

Each GPIO pin has an output bit in the GPIO block. These output bits are the targets of write operations to the pins. Two conditions must be met in order for a pin’s output bit to be driven onto the pin:

1. The pin must be selected for GPIO operation via IOCON (this is the default), and
2. the pin must be selected for output by a 1 in its port’s DIR register.

If either or both of these conditions is (are) not met, writing to the pin has no effect.

There are seven ways to change GPIO output bits:

- Writing to a Byte Pin register loads the output bit from the least significant bit.
- Writing to a Word Pin register loads the output bit with the OR of all of the bits written. (This feature follows the definition of truth of a multi-bit value in programming languages.)
- Writing to a port’s PORT register loads the output bits of all the pins written to.
- Writing to a port’s MPORT register loads the output bits of pins identified by zeros in corresponding positions of the port’s MASK register.
- Writing ones to a port’s SET register sets output bits.
- Writing ones to a port’s CLR register clears output bits.
- Writing ones to a port’s NOT register toggles/complements/inverts output bits.

The state of a port’s output bits can be read from its SET register. Reading any of the registers described in [10.6.1](#) returns the state of pins, regardless of their direction or alternate functions.

10.6.3 Masked I/O

A port's MASK register defines which of its pins should be accessible in its MPORT register. Zeroes in MASK enable the corresponding pins to be read from and written to MPORT. Ones in MASK force a pin to read as 0 and its output bit to be unaffected by writes to MPORT. When a port's MASK register contains all zeros, its PORT and MPORT registers operate identically for reading and writing.

Applications in which interrupts can result in Masked GPIO operation, or in task switching among tasks that do Masked GPIO operation, must treat code that uses the Mask register as a protected/restricted region. This can be done by interrupt disabling or by using a semaphore.

The simpler way to protect a block of code that uses a MASK register is to disable interrupts before setting the MASK register, and re-enable them after the last operation that uses the MPORT or MASK register.

More efficiently, software can dedicate a semaphore to the MASK registers, and set/capture the semaphore controlling exclusive use of the MASK registers before setting the MASK registers, and release the semaphore after the last operation that uses the MPORT or MASK registers.

10.6.4 Recommended practices

The following lists some recommended uses for using the GPIO port registers:

- For initial setup after Reset or re-initialization, write the PORT registers.
- To change the state of one pin, write a Byte Pin or Word Pin register.
- To change the state of multiple pins at a time, write the SET and/or CLR registers.
- To change the state of multiple pins in a tightly controlled environment like a software state machine, consider using the NOT register. This can require less write operations than SET and CLR.
- To read the state of one pin, read a Byte Pin or Word Pin register.
- To make a decision based on multiple pins, read and mask a PORT register.

11.1 How to read this chapter

The pin interrupt generator and the pattern match engine are available on all LPC5410x parts.

11.2 Features

- Pin interrupts
 - Up to eight pins can be selected from all GPIO pins on ports 0 and 1 as edge- or level-sensitive interrupt requests. Each request creates a separate interrupt in the NVIC.
 - Edge-sensitive interrupt pins can interrupt on rising or falling edges or both.
 - Level-sensitive interrupt pins can be HIGH- or LOW-active.
- Pattern match engine
 - Up to 8 pins can be selected from all digital pins on ports 0 and 1 to contribute to a boolean expression. The boolean expression consists of specified levels and/or transitions on various combinations of these pins.
 - Each bit slice minterm (product term) comprising the specified boolean expression can generate its own, dedicated interrupt request.
 - Any occurrence of a pattern match can be programmed to also generate an RXEV notification to the CPU. The RXEV signal can be connected to a pin.
 - Pattern match can be used, in conjunction with software, to create complex state machines based on pin inputs.

11.3 Basic configuration

- Pin interrupts:
 - Select up to eight external interrupt pins from all digital port pins on ports 0 and 1 in the Input Mux block ([Table 126](#)). The pin selection process is the same for pin interrupts and the pattern match engine. The two features are mutually exclusive. Enable the clock to the pin interrupt register block in the AHBCLKCTRL0 register ([Table 51](#)).
 - In order to use the pin interrupts to wake up the part from deep-sleep mode or power-down mode, enable the pin interrupt wake-up feature in the STARTER0 register for pin interrupt 0 through 3 and the STARTER1 register for pin interrupt 4 through 7 ([Table 75](#) and [Table 76](#) respectively).
 - Each selected pin interrupt is assigned to one interrupt in the NVIC (interrupts #5 through #8 for pin interrupts 0 to 3, and 33 through 36 for pin interrupts 4 through 7). Note that only the first 4 are available to the Cortex M0+, which is present on LPC54102 devices.
- Pattern match engine:
 - Select up to eight external pins from all digital port pins on ports 0 and 1 in the Input mux block ([Table 126](#)). The pin selection process is the same for pin interrupts and the pattern match engine. The two features are mutually exclusive. Enable the clock to the pin interrupt register block in the AHBCLKCTRL0 register ([Table 51](#)).
 - Each bit slice of the pattern match engine is assigned to one interrupt in the NVIC (interrupts #5 through #8 for pin interrupts 0 to 3, and 33 through 36 for pin interrupts 4 through 7).

11.3.1 Configure pins as pin interrupts or as inputs to the pattern match engine

Follow these steps to configure pins as pin interrupts:

1. Determine the pins that serve as pin interrupts on the LPC5410x package. See the data sheet for determining the GPIO port pin number associated with the package pin.
2. For each pin interrupt, program the GPIO port pin number from ports 0 and 1 into one of the eight PINTSEL registers in the Input mux block.
Remark: The port pin number serves to identify the pin to the PINTSEL register. Any function, including GPIO, can be assigned to this pin via IOCON.
3. Enable each pin interrupt in the NVIC.

Once the pin interrupts or pattern match inputs are configured, the pin interrupt detection levels or the pattern match boolean expression can set up.

See [Section 9.6.1 “Pin interrupt select registers”](#) in the Input mux block for the PINTSEL registers.

Remark: The inputs to the Pin interrupt select registers bypass the IOCON function selection. They do not have to be selected as GPIO in IOCON. Make sure that no analog function is selected on pins that are input to the pin interrupts.

11.4 Pin description

The inputs to the pin interrupt and pattern match engine are determined by the pin interrupt select registers in the Input mux. See [Section 9.6.1 “Pin interrupt select registers”](#).

11.5 General description

Pins with configurable functions can serve as external interrupts or inputs to the pattern match engine. Up to eight pins can be configured using the PINTSEL registers in the Input mux block for these features.

11.5.1 Pin interrupts

From all available GPIO pins, up to eight pins can be selected in the system control block to serve as external interrupt pins (see [Table 126](#)). The external interrupt pins are connected to eight individual interrupts in the NVIC and are created based on rising or falling edges or on the input level on the pin.

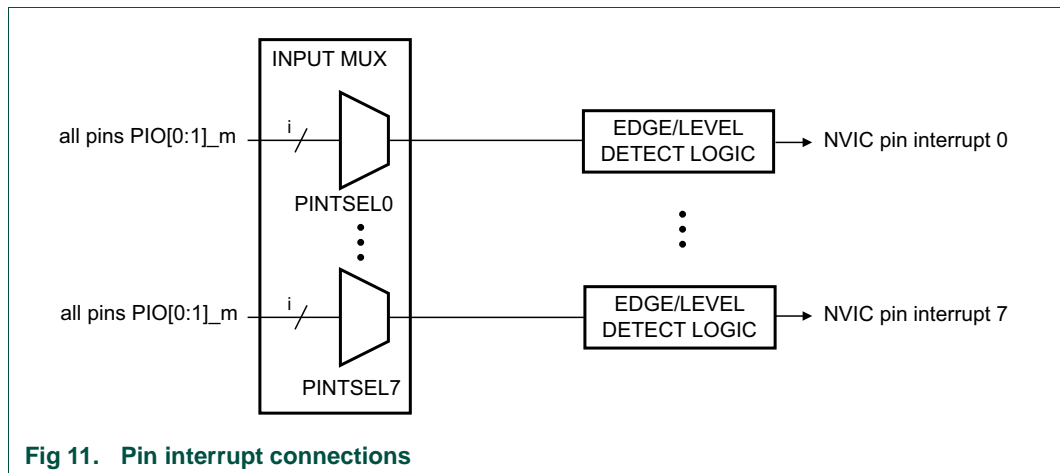


Fig 11. Pin interrupt connections

11.5.2 Pattern match engine

The pattern match feature allows complex boolean expressions to be constructed from the same set of eight GPIO pins that were selected for the GPIO pin interrupts. Each term in the boolean expression is implemented as one slice of the pattern match engine. A slice consists of an input selector and a detect logic that monitors the selected input continuously and creates a HIGH output if the input qualifies as detected, that is as true. Several terms can be combined to a minterm and a pin interrupt is asserted when the minterm evaluates as true.

The detect logic of each slice can detect the following events on the selected input:

- Edge with memory (sticky): A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism has been cleared. The input qualifies as detected (the detect logic output remains HIGH) until the pattern match engine detect logic is cleared again.

- Event (non-sticky): Every time an edge (rising or falling) is detected, the detect logic output for this pin goes HIGH. This bit is cleared after one clock cycle, and the detect logic can detect another edge,
- Level: A HIGH or LOW level on the selected input.

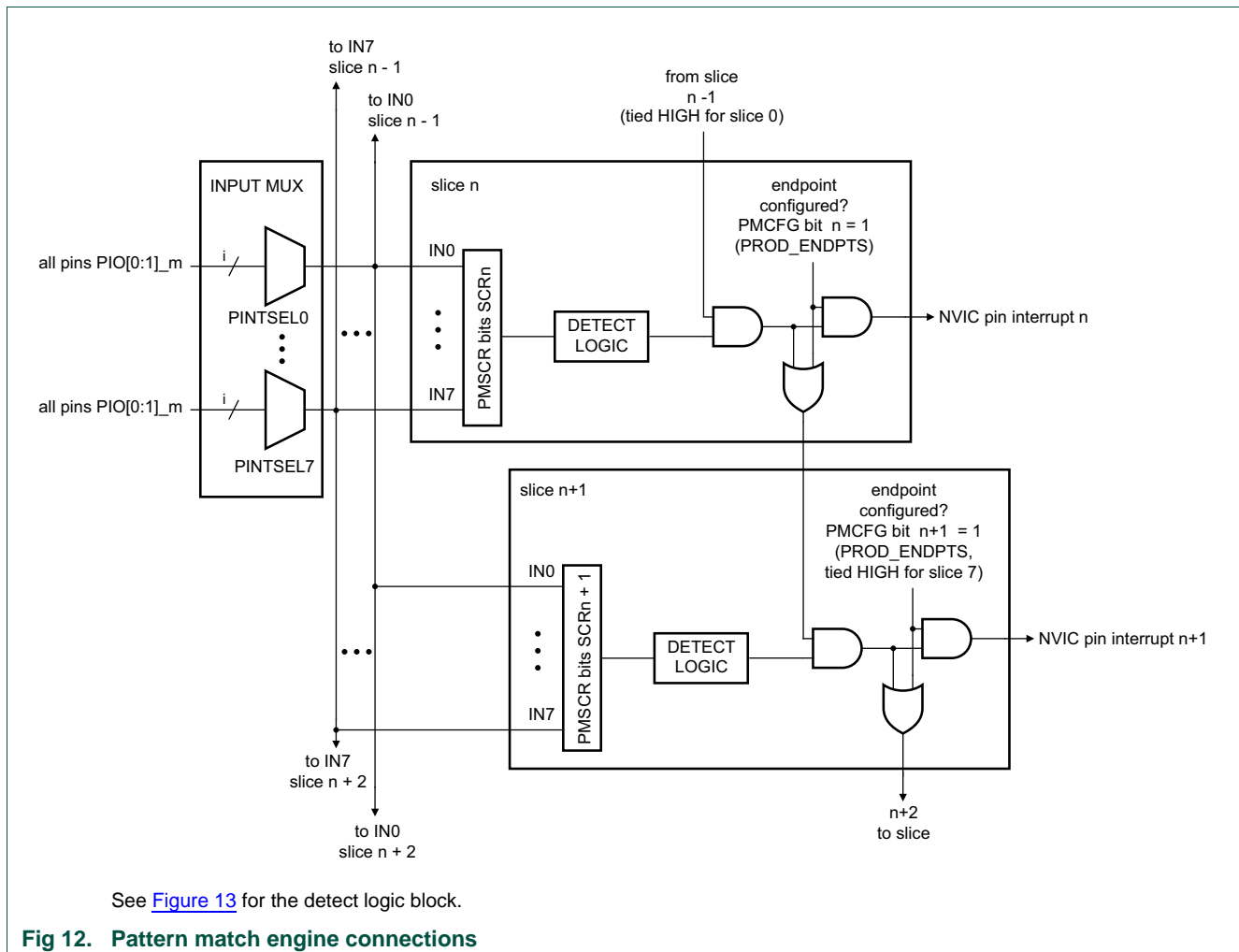
Figure 13 shows the details of the edge detection logic for each slice.

Sticky events can be combined with non-sticky events to create a pin interrupt whenever a rising or falling edge occurs after a qualifying edge event.

A time window can be created during which rising or falling edges can create a pin interrupt by combining a level detect with an event detect. See Section 11.7.3 for details.

The connections between the pins and the pattern match engine are shown in Figure 12. All pins that are inputs to the pattern match engine are selected in the Syscon block and can be GPIO port pins or other pin function depending on the IOCON configuration.

Remark: note that the pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during reduced power modes below Sleep mode.



The pattern match logic continuously monitors the eight inputs and generates interrupts when any one or more minterms (product terms) of the specified boolean expression is matched. A separate interrupt request is generated for each individual minterm.

In addition, the pattern match module can be enabled to generate a Receive Event (RXEV) output to the ARM core when the entire boolean expression is true (i.e. when any minterm is matched).

The pattern match function utilizes the same eight interrupt request lines as the pin interrupts so these two features are mutually exclusive as far as interrupt generation is concerned. A control bit is provided to select whether interrupt requests are generated in response to the standard pin interrupts or to pattern matches. Note that, if the pin interrupts are selected, the RXEV request to the CPU can still be enabled for pattern matches.

Remark: Pattern matching cannot be used to wake the part up from power-down modes. Pin interrupts must be selected in order to use the GPIO for wake-up.

The pattern match module is constructed of eight bit-slice elements. Each bit slice is programmed to represent one component of one minterm (product term) within the boolean expression. The interrupt request associated with the last bit slice for a particular minterm will be asserted whenever that minterm is matched. (See bit slice drawing [Figure 13](#)).

The pattern match capability can be used to create complex software state machines. Each minterm (and its corresponding individual interrupt) represents a different transition event to a new state. Software can then establish the new set of conditions (that is a new boolean expression) that will cause a transition out of the current state.

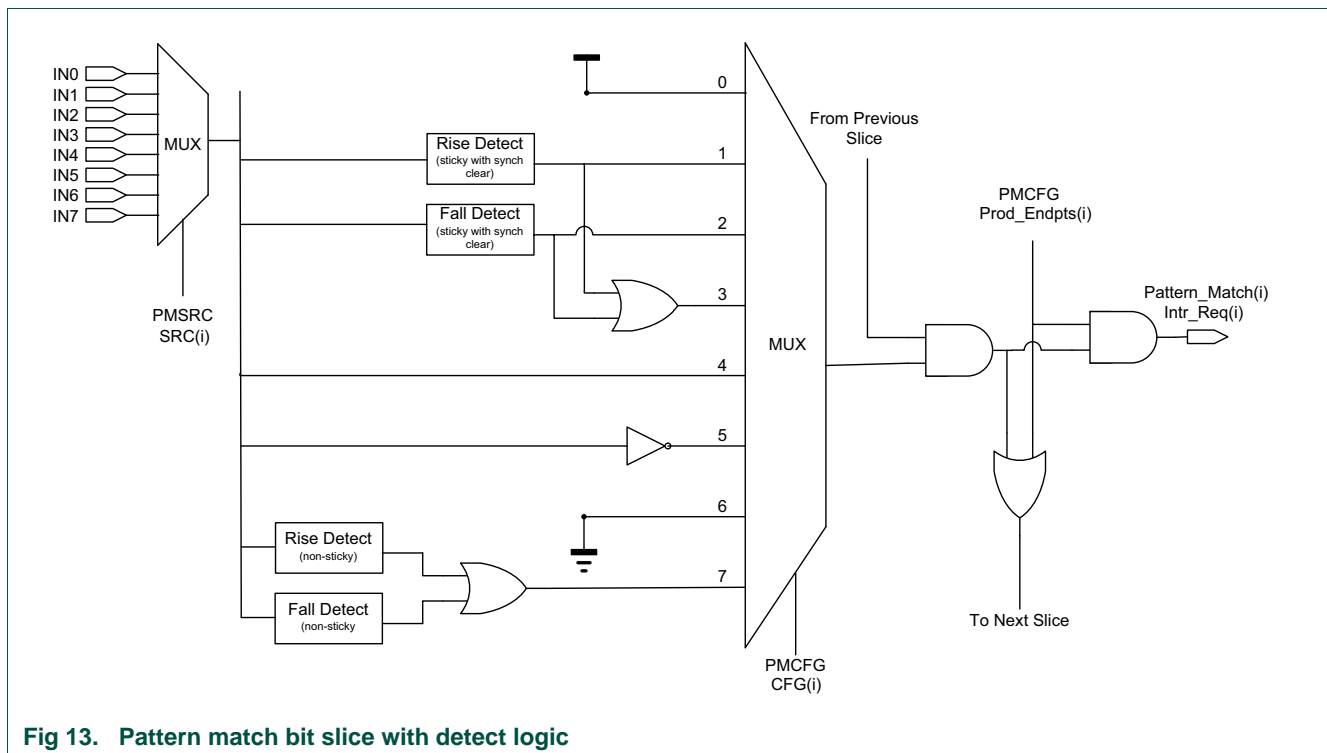


Fig 13. Pattern match bit slice with detect logic

11.5.2.1 Example

Assume the expression: $(IN0)\sim(IN1)(IN3)^{\wedge} + (IN1)(IN2) + (IN0)\sim(IN3)\sim(IN4)$ is specified through the registers PMSRC ([Table 165](#)) and PMCFG ([Table 166](#)). Each term in the boolean expression, $(IN0)$, $\sim(IN1)$, $(IN3)^{\wedge}$, etc., represents one bit slice of the pattern match engine.

- In the first minterm $(IN0)\sim(IN1)(IN3)^{\wedge}$, bit slice 0 monitors for a high-level on input $(IN0)$, bit slice 1 monitors for a low level on input $(IN1)$ and bit slice 2 monitors for a rising-edge on input $(IN3)$. If this combination is detected, that is if all three terms are true, the interrupt associated with bit slice 2 will be asserted.
- In the second minterm $(IN1)(IN2)$, bit slice 3 monitors input $(IN1)$ for a high level, bit slice 4 monitors input $(IN2)$ for a high level. If this combination is detected, the interrupt associated with bit slice 4 will be asserted.
- In the third minterm $(IN0)\sim(IN3)\sim(IN4)$, bit slice 5 monitors input $(IN0)$ for a high level, bit slice 6 monitors input $(IN3)$ for a low level, and bit slice 7 monitors input $(IN4)$ for a low level. If this combination is detected, the interrupt associated with bit slice 7 will be asserted.
- The ORed result of all three minterms asserts the RXEV request to the CPU and the GPIO_INT_BMAT output. That is, if any of the three terms are true, the output is asserted.

Related links:

[Section 11.7.2](#)

11.6 Register description

Table 153. Register overview: Pin interrupts/pattern match engine (base address: 0x4001 8000)

| Name | Access | Address offset | Description | Reset value | Reference |
|--------|--------|----------------|--|-------------|---------------------------|
| ISEL | R/W | 0x000 | Pin Interrupt Mode register | 0 | Table 154 |
| IENR | R/W | 0x004 | Pin interrupt level or rising edge interrupt enable register | 0 | Table 155 |
| SIENR | WO | 0x008 | Pin interrupt level or rising edge interrupt set register | NA | Table 156 |
| CIENR | WO | 0x00C | Pin interrupt level (rising edge interrupt) clear register | NA | Table 157 |
| IENF | R/W | 0x010 | Pin interrupt active level or falling edge interrupt enable register | 0 | Table 158 |
| SIENF | WO | 0x014 | Pin interrupt active level or falling edge interrupt set register | NA | Table 159 |
| CIENF | WO | 0x018 | Pin interrupt active level or falling edge interrupt clear register | NA | Table 160 |
| RISE | R/W | 0x01C | Pin interrupt rising edge register | 0 | Table 161 |
| FALL | R/W | 0x020 | Pin interrupt falling edge register | 0 | Table 162 |
| IST | R/W | 0x024 | Pin interrupt status register | 0 | Table 163 |
| PMCTRL | R/W | 0x028 | Pattern match interrupt control register | 0 | Table 164 |
| PMSRC | R/W | 0x02C | Pattern match interrupt bit-slice source register | 0 | Table 165 |
| PMCFG | R/W | 0x030 | Pattern match interrupt bit slice configuration register | 0 | Table 166 |

11.6.1 Pin interrupt mode register

For each of the 8 pin interrupts selected in the PINTSELn registers (see [Table 126](#)), one bit in the ISEL register determines whether the interrupt is edge or level sensitive.

Table 154. Pin interrupt mode register (ISEL, address 0x4001 8000) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|--------|--|-------------|--------|
| 7:0 | PMODE | Selects the interrupt mode for each pin interrupt. Bit n configures the pin interrupt selected in PINTSELn. 0 = Edge sensitive 1 = Level sensitive | 0 | R/W |
| 31:8 | - | Reserved. Read value is undefined, only zero should be written. | - | - |

11.6.2 Pin interrupt level or rising edge interrupt enable register

For each of the 8 pin interrupts selected in the PINTSELn registers (see [Table 126](#)), one bit in the IENR register enables the interrupt depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (PMODE = 0), the rising edge interrupt is enabled.
- If the pin interrupt mode is level sensitive (PMODE = 1), the level interrupt is enabled. The IENF register configures the active level (HIGH or LOW) for this interrupt.

Table 155. Pin interrupt level or rising edge interrupt enable register (IENR, address 0x4001 8004) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|--------|--|-------------|--------|
| 7:0 | ENRL | Enables the rising edge or level interrupt for each pin interrupt. Bit n configures the pin interrupt selected in PINTSELn. 0 = Disable rising edge or level interrupt. 1 = Enable rising edge or level interrupt. | 0 | R/W |
| 31:8 | - | Reserved. Read value is undefined, only zero should be written. | - | - |

11.6.3 Pin interrupt level or rising edge interrupt set register

For each of the 8 pin interrupts selected in the PINTSELn registers (see [Table 126](#)), one bit in the SIENR register sets the corresponding bit in the IENR register depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (PMODE = 0), the rising edge interrupt is set.
- If the pin interrupt mode is level sensitive (PMODE = 1), the level interrupt is set.

Table 156. Pin interrupt level or rising edge interrupt set register (SIENR, address 0x4001 8008) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|---------|--|-------------|--------|
| 7:0 | SETENRL | Ones written to this address set bits in the IENR, thus enabling interrupts. Bit n sets bit n in the IENR register. 0 = No operation. 1 = Enable rising edge or level interrupt. | NA | WO |
| 31:8 | - | Reserved. | - | - |

11.6.4 Pin interrupt level or rising edge interrupt clear register

For each of the 8 pin interrupts selected in the PINTSELn registers (see [Table 126](#)), one bit in the CIENR register clears the corresponding bit in the IENR register depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (PMODE = 0), the rising edge interrupt is cleared.
- If the pin interrupt mode is level sensitive (PMODE = 1), the level interrupt is cleared.

Table 157. Pin interrupt level or rising edge interrupt clear register (CIENR, address 0x4001 800C) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|--------|--|-------------|--------|
| 7:0 | CENRL | Ones written to this address clear bits in the IENR, thus disabling the interrupts. Bit n clears bit n in the IENR register. 0 = No operation. 1 = Disable rising edge or level interrupt. | NA | WO |
| 31:8 | - | Reserved. | - | - |

11.6.5 Pin interrupt active level or falling edge interrupt enable register

For each of the 8 pin interrupts selected in the PINTSELn registers (see [Table 126](#)), one bit in the IENF register enables the falling edge interrupt or the configures the level sensitivity depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (PMODE = 0), the falling edge interrupt is enabled.
- If the pin interrupt mode is level sensitive (PMODE = 1), the active level of the level interrupt (HIGH or LOW) is configured.

Table 158. Pin interrupt active level or falling edge interrupt enable register (IENF, address 0x4001 8010) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|--------|--|-------------|--------|
| 7:0 | ENAF | Enables the falling edge or configures the active level interrupt for each pin interrupt. Bit n configures the pin interrupt selected in PINTSELn. 0 = Disable falling edge interrupt or set active interrupt level LOW. 1 = Enable falling edge interrupt enabled or set active interrupt level HIGH. | 0 | R/W |
| 31:8 | - | Reserved. | - | - |

11.6.6 Pin interrupt active level or falling edge interrupt set register

For each of the 8 pin interrupts selected in the PINTSELn registers (see [Table 126](#)), one bit in the SIENF register sets the corresponding bit in the IENF register depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (PMODE = 0), the falling edge interrupt is set.
- If the pin interrupt mode is level sensitive (PMODE = 1), the HIGH-active interrupt is selected.

Table 159. Pin interrupt active level or falling edge interrupt set register (SIENF, address 0x4001 8014) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|---------|--|-------------|--------|
| 7:0 | SETENAF | Ones written to this address set bits in the IENF, thus enabling interrupts. Bit n sets bit n in the IENF register. 0 = No operation. 1 = Select HIGH-active interrupt or enable falling edge interrupt. | NA | WO |
| 31:8 | - | Reserved. | - | - |

11.6.7 Pin interrupt active level or falling edge interrupt clear register

For each of the 8 pin interrupts selected in the PINTSELn registers (see [Table 126](#)), one bit in the CIENF register sets the corresponding bit in the IENF register depending on the pin interrupt mode configured in the ISEL register:

- If the pin interrupt mode is edge sensitive (PMODE = 0), the falling edge interrupt is cleared.
- If the pin interrupt mode is level sensitive (PMODE = 1), the LOW-active interrupt is selected.

Table 160. Pin interrupt active level or falling edge interrupt clear register (CIENF, address 0x4001 8018) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|--------|---|-------------|--------|
| 7:0 | CENAF | Ones written to this address clears bits in the IENF, thus disabling interrupts. Bit n clears bit n in the IENF register. 0 = No operation. 1 = LOW-active interrupt selected or falling edge interrupt disabled. | NA | WO |
| 31:8 | - | Reserved. | - | - |

11.6.8 Pin interrupt rising edge register

This register contains ones for pin interrupts selected in the PINTSELn registers (see [Table 126](#)) on which a rising edge has been detected. Writing ones to this register clears rising edge detection. Ones in this register assert an interrupt request for pins that are enabled for rising-edge interrupts. All edges are detected for all pins selected by the PINTSELn registers, regardless of whether they are interrupt-enabled.

Table 161. Pin interrupt rising edge register (RISE, address 0x4001 801C) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|--------|---|-------------|--------|
| 7:0 | RDET | Rising edge detect. Bit n detects the rising edge of the pin selected in PINTSELn. Read 0: No rising edge has been detected on this pin since Reset or the last time a one was written to this bit. Write 0: no operation. Read 1: a rising edge has been detected since Reset or the last time a one was written to this bit. Write 1: clear rising edge detection for this pin. | 0 | R/W |
| 31:8 | - | Reserved. | - | - |

11.6.9 Pin interrupt falling edge register

This register contains ones for pin interrupts selected in the PINTSELn registers (see [Table 126](#)) on which a falling edge has been detected. Writing ones to this register clears falling edge detection. Ones in this register assert an interrupt request for pins that are enabled for falling-edge interrupts. All edges are detected for all pins selected by the PINTSELn registers, regardless of whether they are interrupt-enabled.

Table 162. Pin interrupt falling edge register (FALL, address 0x4001 8020) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|--------|--|-------------|--------|
| 7:0 | FDET | Falling edge detect. Bit n detects the falling edge of the pin selected in PINTSELn. Read 0: No falling edge has been detected on this pin since Reset or the last time a one was written to this bit. Write 0: no operation. Read 1: a falling edge has been detected since Reset or the last time a one was written to this bit. Write 1: clear falling edge detection for this pin. | 0 | R/W |
| 31:8 | - | Reserved. | - | - |

11.6.10 Pin interrupt status register

Reading this register returns ones for pin interrupts that are currently requesting an interrupt. For pins identified as edge-sensitive in the Interrupt Select register, writing ones to this register clears both rising- and falling-edge detection for the pin. For level-sensitive pins, writing ones inverts the corresponding bit in the Active level register, thus switching the active level on the pin.

Table 163. Pin interrupt status register (IST, address 0x4001 8024) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|--------|--|-------------|--------|
| 7:0 | PSTAT | Pin interrupt status. Bit n returns the status, clears the edge interrupt, or inverts the active level of the pin selected in PINTSELn. Read 0: interrupt is not being requested for this interrupt pin. Write 0: no operation. Read 1: interrupt is being requested for this interrupt pin. Write 1 (edge-sensitive): clear rising- and falling-edge detection for this pin. Write 1 (level-sensitive): switch the active level for this pin (in the IENF register). | 0 | R/W |
| 31:8 | - | Reserved. | - | - |

11.6.11 Pattern Match Interrupt Control Register

The pattern match control register contains one bit to select pattern-match interrupt generation (as opposed to pin interrupts which share the same interrupt request lines), and another to enable the RXEV output to the CPU. This register also allows the current state of any pattern matches to be read.

If the pattern match feature is not used (either for interrupt generation or for RXEV assertion) bits SEL_PMATCH and ENA_RXEV of this register should be left at 0 to conserve power.

Remark: Set up the pattern-match configuration in the PMSRC and PMCFG registers before writing to this register to enable (or re-enable) the pattern-match functionality. This eliminates the possibility of spurious interrupts as the feature is being enabled.

Remark: note that the pattern match feature requires clocks in order to operate, and can thus not generate an interrupt or wake up the device during reduced power modes below Sleep mode.

Table 164. Pattern match interrupt control register (PMCTRL, address 0x4001 8028) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----------|------------|-------|---|-------------|
| 0 | SEL_PMATCH | | Specifies whether the 8 pin interrupts are controlled by the pin interrupt function or by the pattern match function. | 0 |
| | | 0 | Pin interrupt. Interrupts are driven in response to the standard pin interrupt function. | |
| | | 1 | Pattern match. Interrupts are driven in response to pattern matches. | |
| 1 | ENA_RXEV | | Enables the RXEV output to the CPU and/or to a GPIO output when the specified boolean expression evaluates to true. | 0 |
| | | 0 | Disabled. RXEV output to the CPU is disabled. | |
| | | 1 | Enabled. RXEV output to the CPU is enabled. | |
| 23:2 | - | | Reserved. Do not write 1s to unused bits. | 0 |
| 31:2 4 | PMAT | - | This field displays the current state of pattern matches. A 1 in any bit of this field indicates that the corresponding product term is matched by the current state of the appropriate inputs. | 0x0 |

11.6.12 Pattern Match Interrupt Bit-Slice Source register

The bit-slice source register specifies the input source for each of the eight pattern match bit slices.

Each of the possible eight inputs is selected in the pin interrupt select registers in the SYSCON block. See [Section 9.6.1](#). Input 0 corresponds to the pin selected in the PINTSEL0 register, input 1 corresponds to the pin selected in the PINTSEL1 register, and so forth.

Remark: Writing any value to either the PMCFG register or the PMSRC register, or disabling the pattern-match feature (by clearing both the SEL_PMATCH and ENA_RXEV bits in the PMCTRL register to zeros) will erase all edge-detect history.

Table 165. Pattern match bit-slice source register (PMSRC, address 0x4001 802C) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|----------|-------|--|-------------|
| 7:0 | Reserved | | Software should not write 1s to unused bits. | 0 |
| 10:8 | SRC0 | | Selects the input source for bit slice 0 | 0 |
| | | 0x0 | Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 0. | |
| | | 0x1 | Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 0. | |
| | | 0x2 | Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 0. | |
| | | 0x3 | Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 0. | |
| | | 0x4 | Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 0. | |
| | | 0x5 | Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 0. | |
| | | 0x6 | Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 0. | |
| | | 0x7 | Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 0. | |

Table 165. Pattern match bit-slice source register (PMSRC, address 0x4001 802C) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|--|-------------|
| 13:11 | SRC1 | | Selects the input source for bit slice 1 | 0 |
| | | 0x0 | Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 1. | |
| | | 0x1 | Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 1. | |
| | | 0x2 | Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 1. | |
| | | 0x3 | Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 1. | |
| | | 0x4 | Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 1. | |
| | | 0x5 | Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 1. | |
| | | 0x6 | Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 1. | |
| | | 0x7 | Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 1. | |
| 16:14 | SRC2 | | Selects the input source for bit slice 2 | 0 |
| | | 0x0 | Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 2. | |
| | | 0x1 | Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 2. | |
| | | 0x2 | Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 2. | |
| | | 0x3 | Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 2. | |
| | | 0x4 | Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 2. | |
| | | 0x5 | Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 2. | |
| | | 0x6 | Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 2. | |
| | | 0x7 | Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 2. | |
| 19:17 | SRC3 | | Selects the input source for bit slice 3 | 0 |
| | | 0x0 | Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 3. | |
| | | 0x1 | Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 3. | |
| | | 0x2 | Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 3. | |
| | | 0x3 | Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 3. | |
| | | 0x4 | Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 3. | |
| | | 0x5 | Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 3. | |
| | | 0x6 | Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 3. | |
| | | 0x7 | Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 3. | |
| 22:20 | SRC4 | | Selects the input source for bit slice 4 | 0 |
| | | 0x0 | Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 4. | |
| | | 0x1 | Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 4. | |
| | | 0x2 | Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 4. | |
| | | 0x3 | Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 4. | |
| | | 0x4 | Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 4. | |
| | | 0x5 | Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 4. | |
| | | 0x6 | Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 4. | |
| | | 0x7 | Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 4. | |

Table 165. Pattern match bit-slice source register (PMSRC, address 0x4001 802C) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|--|-------------|
| 25:23 | SRC5 | | Selects the input source for bit slice 5 | 0 |
| | | 0x0 | Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 5. | |
| | | 0x1 | Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 5. | |
| | | 0x2 | Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 5. | |
| | | 0x3 | Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 5. | |
| | | 0x4 | Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 5. | |
| | | 0x5 | Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 5. | |
| | | 0x6 | Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 5. | |
| | | 0x7 | Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 5. | |
| 28:26 | SRC6 | | Selects the input source for bit slice 6 | 0 |
| | | 0x0 | Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 6. | |
| | | 0x1 | Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 6. | |
| | | 0x2 | Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 6. | |
| | | 0x3 | Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 6. | |
| | | 0x4 | Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 6. | |
| | | 0x5 | Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 6. | |
| | | 0x6 | Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 6. | |
| | | 0x7 | Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 6. | |
| 31:29 | SRC7 | | Selects the input source for bit slice 7 | 0 |
| | | 0x0 | Input 0. Selects the pin selected in the PINTSEL0 register as the source to bit slice 7. | |
| | | 0x1 | Input 1. Selects the pin selected in the PINTSEL1 register as the source to bit slice 7. | |
| | | 0x2 | Input 2. Selects the pin selected in the PINTSEL2 register as the source to bit slice 7. | |
| | | 0x3 | Input 3. Selects the pin selected in the PINTSEL3 register as the source to bit slice 7. | |
| | | 0x4 | Input 4. Selects the pin selected in the PINTSEL4 register as the source to bit slice 7. | |
| | | 0x5 | Input 5. Selects the pin selected in the PINTSEL5 register as the source to bit slice 7. | |
| | | 0x6 | Input 6. Selects the pin selected in the PINTSEL6 register as the source to bit slice 7. | |
| | | 0x7 | Input 7. Selects the pin selected in the PINTSEL7 register as the source to bit slice 7. | |

11.6.13 Pattern Match Interrupt Bit Slice Configuration register

The bit-slice configuration register configures the detect logic and contains bits to select from among eight alternative conditions for each bit slice that cause that bit slice to contribute to a pattern match. The seven LSBs of this register specify which bit-slices are the end-points of product terms in the boolean expression (i.e. where OR terms are to be inserted in the expression).

Two types of edge detection on each input are possible:

- **Sticky:** A rising edge, a falling edge, or a rising or falling edge that is detected at any time after the edge-detection mechanism has been cleared. The input qualifies as detected (the detect logic output remains HIGH) until the pattern match engine detect logic is cleared again.

- Non-sticky: Every time an edge (rising or falling) is detected, the detect logic output for this pin goes HIGH. This bit is cleared after one clock cycle, and the edge detect logic can detect another edge,

Remark: To clear the pattern match engine detect logic, write any value to either the PMCFG register or the PMSRC register, or disable the pattern-match feature (by clearing both the SEL_PMATCH and ENA_RXEV bits in the PMCTRL register to zeros). This will erase all edge-detect history.

To select whether a slice marks the final component in a minterm of the boolean expression, write a 1 in the corresponding PROD_ENPTS_n bit. Setting a term as the final component has two effects:

1. The interrupt request associated with this bit slice will be asserted whenever a match to that product term is detected.
2. The next bit slice will start a new, independent product term in the boolean expression (i.e. an OR will be inserted in the boolean expression following the element controlled by this bit slice).

Table 166. Pattern match bit slice configuration register (PMCFG, address 0x4001 8030) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|------------------|-------|--|-------------|
| 0 | PROD_EN DPTS0 | | Determines whether slice 0 is an endpoint. | 0 |
| | | 0 | No effect. Slice 0 is not an endpoint. | |
| | | 1 | endpoint. Slice 0 is the endpoint of a product term (minterm). Pin interrupt 0 in the NVIC is raised if the minterm evaluates as true. | |
| 1 | PROD_EN DPTS1 | | Determines whether slice 1 is an endpoint. | 0 |
| | | 0 | No effect. Slice 1 is not an endpoint. | |
| | | 1 | endpoint. Slice 1 is the endpoint of a product term (minterm). Pin interrupt 1 in the NVIC is raised if the minterm evaluates as true. | |
| 2 | PROD_EN DPTS2 | | Determines whether slice 2 is an endpoint. | 0 |
| | | 0 | No effect. Slice 2 is not an endpoint. | |
| | | 1 | endpoint. Slice 2 is the endpoint of a product term (minterm). Pin interrupt 2 in the NVIC is raised if the minterm evaluates as true. | |
| 3 | PROD_EN DPTS3 | | Determines whether slice 3 is an endpoint. | 0 |
| | | 0 | No effect. Slice 3 is not an endpoint. | |
| | | 1 | endpoint. Slice 3 is the endpoint of a product term (minterm). Pin interrupt 3 in the NVIC is raised if the minterm evaluates as true. | |
| 4 | PROD_EN DPTS4 | | Determines whether slice 4 is an endpoint. | 0 |
| | | 0 | No effect. Slice 4 is not an endpoint. | |
| | | 1 | endpoint. Slice 4 is the endpoint of a product term (minterm). Pin interrupt 4 in the NVIC is raised if the minterm evaluates as true. | |
| 5 | PROD_EN DPTS5 | | Determines whether slice 5 is an endpoint. | 0 |
| | | 0 | No effect. Slice 5 is not an endpoint. | |
| | | 1 | endpoint. Slice 5 is the endpoint of a product term (minterm). Pin interrupt 5 in the NVIC is raised if the minterm evaluates as true. | |
| 6 | PROD_EN DPTS6 | | Determines whether slice 6 is an endpoint. | 0 |
| | | 0 | No effect. Slice 6 is not an endpoint. | |
| | | 1 | endpoint. Slice 6 is the endpoint of a product term (minterm). Pin interrupt 6 in the NVIC is raised if the minterm evaluates as true. | |

Table 166. Pattern match bit slice configuration register (PMCFG, address 0x4001 8030) bit description ...continued

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|--|-------------|
| 7 | - | | Reserved. Bit slice 7 is automatically considered a product end point. | 0 |
| 10:8 | CFG0 | | Specifies the match contribution condition for bit slice 0. | 0b000 |
| | | 0x0 | Constant HIGH. This bit slice always contributes to a product term match. | |
| | | 0x1 | Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x2 | Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x3 | Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x4 | High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register. | |
| | | 0x5 | Low level. Match occurs when there is a low level on the specified input. | |
| | | 0x6 | Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices). | |
| | | 0x7 | Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle. | |
| 13:11 | CFG1 | | Specifies the match contribution condition for bit slice 1. | 0b000 |
| | | 0x0 | Constant HIGH. This bit slice always contributes to a product term match. | |
| | | 0x1 | Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x2 | Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x3 | Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x4 | High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register. | |
| | | 0x5 | Low level. Match occurs when there is a low level on the specified input. | |
| | | 0x6 | Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices). | |
| | | 0x7 | Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle. | |

Table 166. Pattern match bit slice configuration register (PMCFG, address 0x4001 8030) bit description ...continued

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|--|-------------|
| 16:14 | CFG2 | | Specifies the match contribution condition for bit slice 2. | 0b000 |
| | | 0x0 | Constant HIGH. This bit slice always contributes to a product term match. | |
| | | 0x1 | Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x2 | Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x3 | Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x4 | High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register. | |
| | | 0x5 | Low level. Match occurs when there is a low level on the specified input. | |
| | | 0x6 | Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices). | |
| | | 0x7 | Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle. | |
| 19:17 | CFG3 | | Specifies the match contribution condition for bit slice 3. | 0b000 |
| | | 0x0 | Constant HIGH. This bit slice always contributes to a product term match. | |
| | | 0x1 | Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x2 | Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x3 | Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x4 | High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register. | |
| | | 0x5 | Low level. Match occurs when there is a low level on the specified input. | |
| | | 0x6 | Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices). | |
| | | 0x7 | Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle. | |

Table 166. Pattern match bit slice configuration register (PMCFG, address 0x4001 8030) bit description ...continued

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|--|-------------|
| 22:20 | CFG4 | | Specifies the match contribution condition for bit slice 4. | 0b000 |
| | | 0x0 | Constant HIGH. This bit slice always contributes to a product term match. | |
| | | 0x1 | Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x2 | Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x3 | Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x4 | High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register. | |
| | | 0x5 | Low level. Match occurs when there is a low level on the specified input. | |
| | | 0x6 | Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices). | |
| | | 0x7 | Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle. | |
| 25:23 | CFG5 | | Specifies the match contribution condition for bit slice 5. | 0b000 |
| | | 0x0 | Constant HIGH. This bit slice always contributes to a product term match. | |
| | | 0x1 | Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x2 | Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x3 | Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x4 | High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register. | |
| | | 0x5 | Low level. Match occurs when there is a low level on the specified input. | |
| | | 0x6 | Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices). | |
| | | 0x7 | Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle. | |

Table 166. Pattern match bit slice configuration register (PMCFG, address 0x4001 8030) bit description ...continued

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|--|-------------|
| 28:26 | CFG6 | | Specifies the match contribution condition for bit slice 6. | 0b000 |
| | | 0x0 | Constant HIGH. This bit slice always contributes to a product term match. | |
| | | 0x1 | Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x2 | Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x3 | Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x4 | High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register. | |
| | | 0x5 | Low level. Match occurs when there is a low level on the specified input. | |
| | | 0x6 | Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices). | |
| 31:29 | CFG7 | | Specifies the match contribution condition for bit slice 7. | 0b000 |
| | | 0x0 | Constant HIGH. This bit slice always contributes to a product term match. | |
| | | 0x1 | Sticky rising edge. Match occurs if a rising edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x2 | Sticky falling edge. Match occurs if a falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x3 | Sticky rising or falling edge. Match occurs if either a rising or falling edge on the specified input has occurred since the last time the edge detection for this bit slice was cleared. This bit is only cleared when the PMCFG or the PMSRC registers are written to. | |
| | | 0x4 | High level. Match (for this bit slice) occurs when there is a high level on the input specified for this bit slice in the PMSRC register. | |
| | | 0x5 | Low level. Match occurs when there is a low level on the specified input. | |
| | | 0x6 | Constant 0. This bit slice never contributes to a match (should be used to disable any unused bit slices). | |
| | | 0x7 | Event. Non-sticky rising or falling edge. Match occurs on an event - i.e. when either a rising or falling edge is first detected on the specified input (this is a non-sticky version of value 0x3). This bit is cleared after one clock cycle. | |

11.7 Functional description

11.7.1 Pin interrupts

In this interrupt facility, up to 8 pins are identified as interrupt sources by the Pin Interrupt Select registers (PINTSEL0-7). All registers in the pin interrupt block contain 8 bits, corresponding to the pins called out by the PINTSEL0-7 registers. The ISEL register defines whether each interrupt pin is edge- or level-sensitive. The RISE and FALL registers detect edges on each interrupt pin, and can be written to clear (and set) edge detection. The IST register indicates whether each interrupt pin is currently requesting an interrupt, and this register can also be written to clear interrupts.

The other pin interrupt registers play different roles for edge-sensitive and level-sensitive pins, as described in [Table 167](#).

Table 167. Pin interrupt registers for edge- and level-sensitive pins

| Name | Edge-sensitive function | Level-sensitive function |
|-------|---|------------------------------------|
| IENR | Enables rising-edge interrupts. | Enables level interrupts. |
| SIENR | Write to enable rising-edge interrupts. | Write to enable level interrupts. |
| CIENR | Write to disable rising-edge interrupts. | Write to disable level interrupts. |
| IENF | Enables falling-edge interrupts. | Selects active level. |
| SIENF | Write to enable falling-edge interrupts. | Write to select high-active. |
| CIENF | Write to disable falling-edge interrupts. | Write to select low-active. |

11.7.2 Pattern Match engine example

Suppose the desired boolean pattern to be matched is:
 $(IN1) + (IN1 * IN2) + (\sim IN2 * \sim IN3 * IN6fe) + (IN5 * IN7ev)$

with:

IN6fe = (sticky) falling-edge on input 6

IN7ev = (non-sticky) event (rising or falling edge) on input 7

Each individual term in the expression shown above is controlled by one bit-slice. To specify this expression, program the pattern match bit slice source and configuration register fields as follows:

- PMSRC register ([Table 165](#)):
 - Since bit slice 5 will be used to detect a sticky event on input 6, a 1 can be written to the SRC5 bits to clear any pre-existing edge detects on bit slice 5.
 - SRC0: 001 - select input 1 for bit slice 0
 - SRC1: 001 - select input 1 for bit slice 1
 - SRC2: 010 - select input 2 for bit slice 2
 - SRC3: 010 - select input 2 for bit slice 3
 - SRC4: 011 - select input 3 for bit slice 4
 - SRC5: 110 - select input 6 for bit slice 5
 - SRC6: 101 - select input 5 for bit slice 6

- SRC7: 111 - select input 7 for bit slice 7
- PMCFG register ([Table 166](#)):
 - PROD_ENDPTS0 = 1
 - PROD_ENDPTS02 = 1
 - PROD_ENDPTS5 = 1
 - All other slices are not product term endpoints and their PROD_ENDPTS bits are 0. Slice 7 is always a product term endpoint and does not have a register bit associated with it.
 - = 0100101 - bit slices 0, 2, 5, and 7 are product-term endpoints. (Bit slice 7 is an endpoint by default - no associated register bit).
 - CFG0: 000 - high level on the selected input (input 1) for bit slice 0
 - CFG1: 000 - high level on the selected input (input 1) for bit slice 1
 - CFG2: 000 - high level on the selected input (input 2) for bit slice 2
 - CFG3: 101 - low level on the selected input (input 2) for bit slice 3
 - CFG4: 101 - low level on the selected input (input 3) for bit slice 4
 - CFG5: 010 - (sticky) falling edge on the selected input (input 6) for bit slice 5
 - CFG6: 000 - high level on the selected input (input 5) for bit slice 6
 - CFG7: 111 - event (any edge, non-sticky) on the selected input (input 7) for bit slice 7
- PMCTRL register ([Table 164](#)):
 - Bit0: Setting this bit will select pattern matches to generate the pin interrupts in place of the normal pin interrupt mechanism.

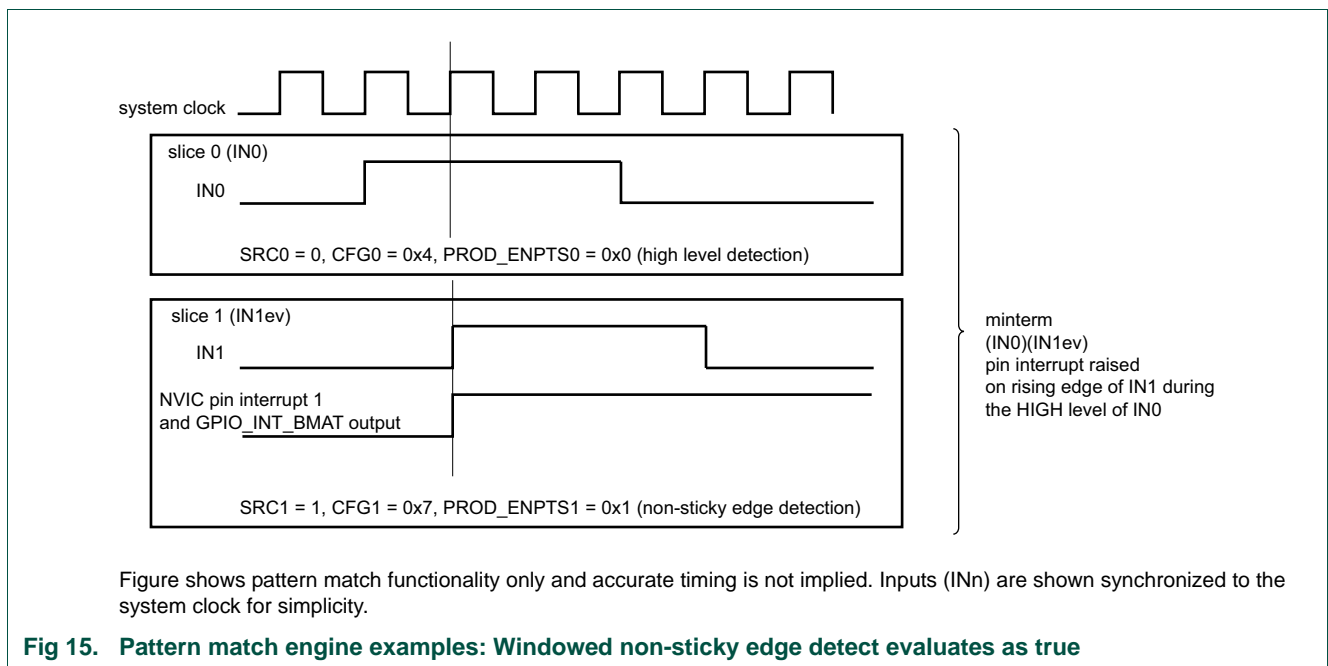
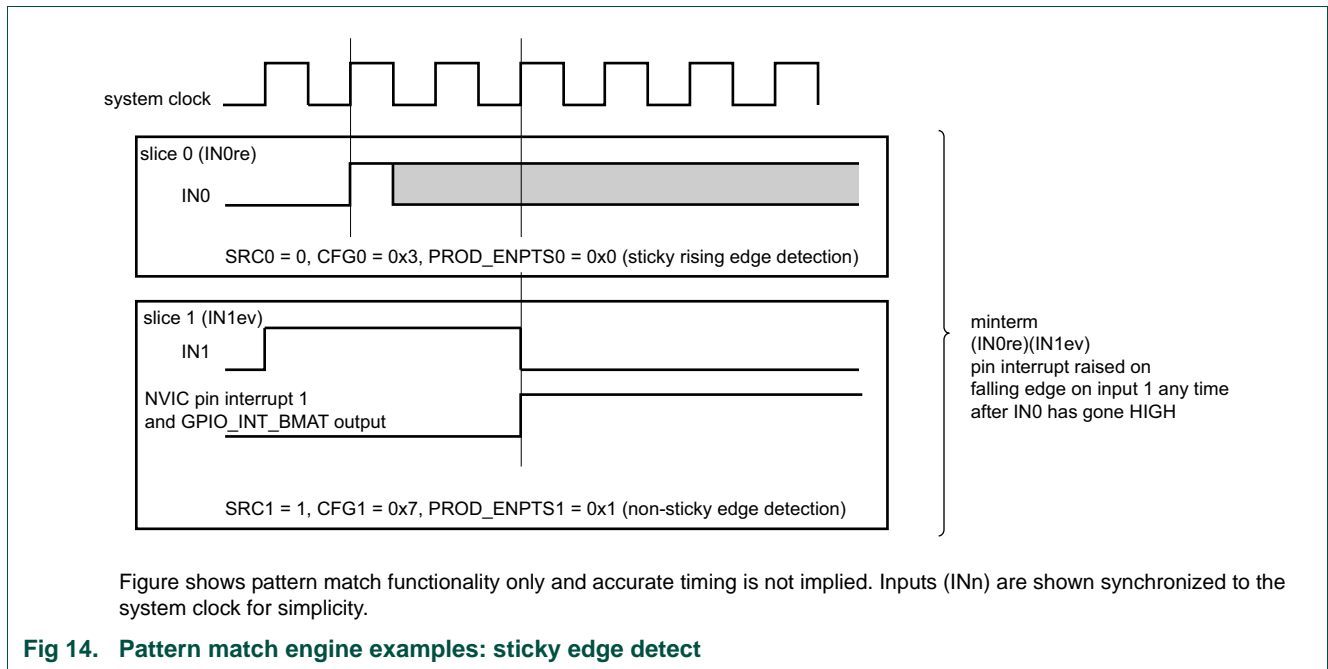
For this example, pin interrupt 0 will be asserted when a match is detected on the first product term (which, in this case, is just a high level on input 1).

Pin interrupt 2 will be asserted in response to a match on the second product term.

Pin interrupt 5 will be asserted when there is a match on the third product term.

Pin interrupt 7 will be asserted on a match on the last term.
 - Bit1: Setting this bit will cause the RxEv signal to the CPU to be asserted whenever a match occurs on ANY of the product terms in the expression. Otherwise, the RXEV line will not be used.
 - Bit31:24: At any given time, bits 0, 2, 5 and/or 7 may be high if the corresponding product terms are currently matching.
 - The remaining bits will always be low.

11.7.3 Pattern match engine edge detect examples



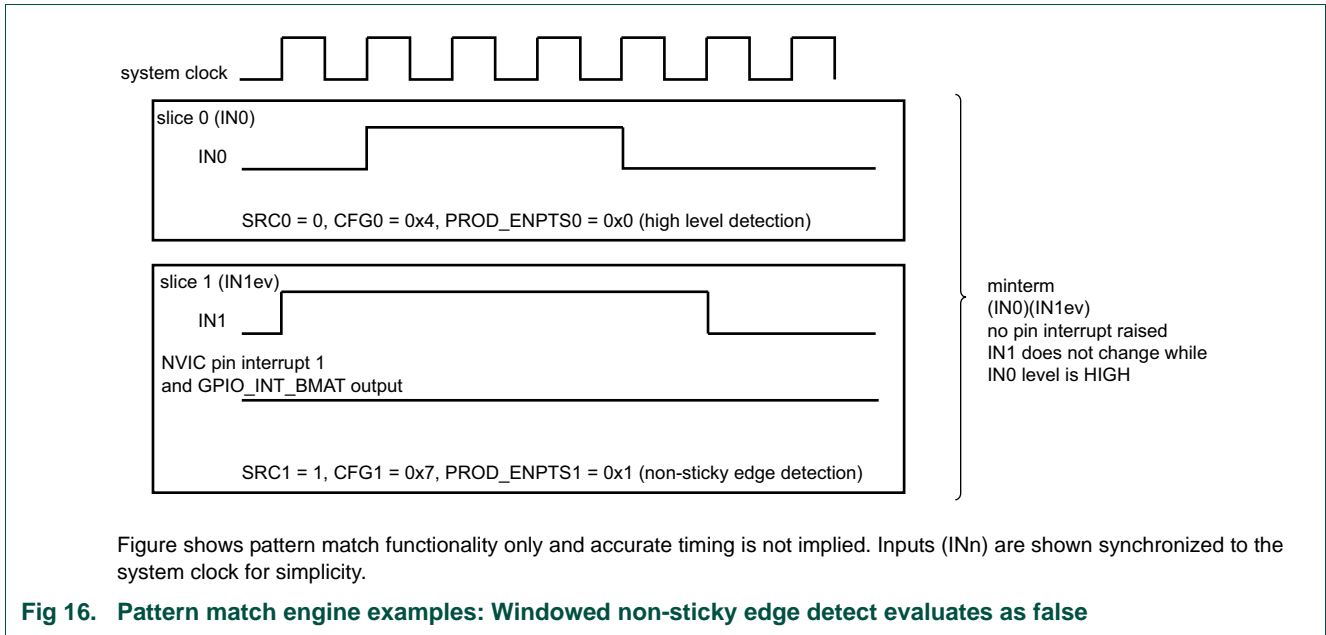


Fig 16. Pattern match engine examples: Windowed non-sticky edge detect evaluates as false

12.1 Features

- The inputs from any number of digital pins can be enabled to contribute to a combined group interrupt.
- The polarity of each input enabled for the group interrupt can be configured HIGH or LOW.
- Enabled interrupts can be logically combined through an OR or AND operation.
- Two group interrupts are supported to reflect two distinct interrupt patterns.
- The grouped interrupts can wake up the part from sleep, deep-sleep or power-down modes.

12.2 Basic configuration

For the group interrupt feature, enable the clock to both the GROUP0 and GROUP1 register interfaces in the AHBCLKCTRL0 register (([Table 51](#)). The group interrupt wake-up feature is enabled in the STARTER0 register for GINT0 and the STARTER1 register for GINT1 ([Table 75](#) and [Table 76](#) respectively).

The pins can be configured as GPIO pins through IOCON, but they don't have to be. The GINT block reads the input from the pin bypassing IOCON multiplexing. Make sure that no analog function is selected on pins that are input to the group interrupts. Selecting an analog function in IOCON disables the digital pad and the digital signal is tied to 0.

12.3 General description

The GPIO pins can be used in several ways to set pins as inputs or outputs and use the inputs as combinations of level and edge sensitive interrupts.

For each port/pin connected to one of the two the GPIO Grouped Interrupt blocks (GROUP0 and GROUP1), the GPIO grouped interrupt registers determine which pins are enabled to generate interrupts and what the active polarities of each of those inputs are.

The GPIO grouped interrupt registers also select whether the interrupt output will be level or edge triggered and whether it will be based on the OR or the AND of all of the enabled inputs.

When the designated pattern is detected on the selected input pins, the GPIO grouped interrupt block generates an interrupt. If the part is in a power-savings mode, it first asynchronously wakes the part up prior to asserting the interrupt request. The interrupt request line can be cleared by writing a one to the interrupt status bit in the control register.

12.4 Register description

Note: In all registers, bits that are not shown are **reserved**.

Table 168. Register overview: GROUP0 interrupt (base address 0x4001 0000 (GINT0) and 0x4001 4000 (GINT1))

| Name | Access | Address offset | Description | Reset value | Reference |
|-----------|--------|----------------|---|-------------|---------------------------|
| CTRL | R/W | 0x000 | GPIO grouped interrupt control register | 0 | Table 169 |
| PORT_POL0 | R/W | 0x020 | GPIO grouped interrupt port 0 polarity register | 0xFFFF FFFF | Table 170 |
| PORT_POL1 | R/W | 0x024 | GPIO grouped interrupt port 1 polarity register | 0xFFFF FFFF | Table 170 |
| PORT_ENA0 | R/W | 0x040 | GPIO grouped interrupt port 0 enable register | 0 | Table 171 |
| PORT_ENA1 | R/W | 0x044 | GPIO grouped interrupt port 1 enable register | 0 | Table 171 |

12.4.1 Grouped interrupt control register

Table 169. GPIO grouped interrupt control register (CTRL, addresses 0x4001 0000 (GINT0) and 0x4001 4000 (GINT1)) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---|-------------|
| 0 | INT | | Group interrupt status. This bit is cleared by writing a one to it. Writing zero has no effect. | 0 |
| | | 0 | No request. No interrupt request is pending. | |
| | | 1 | Request active. Interrupt request is active. | |
| 1 | COMB | | Combine enabled inputs for group interrupt | 0 |
| | | 0 | Or. OR functionality: A grouped interrupt is generated when any one of the enabled inputs is active (based on its programmed polarity). | |
| | | 1 | And. AND functionality: An interrupt is generated when all enabled bits are active (based on their programmed polarity). | |
| 2 | TRIG | | Group interrupt trigger | 0 |
| | | 0 | Edge-triggered. | |
| | | 1 | Level-triggered. | |
| 31:3 | - | - | Reserved. Read value is undefined, only zero should be written. | 0 |

12.4.2 GPIO grouped interrupt port polarity registers

The grouped interrupt port polarity registers determine how the polarity of each enabled pin contributes to the grouped interrupt. Each port is associated with its own port polarity register, and the values of both registers together determine the grouped interrupt.

Each register PORT_POL_m controls the polarity of pins in port m.

Table 170. GPIO grouped interrupt port polarity registers (PORT_POL[0:1], addresses 0x4001 0020 (PORT_POL0) to 0x4001 0024 (PORT_POL1) (GINT0) and 0x4001 4020 (PORT_POL0) to 0x4001 4024 (PORT_POL1) (GINT1)) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|--------|--|-------------|--------|
| 31:0 | POL | Configure pin polarity of port m pins for group interrupt. Bit n corresponds to pin PIO _m _n of port m. 0 = the pin is active LOW. If the level on this pin is LOW, the pin contributes to the group interrupt. 1 = the pin is active HIGH. If the level on this pin is HIGH, the pin contributes to the group interrupt. | 1 | - |

12.4.3 GPIO grouped interrupt port enable registers

The grouped interrupt port enable registers enable the pins which contribute to the grouped interrupt. Each port is associated with its own port enable register, and the values of both registers together determine which pins contribute to the grouped interrupt.

Each register PORT_EN_m enables pins in port m.

Table 171. GPIO grouped interrupt port enable registers (PORT_ENA[0:1], addresses 0x4001 0040 (PORT_ENA0) to 0x4001 0044 (PORT_ENA1) (GINT0) and 0x4001 4040 (PORT_ENA0) to 0x4001 4044 (PORT_ENA1) (GINT1)) bit description

| Bit | Symbol | Description | Reset value | Access |
|------|--------|--|-------------|--------|
| 31:0 | ENA | Enable port 0 pin for group interrupt. Bit n corresponds to pin Pm_n of port m. 0 = the port 0 pin is disabled and does not contribute to the grouped interrupt. 1 = the port 0 pin is enabled and contributes to the grouped interrupt. | 0 | - |

12.5 Functional description

Any subset of the pins in each port can be selected to contribute to a common group interrupt (GINT) and can be enabled to wake the part up from Deep-sleep mode or Power-down mode.

An interrupt can be requested for each port, based on any selected subset of pins within each port. The pins that contribute to each port interrupt are selected by 1s in the port's Enable register, and an interrupt polarity can be selected for each pin in the port's Polarity register. The level on each pin is exclusive-ORed with its polarity bit, and the result is ANDed with its enable bit. These results are then inclusive-ORed among all the pins in the port to create the port's raw interrupt request.

The raw interrupt request from each of the two group interrupts is sent to the NVIC, which can be programmed to treat it as level- or edge-sensitive, or it can be edge-detected by the wake-up interrupt logic (see [Table 75](#)).

13.1 How to read this chapter

The DMA controller is available on all parts.

13.2 Features

- 22 channels, 21 of which are connected to peripheral DMA requests. These come from the USART, SPI, and I²C peripherals. One spare channels has no DMA request connected, and can be used for functions such as memory-to-memory moves.
- DMA operations can be triggered by on- or off-chip events. Each DMA channel can select one trigger input from 20 sources. Trigger sources include ADC interrupts, Timer interrupts, pin interrupts, and the SCT DMA request lines.
- Priority is user selectable for each channel (up to eight priority levels).
- Continuous priority arbitration.
- Address cache with four entries (each entry is a pair of addresses).
- Efficient use of data bus.
- Supports single transfers up to 1,024 words.
- Address increment options allow packing and/or unpacking data.

13.3 Basic configuration

Configure the DMA as follows:

Use the AHBCLKCTRL0 register ([Table 51](#)) to enable the clock to the DMA registers interface.

- Clear the DMA peripheral reset using the PRESETCTRL0 register ([Table 35](#)).
- The DMA interrupt is connected to slot #3 in the NVIC.
- Each DMA channel has one DMA request line associated and can also select one of 20 input triggers through the input mux registers DMA_ITRIG_INMUX[0:21].
- Trigger outputs are connected to DMA_INMUX_INMUX[0:3] as inputs to DMA triggers.

For details on the trigger input and output multiplexing, see [Section 9.5.2 “DMA trigger input multiplexing”](#).

13.3.1 Hardware triggers

Each DMA channel can use one trigger that is independent of the request input for this channel. The trigger input is selected in the DMA_ITRIG_INMUX registers. There are 20 possible internal trigger sources for each channel with each trigger signal issued by the output of a peripheral. In addition, the DMA trigger output can be routed to the trigger input of another channel through the trigger input multiplexing. See [Table 173](#) and [Section 9.5.2 “DMA trigger input multiplexing”](#).

Table 172. DMA trigger sources

| DMA trigger # | Trigger input |
|---------------|----------------------------------|
| 0 | ADC0 sequence A interrupt |
| 1 | ADC0 sequence B interrupt |
| 2 | SCT0 DMA request 0 |
| 3 | SCT0 DMA request 1 |
| 4 | Timer CT32B0 Match 0 DMA request |
| 5 | Timer CT32B0 Match 1 DMA request |
| 6 | Timer CT32B1 Match 0 DMA request |
| 7 | Timer CT32B2 Match 0 DMA request |
| 8 | Timer CT32B2 Match 1 DMA request |
| 9 | Timer CT32B3 Match 0 DMA request |
| 10 | Timer CT32B4 Match 0 DMA request |
| 11 | Timer CT32B4 Match 1 DMA request |
| 12 | Pin interrupt 0 |
| 13 | Pin interrupt 1 |
| 14 | Pin interrupt 2 |
| 15 | Pin interrupt 3 |
| 16 | DMA output trigger 0 |
| 17 | DMA output trigger 1 |
| 18 | DMA output trigger 2 |
| 19 | DMA output trigger 3 |

13.3.2 Trigger outputs

Each channel of the DMA controller provides a trigger output. This allows the possibility of using the trigger outputs as a trigger source to a different channel in order to support complex transfers on selected peripherals. This kind of transfer can, for example, use more than one peripheral DMA request. An example use would be to input data to a holding buffer from one peripheral, and then output the data to another peripheral, with both transfers being paced by the appropriate peripheral DMA request. This kind of operation is called “chained operation” or “channel chaining”.

13.3.3 DMA requests

DMA requests are directly connected to the peripherals. Each channel supports one DMA request line and one trigger input which is multiplexed to many possible input sources, as shown in [Table 173](#).

Table 173. DMA requests

| DMA channel # | Request input | DMA trigger mux |
|---------------|------------------|-------------------|
| 0 | USART0 RX | DMA_ITRIG_INMUX0 |
| 1 | USART0 TX | DMA_ITRIG_INMUX1 |
| 2 | USART1 RX | DMA_ITRIG_INMUX2 |
| 3 | USART1 TX | DMA_ITRIG_INMUX3 |
| 4 | USART2 RX | DMA_ITRIG_INMUX4 |
| 5 | USART2 TX | DMA_ITRIG_INMUX5 |
| 6 | USART3 RX | DMA_ITRIG_INMUX6 |
| 7 | USART3 TX | DMA_ITRIG_INMUX7 |
| 8 | SPI0 RX | DMA_ITRIG_INMUX8 |
| 9 | SPI0 TX | DMA_ITRIG_INMUX9 |
| 10 | SPI1 RX | DMA_ITRIG_INMUX10 |
| 11 | SPI1 TX | DMA_ITRIG_INMUX11 |
| 12 | I2C0 Slave | DMA_ITRIG_INMUX12 |
| 13 | I2C0 Master | DMA_ITRIG_INMUX13 |
| 14 | I2C1 Slave | DMA_ITRIG_INMUX14 |
| 15 | I2C1 Master | DMA_ITRIG_INMUX15 |
| 16 | I2C2 Slave | DMA_ITRIG_INMUX16 |
| 17 | I2C2 Master | DMA_ITRIG_INMUX17 |
| 18 | I2C0 Monitor | DMA_ITRIG_INMUX18 |
| 19 | I2C1 Monitor | DMA_ITRIG_INMUX19 |
| 20 | I2C2 Monitor | DMA_ITRIG_INMUX20 |
| 21 | (no DMA request) | DMA_ITRIG_INMUX21 |

13.3.4 DMA in sleep mode

The DMA can operate and access all SRAM blocks in sleep mode.

13.4 Pin description

The DMA controller has no configurable pins.

13.5 General description

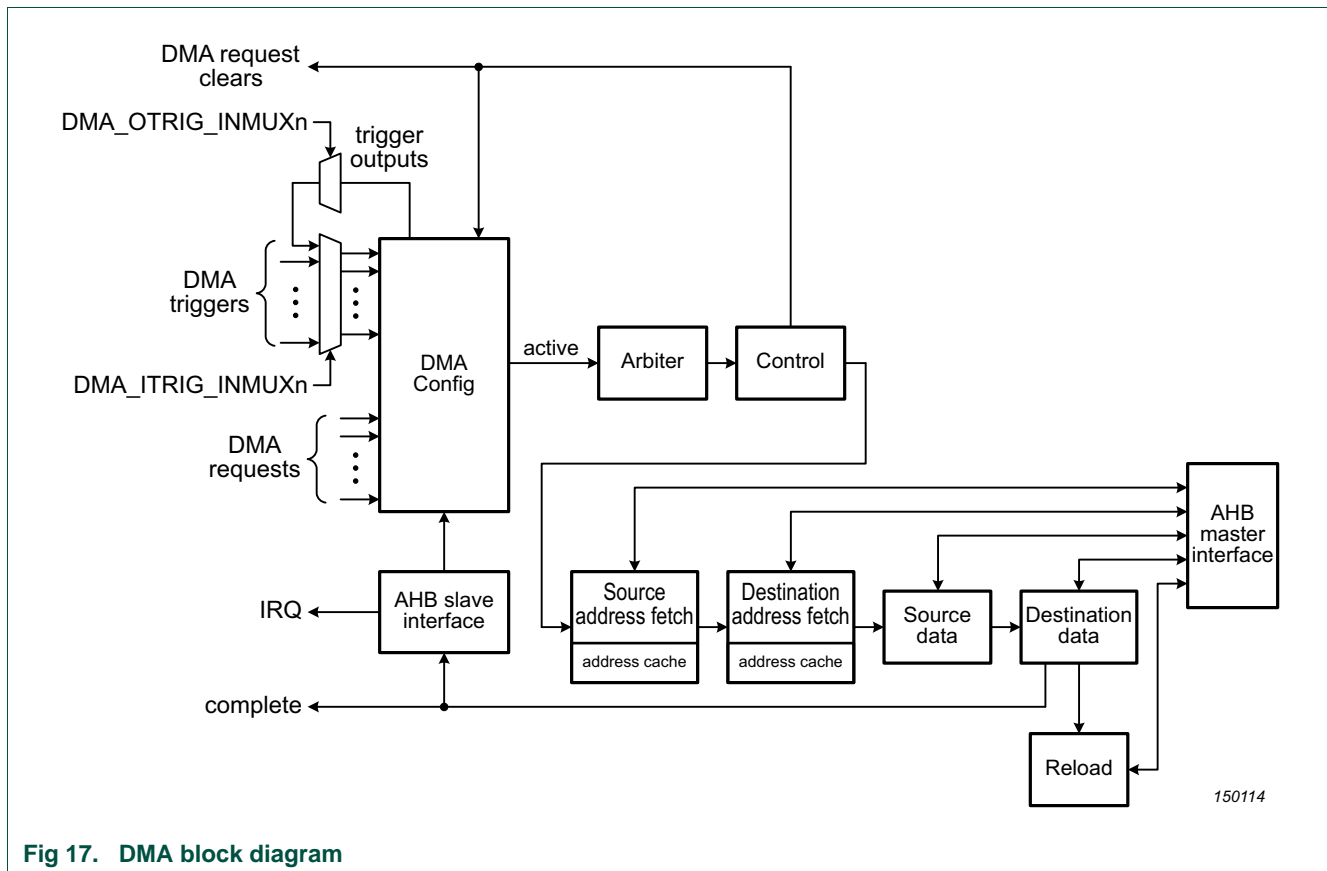


Fig 17. DMA block diagram

13.5.1 DMA requests and triggers

An operation on a DMA channel can be initiated by either a DMA request or a trigger event. DMA requests come from peripherals and specifically indicate when a peripheral either needs input data to be read from it, or that output data may be sent to it. DMA requests are created by the USART, SPI, and I2C peripherals.

A trigger initiates a DMA operation and can be a signal from an unrelated peripheral. Peripherals that generate triggers are the SCT, and the ADC. In addition, the DMA triggers also create an trigger output that can trigger DMA transactions on another channel. Triggers can be used to send a character or a string to a USART or other serial output at a fixed time interval or when an event occurs.

A DMA channel using a trigger can respond by moving data from any memory address to any other memory address. This can include fixed peripheral data registers, or incrementing through RAM buffers. The amount of data moved by a single trigger event

can range from a single transfer to many transfers. A transfer that is started by a trigger can still be paced using the channel's DMA request. This allows sending a string to a serial peripheral, for instance, without overrunning the peripheral's transmit buffer.

Each DMA channel has an output that can be used as a trigger input to another channel. The trigger outputs appear in the trigger source list for each channel and can be selected through the DMA_INMUX registers as inputs to other channels.

13.5.2 DMA Modes

The DMA controller doesn't really have separate operating modes, but there are ways of using the DMA controller that have commonly used terminology in the industry.

Once the DMA controller is set up for operation, using any specific DMA channel requires initializing the registers associated with that channel (see [Table 173](#)), and supplying at least the channel descriptor, which is located somewhere in memory, typically in on-chip SRAM (see [Section 13.6.3](#)). The channel descriptor is shown in [Table 174](#).

Table 174: Channel descriptor

| Offset | Description |
|--------|-------------------------|
| + 0x0 | Reserved |
| + 0x4 | Source data end address |
| + 0x8 | Destination end address |
| + 0xC | Link to next descriptor |

The source and destination end addresses, as well as the link to the next descriptor are just memory addresses that can point to any valid address on the device. The starting address for both source and destination data is the specified end address minus the transfer length ($XFERCOUNT * \text{the address increment as defined by SRCINC and DSTINC}$). The link to the next descriptor is used only if it is a linked transfer.

After the channel has had a sufficient number of DMA requests and/or triggers, depending on its configuration, the initial descriptor will be exhausted. At that point, if the transfer configuration directs it, the channel descriptor will be reloaded with data from memory pointed to by the "Link to next descriptor" entry of the initial channel descriptor.

Descriptors loaded in this manner look slightly different the channel descriptor, as shown in [Table 175](#). The difference is that a new transfer configuration is specified in the reload descriptor instead of being written to the XFERCFG register for that channel.

This process repeats as each descriptor is exhausted as long as reload is selected in the transfer configuration for each new descriptor.

Table 175: Reload descriptors

| Offset | Description |
|--------|---|
| + 0x0 | Transfer configuration. |
| + 0x4 | Source end address. This points to the address of the last entry of the source address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size. |
| + 0x8 | Destination end address. This points to the address of the last entry of the destination address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size. |
| + 0xC | Link to next descriptor. If used, this address must be aligned to a multiple of 16 bytes (i.e., the size of a descriptor). |

13.5.3 Single buffer

This generally applies to memory to memory moves, and peripheral DMA that occurs only occasionally and is set up for each transfer. For this kind of operation, only the initial channel descriptor shown in [Table 176](#) is needed.

Table 176: Channel descriptor for a single transfer

| Offset | Description |
|--------|------------------------------|
| + 0x0 | Reserved |
| + 0x4 | Source data end address |
| + 0x8 | Destination data end address |
| + 0xC | (not used) |

This case is identified by the Reload bit in the XFERCFG register = 0. When the DMA channel receives a DMA request or trigger (depending on how it is configured), it performs one or more transfers as configured, then stops. Once the channel descriptor is exhausted, additional DMA requests or triggers will have no effect until the channel configuration is updated by software.

13.5.4 Ping-Pong

Ping-pong is a special case of a linked transfer. It is described separately because it is typically used more frequently than more complicated versions of linked transfers.

A ping-pong transfer uses two buffers alternately. At any one time, one buffer is being loaded or unloaded by DMA operations. The other buffer has the opposite operation being handled by software, readying the buffer for use when the buffer currently being used by the DMA controller is full or empty. [Table 177](#) shows an example of descriptors for ping-pong from a peripheral to two buffers in memory.

Table 177: Example descriptors for ping-pong operation: peripheral to buffer

| Channel Descriptor | Descriptor B | Descriptor A |
|-----------------------------------|---------------------------------------|---------------------------------------|
| + 0x0 (not used) | + 0x0 Buffer B transfer configuration | + 0x0 Buffer A transfer configuration |
| + 0x4 Peripheral data end address | + 0x4 Peripheral data end address | + 0x4 Peripheral data end address |
| + 0x8 Buffer A memory end address | + 0x8 Buffer B memory end address | + 0x8 Buffer A memory end address |
| + 0xC Address of descriptor B | + 0xC Address of descriptor A | + 0xC Address of descriptor B |

In this example, the channel descriptor is used first, with a first buffer in memory called buffer A. The configuration of the DMA channel must have been set to indicate a reload. Similarly, both descriptor A and descriptor B must also specify reload. When the channel descriptor is exhausted, descriptor B is loaded using the link to descriptor B, and a transfer interrupt informs the CPU that buffer A is available.

Descriptor B is then used until it is also exhausted, when descriptor A is loaded using the link to descriptor A contained in descriptor B. Then a transfer interrupt informs the CPU that buffer B is available for processing. The process repeats when descriptor A is exhausted, alternately using each of the 2 memory buffers.

13.5.5 Linked transfers (linked list)

A linked transfer can use any number of descriptors to define a complicated transfer. This can be configured such that a single transfer, a portion of a transfer, one whole descriptor, or an entire structure of links can be initiated by a single DMA request or trigger.

An example of a linked transfer could start out like the example for a ping-pong transfer ([Table 177](#)). The difference would be that descriptor B would not link back to descriptor A, but would continue on to another different descriptor. This could continue as long as desired, and can be ended anywhere, or linked back to any point to repeat a sequence of descriptors. Of course, any descriptor not currently in use can be altered by software as well.

13.5.6 Address alignment for data transfers

Transfers of 16 bit width require an address alignment to a multiple of 2 bytes. Transfers of 32 bit width require an address alignment to a multiple of 4 bytes. Transfers of 8 bit width can be at any address.

13.6 Register description

The DMA registers are grouped into DMA control, interrupt and status registers and DMA channel registers. DMA transfers are controlled by a set of three registers per channel, the CFG[0:20], CTRLSTAT[0:20], and XFERCFG[0:20] registers.

The reset value reflects the data stored in used bits only. It does not include the content of reserved bits.

Table 178. Register overview: DMA controller (base address 0x1C00 4000)

| Name | Access | Address offset | Description | Reset value | Reference |
|--|--------|----------------|---|-------------|---------------------------|
| Global control and status registers | | | | | |
| CTRL | R/W | 0x000 | DMA control. | 0 | Table 179 |
| INTSTAT | RO | 0x004 | Interrupt status. | 0 | Table 180 |
| SRAMBASE | R/W | 0x008 | SRAM address of the channel configuration table. | 0 | Table 181 |
| Shared registers | | | | | |
| ENABLESET0 | R/W | 0x020 | Channel Enable read and Set for all DMA channels. | 0 | Table 183 |
| ENABLECLR0 | WO | 0x028 | Channel Enable Clear for all DMA channels. | NA | Table 184 |
| ACTIVE0 | RO | 0x030 | Channel Active status for all DMA channels. | 0 | Table 185 |
| BUSY0 | RO | 0x038 | Channel Busy status for all DMA channels. | 0 | Table 186 |
| ERRINT0 | R/W | 0x040 | Error Interrupt status for all DMA channels. | 0 | Table 187 |
| INTENSET0 | R/W | 0x048 | Interrupt Enable read and Set for all DMA channels. | 0 | Table 188 |
| INTENCLR0 | WO | 0x050 | Interrupt Enable Clear for all DMA channels. | NA | Table 189 |
| INTA0 | R/W | 0x058 | Interrupt A status for all DMA channels. | 0 | Table 190 |
| INTB0 | R/W | 0x060 | Interrupt B status for all DMA channels. | 0 | Table 191 |
| SETVALID0 | WO | 0x068 | Set ValidPending control bits for all DMA channels. | NA | Table 192 |
| SETTRIG0 | WO | 0x070 | Set Trigger control bits for all DMA channels. | NA | Table 193 |
| ABORT0 | WO | 0x078 | Channel Abort control for all DMA channels. | NA | Table 194 |
| Channel 0 registers | | | | | |
| CFG0 | R/W | 0x400 | Configuration register for DMA channel 0. | | Table 196 |
| CTLSTAT0 | RO | 0x404 | Control and status register for DMA channel 0. | | Table 199 |
| XFERCFG0 | R/W | 0x408 | Transfer configuration register for DMA channel 0. | | Table 201 |
| Channel 1 registers | | | | | |
| CFG1 | R/W | 0x410 | Configuration register for DMA channel 1. | | Table 196 |
| CTLSTAT1 | RO | 0x414 | Control and status register for DMA channel 1. | | Table 199 |
| XFERCFG1 | R/W | 0x418 | Transfer configuration register for DMA channel 1. | | Table 201 |
| Channel 2 registers | | | | | |
| CFG2 | R/W | 0x420 | Configuration register for DMA channel 2. | | Table 196 |
| CTLSTAT2 | RO | 0x424 | Control and status register for DMA channel 2. | | Table 199 |
| XFERCFG2 | R/W | 0x428 | Transfer configuration register for DMA channel 2. | | Table 201 |
| Channel 3 registers | | | | | |
| CFG3 | R/W | 0x430 | Configuration register for DMA channel 3. | | Table 196 |
| CTLSTAT3 | RO | 0x434 | Control and status register for DMA channel 3. | | Table 199 |
| XFERCFG3 | R/W | 0x438 | Transfer configuration register for DMA channel 3. | | Table 201 |

Table 178. Register overview: DMA controller (base address 0x1C00 4000)

| Name | Access | Address offset | Description | Reset value | Reference |
|-----------------------------|--------|----------------|---|-------------|---------------------------|
| Channel 4 registers | | | | | |
| CFG4 | R/W | 0x440 | Configuration register for DMA channel 4. | | Table 196 |
| CTLSTAT4 | RO | 0x444 | Control and status register for DMA channel 4. | | Table 199 |
| XFERCFG4 | R/W | 0x448 | Transfer configuration register for DMA channel 4. | | Table 201 |
| Channel 5 registers | | | | | |
| CFG5 | R/W | 0x450 | Configuration register for DMA channel 5. | | Table 196 |
| CTLSTAT5 | RO | 0x454 | Control and status register for DMA channel 5. | | Table 199 |
| XFERCFG5 | R/W | 0x458 | Transfer configuration register for DMA channel 5. | | Table 201 |
| Channel 6 registers | | | | | |
| CFG6 | R/W | 0x460 | Configuration register for DMA channel 6. | | Table 196 |
| CTLSTAT6 | RO | 0x464 | Control and status register for DMA channel 6. | | Table 199 |
| XFERCFG6 | R/W | 0x468 | Transfer configuration register for DMA channel 6. | | Table 201 |
| Channel 7 registers | | | | | |
| CFG7 | R/W | 0x470 | Configuration register for DMA channel 7. | | Table 196 |
| CTLSTAT7 | RO | 0x474 | Control and status register for DMA channel 7. | | Table 199 |
| XFERCFG7 | R/W | 0x478 | Transfer configuration register for DMA channel 7. | | Table 201 |
| Channel 8 registers | | | | | |
| CFG8 | R/W | 0x480 | Configuration register for DMA channel 8. | | Table 196 |
| CTLSTAT8 | RO | 0x484 | Control and status register for DMA channel 8. | | Table 199 |
| XFERCFG8 | R/W | 0x488 | Transfer configuration register for DMA channel 8. | | Table 201 |
| Channel 9 registers | | | | | |
| CFG9 | R/W | 0x490 | Configuration register for DMA channel 9. | | Table 196 |
| CTLSTAT9 | RO | 0x494 | Control and status register for DMA channel 9. | | Table 199 |
| XFERCFG9 | R/W | 0x498 | Transfer configuration register for DMA channel 9. | | Table 201 |
| Channel 10 registers | | | | | |
| CFG10 | R/W | 0x4A0 | Configuration register for DMA channel 10. | | Table 196 |
| CTLSTAT10 | RO | 0x4A4 | Control and status register for DMA channel 10. | | Table 199 |
| XFERCFG10 | R/W | 0x4A8 | Transfer configuration register for DMA channel 10. | | Table 201 |
| Channel 11 registers | | | | | |
| CFG11 | R/W | 0x4B0 | Configuration register for DMA channel 11. | | Table 196 |
| CTLSTAT11 | RO | 0x4B4 | Control and status register for DMA channel 11. | | Table 199 |
| XFERCFG11 | R/W | 0x4B8 | Transfer configuration register for DMA channel 11. | | Table 201 |
| Channel 12 registers | | | | | |
| CFG12 | R/W | 0x4C0 | Configuration register for DMA channel 12. | | Table 196 |
| CTLSTAT12 | RO | 0x4C4 | Control and status register for DMA channel 12. | | Table 199 |
| XFERCFG12 | R/W | 0x4C8 | Transfer configuration register for DMA channel 12. | | Table 201 |
| Channel 13 registers | | | | | |
| CFG13 | R/W | 0x4D0 | Configuration register for DMA channel 13. | | Table 196 |
| CTLSTAT13 | RO | 0x4D4 | Control and status register for DMA channel 13. | | Table 199 |
| XFERCFG13 | R/W | 0x4D8 | Transfer configuration register for DMA channel 13. | | Table 201 |

Table 178. Register overview: DMA controller (base address 0x1C00 4000)

| Name | Access | Address offset | Description | Reset value | Reference |
|-----------------------------|--------|----------------|---|-------------|---------------------------|
| Channel 14 registers | | | | | |
| CFG14 | R/W | 0x4E0 | Configuration register for DMA channel 14. | | Table 196 |
| CTLSTAT14 | RO | 0x4E4 | Control and status register for DMA channel 14. | | Table 199 |
| XFERCFG14 | R/W | 0x4E8 | Transfer configuration register for DMA channel 14. | | Table 201 |
| Channel 15 registers | | | | | |
| CFG15 | R/W | 0x4F0 | Configuration register for DMA channel 15. | | Table 196 |
| CTLSTAT15 | RO | 0x4F4 | Control and status register for DMA channel 15. | | Table 199 |
| XFERCFG15 | R/W | 0x4F8 | Transfer configuration register for DMA channel 15. | | Table 201 |
| Channel 16 registers | | | | | |
| CFG16 | R/W | 0x500 | Configuration register for DMA channel 16. | | Table 196 |
| CTLSTAT16 | RO | 0x504 | Control and status register for DMA channel 16. | | Table 199 |
| XFERCFG16 | R/W | 0x508 | Transfer configuration register for DMA channel 16. | | Table 201 |
| Channel 17 registers | | | | | |
| CFG17 | R/W | 0x510 | Configuration register for DMA channel 17. | | Table 196 |
| CTLSTAT17 | RO | 0x514 | Control and status register for DMA channel 17. | | Table 199 |
| XFERCFG17 | R/W | 0x518 | Transfer configuration register for DMA channel 17. | | Table 201 |
| Channel 18 registers | | | | | |
| CFG18 | R/W | 0x520 | Configuration register for DMA channel 18. | | Table 196 |
| CTLSTAT18 | RO | 0x524 | Control and status register for DMA channel 18. | | Table 199 |
| XFERCFG18 | R/W | 0x528 | Transfer configuration register for DMA channel 18. | | Table 201 |
| Channel 19 registers | | | | | |
| CFG19 | R/W | 0x530 | Configuration register for DMA channel 19. | | Table 196 |
| CTLSTAT19 | RO | 0x534 | Control and status register for DMA channel 19. | | Table 199 |
| XFERCFG19 | R/W | 0x538 | Transfer configuration register for DMA channel 19. | | Table 201 |
| Channel 20 registers | | | | | |
| CFG20 | R/W | 0x540 | Configuration register for DMA channel 20. | | Table 196 |
| CTLSTAT20 | RO | 0x544 | Control and status register for DMA channel 20. | | Table 199 |
| XFERCFG20 | R/W | 0x548 | Transfer configuration register for DMA channel 20. | | Table 201 |
| Channel 21 registers | | | | | |
| CFG21 | R/W | 0x550 | Configuration register for DMA channel 21. | | Table 196 |
| CTLSTAT21 | RO | 0x554 | Control and status register for DMA channel 21. | | Table 199 |
| XFERCFG21 | R/W | 0x558 | Transfer configuration register for DMA channel 21. | | Table 201 |

13.6.1 Control register

The CTRL register contains global the control bit for a enabling the DMA controller.

Table 179. Control register (CTRL, address 0x1C00 4000) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---|-------------|
| 0 | ENABLE | | DMA controller master enable. | 0 |
| | | 0 | Disabled. The DMA controller is disabled. This clears any triggers that were asserted at the point when disabled, but does not prevent re-triggering when the DMA controller is re-enabled. | |
| | | 1 | Enabled. The DMA controller is enabled. | |
| 31:1 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

13.6.2 Interrupt Status register

The Read-Only INTSTAT register provides an overview of DMA status. This allows quick determination of whether any enabled interrupts are pending. Details of which channels are involved are found in the interrupt type specific registers.

Table 180. Interrupt Status register (INTSTAT, address 0x1C00 4004) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------------|-------|--|-------------|
| 0 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 1 | ACTIVEINT | | Summarizes whether any enabled interrupts (other than error interrupts) are pending. | 0 |
| | | 0 | Not pending. No enabled interrupts are pending. | |
| | | 1 | Pending. At least one enabled interrupt is pending. | |
| 2 | ACTIVEERRINT | | Summarizes whether any error interrupts are pending. | 0 |
| | | 0 | Not pending. No error interrupts are pending. | |
| | | 1 | Pending. At least one error interrupt is pending. | |
| 31:3 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

13.6.3 SRAM Base address register

The SRAMBASE register must be configured with an address (preferably in on-chip SRAM) where DMA descriptors will be stored. Software must set up the descriptors for those DMA channels that will be used in the application.

Table 181. SRAM Base address register (SRAMBASE, address 0x1C00 4008) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 8:0 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 31:9 | OFFSET | Address bits 31:9 of the beginning of the DMA descriptor table. For 18 channels, the table must begin on a 512 byte boundary. | 0 |

Each DMA channel has an entry for the channel descriptor in the SRAM table. The values for each channel start at the address offsets found in [Table 182](#). Only the descriptors for channels defined at extraction are used. The contents of each channel descriptor are described in [Table 174](#).

Table 182. Channel descriptor map

| Descriptor | Table offset |
|---------------------------------------|--------------|
| Channel descriptor for DMA channel 0 | 0x000 |
| Channel descriptor for DMA channel 1 | 0x010 |
| Channel descriptor for DMA channel 2 | 0x020 |
| Channel descriptor for DMA channel 3 | 0x030 |
| Channel descriptor for DMA channel 4 | 0x040 |
| Channel descriptor for DMA channel 5 | 0x050 |
| Channel descriptor for DMA channel 6 | 0x060 |
| Channel descriptor for DMA channel 7 | 0x070 |
| Channel descriptor for DMA channel 8 | 0x080 |
| Channel descriptor for DMA channel 9 | 0x090 |
| Channel descriptor for DMA channel 10 | 0x0A0 |
| Channel descriptor for DMA channel 11 | 0x0B0 |
| Channel descriptor for DMA channel 12 | 0x0C0 |
| Channel descriptor for DMA channel 13 | 0x0D0 |
| Channel descriptor for DMA channel 14 | 0x0E0 |
| Channel descriptor for DMA channel 15 | 0x0F0 |
| Channel descriptor for DMA channel 16 | 0x100 |
| Channel descriptor for DMA channel 17 | 0x110 |
| Channel descriptor for DMA channel 18 | 0x120 |
| Channel descriptor for DMA channel 19 | 0x130 |
| Channel descriptor for DMA channel 20 | 0x140 |
| Channel descriptor for DMA channel 21 | 0x150 |

13.6.4 Enable read and Set registers

The ENABLESET0 register determines whether each DMA channel is enabled or disabled. Disabling a DMA channel does not reset the channel in any way. A channel can be paused and restarted by clearing, then setting the Enable bit for that channel.

Reading ENABLESET0 provides the current state of all of the DMA channels represented by that register. Writing a 1 to a bit position in ENABLESET0 that corresponds to an implemented DMA channel sets the bit, enabling the related DMA channel. Writing a 0 to any bit has no effect. Enables are cleared by writing to ENABLECLR0.

Table 183. Enable read and Set register 0 (ENABLESET0, address 0x1C00 4020) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 21:0 | ENA | Enable for DMA channels. Bit n enables or disables DMA channel n. 0 = disabled. 1 = enabled. | 0 |
| 31:22 | - | Reserved. Read value is undefined, only zero should be written. | - |

13.6.5 Enable Clear register

The ENABLECLR0 register is used to clear the enable of one or more DMA channels. This register is write-only.

Table 184. Enable Clear register 0 (ENABLECLR0, address 0x1C00 4028) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 21:0 | CLR | Writing ones to this register clears the corresponding bits in ENABLESET0. Bit n clears the channel enable bit n. | NA |
| 31:22 | - | Reserved. | - |

13.6.6 Active status register

The ACTIVE0 register indicates which DMA channels are active at the point when the read occurs. The register is read-only.

A DMA channel is considered active when a DMA operation has been started but not yet fully completed. The Active status will persist from a DMA operation being started, until the pipeline is empty after end of the last descriptor (when there is no reload). An active channel may be aborted by software by setting the appropriate bit in one of the Abort register (see [Section 13.6.15](#)).

Table 185. Active status register 0 (ACTIVE0, address 0x1C00 4030) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 21:0 | ACT | Active flag for DMA channel n. Bit n corresponds to DMA channel n. 0 = not active. 1 = active. | 0 |
| 31:22 | - | Reserved. | - |

13.6.7 Busy status register

The BUSY0 register indicates which DMA channels is busy at the point when the read occurs. This registers is read-only.

A DMA channel is considered busy when there is any operation related to that channel in the DMA controller's internal pipeline. This information can be used after a DMA channel is disabled by software (but still active), allowing confirmation that there are no remaining operations in progress for that channel.

Table 186. Busy status register 0 (BUSY0, address 0x1C00 4038) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 21:0 | BSY | Busy flag for DMA channel n. Bit n corresponds to DMA channel n. 0 = not busy. 1 = busy. | 0 |
| 31:22 | - | Reserved. | - |

13.6.8 Error Interrupt register

The ERRINT0 register contains flags for each DMA channel's Error Interrupt. Any pending interrupt flag in the register will be reflected on the DMA interrupt output.

Reading the registers provides the current state of all DMA channel error interrupts. Writing a 1 to a bit position in ERRINT0 that corresponds to an implemented DMA channel clears the bit, removing the interrupt for the related DMA channel. Writing a 0 to any bit has no effect.

Table 187. Error Interrupt register 0 (ERRINT0, address 0x1C00 4040) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 21:0 | ERR | Error Interrupt flag for DMA channel n. Bit n corresponds to DMA channel n. 0 = error interrupt is not active. 1 = error interrupt is active. | 0 |
| 31:22 | - | Reserved. | - |

13.6.9 Interrupt Enable read and Set register

The INTENSET0 register controls whether the individual Interrupts for DMA channels contribute to the DMA interrupt output.

Reading the registers provides the current state of all DMA channel interrupt enables. Writing a 1 to a bit position in INTENSET0 that corresponds to an implemented DMA channel sets the bit, enabling the interrupt for the related DMA channel. Writing a 0 to any bit has no effect. Interrupt enables are cleared by writing to INTENCLR0.

Table 188. Interrupt Enable read and Set register 0 (INTENSET0, address 0x1C00 4048) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 21: 0 | INTEN | Interrupt Enable read and set for DMA channel n. Bit n corresponds to DMA channel n. 0 = interrupt for DMA channel is disabled. 1 = interrupt for DMA channel is enabled. | 0 |
| 31:22 | - | Reserved. | - |

13.6.10 Interrupt Enable Clear register

The INTENCLR0 register is used to clear interrupt enable bits in INTENSET0. The register is write-only.

Table 189. Interrupt Enable Clear register 0 (INTENCLR0, address 0x1C00 4050) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 21:0 | CLR | Writing ones to this register clears corresponding bits in the INTENSET0. Bit n corresponds to DMA channel n. | NA |
| 31:22 | - | Reserved. | - |

13.6.11 Interrupt A register

The IntA0 register contains the interrupt A status for each DMA channel. The status will be set when the SETINTA bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in this register clears the related INTA flag. Writing 0 has no effect. Any interrupt pending status in the registers will be reflected on the DMA interrupt output if it is enabled in the related INTENSET register.

Table 190. Interrupt A register 0 (INTA0, address 0x1C00 4058) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 21:0 | IA | Interrupt A status for DMA channel n. Bit n corresponds to DMA channel n. 0 = the DMA channel interrupt A is not active. 1 = the DMA channel interrupt A is active. | 0 |
| 31:22 | - | Reserved. Read value is undefined, only zero should be written. | - |

13.6.12 Interrupt B register

The INTB0 register contains the interrupt B status for each DMA channel. The status will be set when the SETINTB bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in the register clears the related INTB flag. Writing 0 has no effect. Any interrupt pending status in this register will be reflected on the DMA interrupt output if it is enabled in the INTENSET register.

Table 191. Interrupt B register 0 (INTB0, address 0x1C00 4060) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 21:0 | IB | Interrupt B status for DMA channel n. Bit n corresponds to DMA channel n. 0 = the DMA channel interrupt B is not active. 1 = the DMA channel interrupt B is active. | 0 |
| 31:22 | - | Reserved. Read value is undefined, only zero should be written. | - |

13.6.13 Set Valid register

The SETVALID0 register allows setting the Valid bit in the CTRLSTAT register for one or more DMA channels. See [Section 13.6.17](#) for a description of the VALID bit.

The CFGVALID and SV (set valid) bits allow more direct DMA block timing control by software. Each Channel Descriptor, in a sequence of descriptors, can be validated by either the setting of the CFGVALID bit or by setting the channel's SETVALID flag. Normally, the CFGVALID bit is set. This tells the DMA that the Channel Descriptor is active and can be executed. The DMA will continue sequencing through descriptor blocks whose CFGVALID bit are set without further software intervention. Leaving a CFGVALID bit set to 0 allows the DMA sequence to pause at the Descriptor until software triggers the continuation. If, during DMA transmission, a Channel Descriptor is found with CFGVALID set to 0, the DMA checks for a previously buffered SETVALID0 setting for the channel. If found, the DMA will set the descriptor valid, clear the SV setting, and resume processing the descriptor. Otherwise, the DMA pauses until the channels SETVALID0 bit is set.

Table 192. Set Valid 0 register (SETVALID0, address 0x1C00 4068) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 21:0 | SV | SETVALID control for DMA channel n. Bit n corresponds to DMA channel n. 0 = no effect. 1 = sets the VALIDPENDING control bit for DMA channel n. | NA |
| 31:22 | - | Reserved. | - |

13.6.14 Set Trigger register

The SETTRIG0 register allows setting the TRIG bit in the CTRLSTAT register for one or more DMA channel. See [Section 13.6.17](#) for a description of the TRIG bit, and [Section 13.5.1](#) for a general description of triggering.

Table 193. Set Trigger 0 register (SETTRIG0, address 0x1C00 4070) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 21:0 | TRIG | Set Trigger control bit for DMA channel 0. Bit n corresponds to DMA channel n. 0 = no effect. 1 = sets the TRIG bit for DMA channel n. | NA |
| 31:22 | - | Reserved. | - |

13.6.15 Abort registers

The Abort0 register allows aborting operation of a DMA channel if needed. To abort a selected channel, the channel should first be disabled by clearing the corresponding Enable bit by writing a 1 to the proper bit ENABLECLR. Then wait until the channel is no longer busy by checking the corresponding bit in BUSY. Finally, write a 1 to the proper bit of ABORT. This prevents the channel from restarting an incomplete operation when it is enabled again.

Table 194. Abort 0 register (ABORT0, address 0x1C00 4078) bit description

| Bit | Symbol | Description | Reset value |
|-------|-----------|---|-------------|
| 21:0 | ABORTCTRL | Abort control for DMA channel 0. Bit n corresponds to DMA channel n. 0 = no effect. 1 = aborts DMA operations on channel n. | NA |
| 31:22 | - | Reserved. | - |

13.6.16 Channel configuration registers

The CFGn register contains various configuration options for DMA channel n.

See [Table 197](#) for a summary of trigger options.

Table 195. Address map CFG[0:21] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| DMA | 0x1C00 4000 | [0x400:0x550] | 0x10 | 22 |

Table 196. Channel configuration registers bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|-------------|-------|---|-------------|
| 0 | PERIPHREQEN | | Peripheral request Enable. If a DMA channel is used to perform a memory-to-memory move, any peripheral DMA request associated with that channel can be disabled to prevent any interaction between the peripheral and the DMA controller. | 0 |
| | | 0 | Disabled. Peripheral DMA requests are disabled. | |
| | | 1 | Enabled. Peripheral DMA requests are enabled. | |
| 1 | HWTRIGEN | | Hardware Triggering Enable for this channel. | 0 |
| | | 0 | Disabled. Hardware triggering is not used. | |
| | | 1 | Enabled. Use hardware triggering. | |
| 3:2 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 4 | TRIGPOL | | Trigger Polarity. Selects the polarity of a hardware trigger for this channel. | 0 |
| | | 0 | Active low - falling edge. Hardware trigger is active low or falling edge triggered, based on TRIGTYPE. | |
| | | 1 | Active high - rising edge. Hardware trigger is active high or rising edge triggered, based on TRIGTYPE. | |
| 5 | TRIGTYPE | | Trigger Type. Selects hardware trigger as edge triggered or level triggered. | 0 |
| | | 0 | Edge. Hardware trigger is edge triggered. Transfers will be initiated and completed, as specified for a single trigger. | |
| | | 1 | Level. Hardware trigger is level triggered. Note that when level triggering without burst (BURSTPOWER = 0) is selected, only hardware triggers should be used on that channel. Transfers continue as long as the trigger level is asserted. Once the trigger is de-asserted, the transfer will be paused until the trigger is, again, asserted. However, the transfer will not be paused until any remaining transfers within the current BURSTPOWER length are completed. | |
| 6 | TRIGBURST | | Trigger Burst. Selects whether hardware triggers cause a single or burst transfer. | 0 |
| | | 0 | Single transfer. Hardware trigger causes a single transfer. | |
| | | 1 | Burst transfer. When the trigger for this channel is set to edge triggered, a hardware trigger causes a burst transfer, as defined by BURSTPOWER. When the trigger for this channel is set to level triggered, a hardware trigger causes transfers to continue as long as the trigger is asserted, unless the transfer is complete. | |
| 7 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

Table 196. Channel configuration registers bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|--------------|-------|--|-------------|
| 11:8 | BURSTPOWER | | <p>Burst Power is used in two ways. It always selects the address wrap size when SRCBURSTWRAP and/or DSTBURSTWRAP modes are selected (see descriptions elsewhere in this register).</p> <p>When the TRIGBURST field elsewhere in this register = 1, Burst Power selects how many transfers are performed for each DMA trigger. This can be used, for example, with peripherals that contain a FIFO that can initiate a DMA operation when the FIFO reaches a certain level.</p> <p>0000: Burst size = 1 (2^0).</p> <p>0001: Burst size = 2 (2^1).</p> <p>0010: Burst size = 4 (2^2).</p> <p>...</p> <p>1010: Burst size = 1024 (2^{10}). This corresponds to the maximum supported transfer count.</p> <p>others: not supported.</p> <p>The total transfer length as defined in the XFERCOUNT bits in the XFERCFG register must be an even multiple of the burst size.</p> | 0 |
| 13:12 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 14 | SRCBURSTWRAP | | <p>Source Burst Wrap. When enabled, the source data address for the DMA is “wrapped”, meaning that the source address range for each burst will be the same. As an example, this could be used to read several sequential registers from a peripheral for each DMA burst, reading the same registers again for each burst.</p> | 0 |
| | | 0 | Disabled. Source burst wrapping is not enabled for this DMA channel. | |
| | | 1 | Enabled. Source burst wrapping is enabled for this DMA channel. | |
| 15 | DSTBURSTWRAP | | <p>Destination Burst Wrap. When enabled, the destination data address for the DMA is “wrapped”, meaning that the destination address range for each burst will be the same. As an example, this could be used to write several sequential registers to a peripheral for each DMA burst, writing the same registers again for each burst.</p> | 0 |
| | | 0 | Disabled. Destination burst wrapping is not enabled for this DMA channel. | |
| | | 1 | Enabled. Destination burst wrapping is enabled for this DMA channel. | |
| 18:16 | CHPRIORITY | | <p>Priority of this channel when multiple DMA requests are pending.</p> <p>Eight priority levels are supported.</p> <p>0x0 = highest priority.</p> <p>0x7 = lowest priority.</p> | 0 |
| 31:19 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

Table 197. Trigger setting summary

| TrigBurst | TrigType | TrigPol | Description |
|-----------|----------|---------|---|
| 0 | 0 | 0 | Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap. |
| 0 | 0 | 1 | Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap. |
| 0 | 1 | 0 | Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap. |

Table 197. Trigger setting summary

| TrigBurst | TrigType | TrigPol | Description |
|-----------|----------|---------|--|
| 0 | 1 | 1 | Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap. |
| 1 | 0 | 0 | Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger. |
| 1 | 0 | 1 | Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger. |
| 1 | 1 | 0 | Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger. |
| 1 | 1 | 1 | Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger. |

13.6.17 Channel control and status registers

The CTLSTATn register provides status flags specific to DMA channel n.

Table 198. Address map CTLSTAT[0:21] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| DMA | 0x1C00 4000 | [0x404:0x554] | 0x10 | 22 |

Table 199. Channel control and status registers bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------------|-------|--|-------------|
| 0 | VALIDPENDING | | Valid pending flag for this channel. This bit is set when a 1 is written to the corresponding bit in the related SETVALID register when CFGVALID = 1 for the same channel. | 0 |
| | | 0 | No effect. No effect on DMA operation. | |
| | | 1 | Valid pending. | |
| 1 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 2 | TRIG | | Trigger flag. Indicates that the trigger for this channel is currently set. This bit is cleared at the end of an entire transfer or upon reload when CLRTRIG = 1. | 0 |
| | | 0 | Not triggered. The trigger for this DMA channel is not set. DMA operations will not be carried out. | |
| | | 1 | Triggered. The trigger for this DMA channel is set. DMA operations will be carried out. | |
| 31:3 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

13.6.18 Channel transfer configuration registers

The XFERCFGn register contains transfer related configuration information for DMA channel n. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

See [“Trigger operation”](#) for details on trigger operation.

Table 200. Address map XFERCFG[0:21] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| DMA | 0x1C00 4000 | [0x408:0x558] | 0x10 | 22 |

Table 201. Channel transfer configuration registers bit description

| Bit | Symbol | Value | Description | Reset Value |
|-----|----------|-------|--|-------------|
| 0 | CFGVALID | | Configuration Valid flag. This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled. | 0 |
| | | 0 | Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting. | |
| | | 1 | Valid. The current channel descriptor is considered valid. | |
| 1 | RELOAD | | Indicates whether the channel's control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers. | 0 |
| | | 0 | Disabled. Do not reload the channels' control structure when the current descriptor is exhausted. | |
| | | 1 | Enabled. Reload the channels' control structure when the current descriptor is exhausted. | |
| 2 | SWTRIG | | Software Trigger. | 0 |
| | | 0 | Not set. When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel. | |
| | | 1 | Set. When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0. | |
| 3 | CLRTRIG | | Clear Trigger. | 0 |
| | | 0 | Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started. | |
| | | 1 | Cleared. The trigger is cleared when this descriptor is exhausted. | |
| 4 | SETINTA | | Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. | 0 |
| | | 0 | No effect. | |
| | | 1 | Set. The INTA flag for this channel will be set when the current descriptor is exhausted. | |

Table 201. Channel transfer configuration registers bit description

| Bit | Symbol | Value | Description | Reset Value |
|-------|-----------|-------|--|-------------|
| 5 | SETINTB | | Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed. | 0 |
| | | 0 | No effect. | |
| | | 1 | Set. The INTB flag for this channel will be set when the current descriptor is exhausted. | |
| 7:6 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 9:8 | WIDTH | | Transfer width used for this DMA channel. | 0 |
| | | 0x0 | 8-bit. 8-bit transfers are performed (8-bit source reads and destination writes). | |
| | | 0x1 | 16-bit. 6-bit transfers are performed (16-bit source reads and destination writes). | |
| | | 0x2 | 32-bit. 32-bit transfers are performed (32-bit source reads and destination writes). | |
| | | 0x3 | Reserved. Reserved setting, do not use. | |
| 11:10 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 13:12 | SRCINC | | Determines whether the source address is incremented for each DMA transfer. | 0 |
| | | 0x0 | No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device. | |
| | | 0x1 | 1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory. | |
| | | 0x2 | 2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer. | |
| | | 0x3 | 4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer. | |
| 15:14 | DSTINC | | Determines whether the destination address is incremented for each DMA transfer. | 0 |
| | | 0x0 | No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device. | |
| | | 0x1 | 1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory. | |
| | | 0x2 | 2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer. | |
| | | 0x3 | 4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer. | |
| 25:16 | XFERCOUNT | | Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: $(XFERCOUNT + 1) \times$ data width (as defined by the WIDTH field). Remark: The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 = a total of 1 transfer will be performed. 0x1 = a total of 2 transfers will be performed. ... 0x3FF = a total of 1,024 transfers will be performed. | 0 |
| 31:26 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

13.7 Functional description

13.7.1 Trigger operation

A trigger of some kind is always needed to start a transfer on a DMA channel. This can be a hardware or software trigger, and can be used in several ways.

If a channel is configured with the SWTRIG bit equal to 0, the channel can be later triggered either by hardware or software. Software triggering is accomplished by writing a 1 to the appropriate bit in the SETTRIG register. Hardware triggering requires setup of the HWTRIGEN, TRIGPOL, TRIGTYPE, and TRIGBURST fields in the CFG register for the related channel. When a channel is initially set up, the SWTRIG bit in the XFERCFG register can be set, causing the transfer to begin immediately.

Once triggered, transfer on a channel will be paced by DMA requests if the PERIPHREQEN bit in the related CFG register is set. Otherwise, the transfer will proceed at full speed.

The TRIG bit in the CTLSTAT register can be cleared at the end of a transfer, determined by the value CLRTRIG (bit 0) in the XFERCFG register. When a 1 is found in CLRTRIG, the trigger is cleared when the descriptor is exhausted.

14.1 How to read this chapter

The SCTimer/PWM is available on all parts.

Remark: For a detailed description of SCTimer/PWM applications and code examples, see [Ref. 3 “AN11538”](#).

14.2 Features

- The SCTimer/PWM supports:
 - Thirteen match/capture registers.
 - Thirteen events.
 - Thirteen states.
 - Eight inputs.
 - Eight outputs.
- Counter/timer features:
 - Each SCTimer is configurable as two 16-bit counters or one 32-bit counter.
 - Counters clocked by system clock or selected input.
 - Configurable as up counters or up-down counters.
 - Configurable number of match and capture registers. Up to five match and capture registers total.
 - Upon match and/or an input or output transition create the following events: interrupt; stop, limit, halt the timer or change counting direction; toggle outputs; change the state.
 - Counter value can be loaded into capture register triggered by a match or input/output toggle.
- PWM features:
 - Counters can be used in conjunction with match registers to toggle outputs and create time-proportioned PWM signals.
 - Up to 8 single-edge or 6 dual-edge PWM outputs with independent duty cycle and common PWM cycle length.
- Event creation features:
 - The following conditions define an event: a counter match condition, an input (or output) condition such as an rising or falling edge or level, a combination of match and/or input/output condition.
 - Selected events can limit, halt, start, or stop a counter or change its direction.
 - Events trigger state changes, output toggles, interrupts, and DMA transactions.
 - Match register 0 can be used as an automatic limit.
 - In bi-directional mode, events can be enabled based on the count direction.
 - Match events can be held until another qualifying event occurs.

- State control features:
 - A state is defined by events that can happen in the state while the counter is running.
 - A state changes into another state as a result of an event.
 - Each event can be assigned to one or more states.
 - State variable allows sequencing across multiple counter cycles.

14.3 Basic configuration

Configure the SCT as follows:

- Enable the clock to the SCTimer/PWM (SCT) in the AHBCLKCTRL1 register ([Section 4.5.23](#)) to enable the register interface and the peripheral clock.
- Clear the SCT peripheral reset using the PRESETCTRL register ([Section 4.5.7](#)).
- The SCT uses interrupt in slot #16 in the NVIC ([Table 2](#)).
- Use the IOCON registers to connect the SCT outputs to external pins. See [Chapter 8](#).
- The SCT DMA request lines are connected to the DMA trigger inputs via the DMA_ITRIG_PINMUX registers. See [Section 9.6.2 "DMA trigger input mux registers 0 to 21"](#).

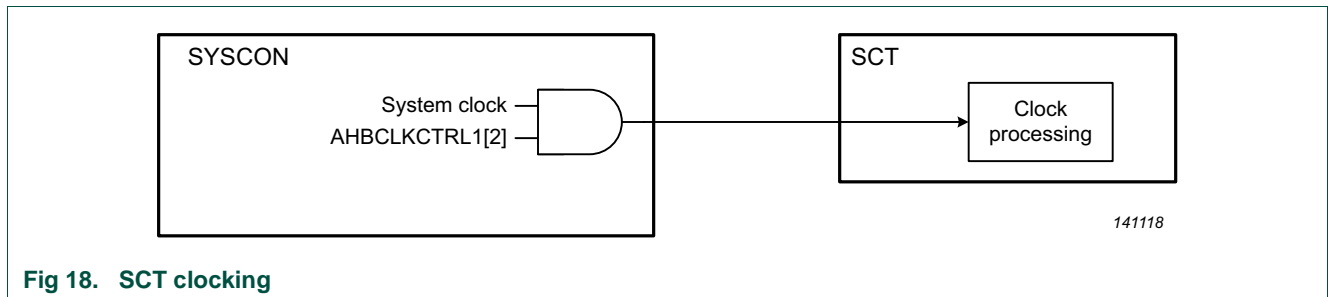


Fig 18. SCT clocking

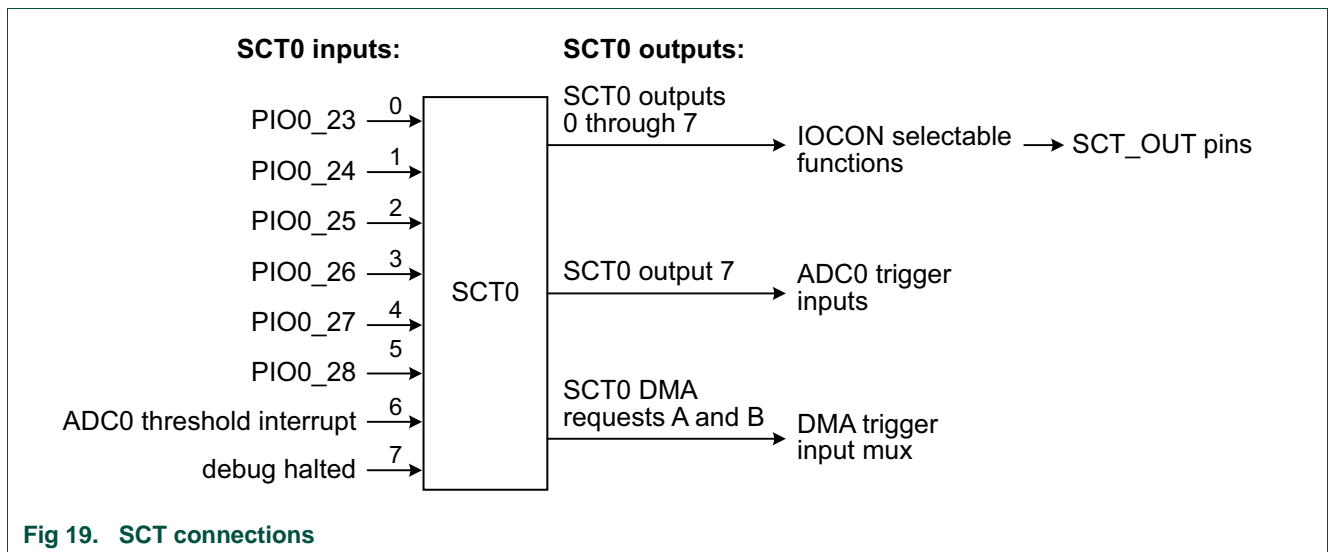


Fig 19. SCT connections

14.4 Pin description

See [Chapter 8](#) to assign the SCT functions to external pins.

SCT input signals are predefined. The signals from external pins and internal signals are connected directly to the SCT inputs and not routed through IOCON.

SCT outputs can be routed to multiple places and can be connected to both a pin and an ADC trigger at the same time.

Recommended IOCON settings are shown in [Table 204](#) and [Table 205](#).

Table 202. SCT0 pin description (inputs)

| Function | Type | Connect to | Reference |
|--------------|-------------------|----------------------------------|---------------------------|
| SCT0_IN[0:7] | external from pin | 6 GPIO pins (PIO0_23 to PIO0_28) | Figure 19 |
| | internal | ADC0_THCMP_IRQ, DEBUG_HALTED | Figure 19 |

Table 203. SCT0 pin description (outputs)

| Type | Function | Connect to | Use register | Reference |
|-----------------|-----------|--------------------------|------------------------------------|---|
| external to pin | SCT0_OUT0 | PIO0_7, PIO018 | IOCON register for the related pin | See Chapter 8 |
| | SCT0_OUT1 | PIO0_1, PIO0_8, PIO0_19 | | |
| | SCT0_OUT2 | PIO0_9, PIO0_29 | | |
| | SCT0_OUT3 | PIO0_0, PIO0_10, PIO0_30 | | |
| | SCT0_OUT4 | PIO0_13, PIO1_1, PIO1_10 | | |
| | SCT0_OUT5 | PIO0_14, PIO1_2, PIO1_15 | | |
| | SCT0_OUT6 | PIO0_5, PIO1_3 | | |
| | SCT0_OUT7 | PIO1_4, PIO1_14 | | |
| internal | - | ADC0 trigger | SEQ_A, SEQ_B | Table 414 , Table 415 , Table 429 |

Table 204: Suggested SCT input pin settings

| IOCON bit(s) | Type D pin | Type A pin | Type I pin |
|-----------------|---|---------------------------------------|---------------------------------------|
| 10 | OD: Set to 0 | Same as type D. | I2CFILTER: Set to 1 |
| 9 | SLEW: Set to 0. | Not used, set to 0 | I2CDRIVE: Set to 0. |
| 8 | FILTEROFF: Generally set to 1. | Same as type D. | Same as type D. |
| 7 | DIGIMODE: Set to 1. | Same as type D. | Same as type D. |
| 6 | INVERT: Set to 0. | Same as type D. | Same as type D. |
| 5 | Not used, set to 0. | Same as type D. | I2CSLEW: Set to 1. |
| 4:3 | MODE: Set to 00 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input). | Same as type D. | Not used, set to 00. |
| 2:0 | FUNC: not used, set to 00. Specific pin inputs are directly connected to the SCT. | Same as type D. | Same as type D. |
| General comment | A good choice for an SCT input. | A reasonable choice for an SCT input. | A reasonable choice for an SCT input. |

Table 205: Suggested SCT output pin settings

| IOCON bit(s) | Type D pin | Type A pin | Type I pin |
|-----------------|---|--|----------------------------------|
| 10 | OD: Set to 0 unless open-drain output is desired. | Same as type D. | I2CFILTER: Set to 1 |
| 9 | SLEW: Set to 0. | Not used, set to 0 | I2CDRIVE: Set to 0. |
| 8 | FILTEROFF: Set to 1. | Same as type D. | Same as type D. |
| 7 | DIGIMODE: Set to 1. | Same as type D. | Same as type D. |
| 6 | INVERT: Set to 0. | Same as type D. | Same as type D. |
| 5 | Not used, set to 0. | Same as type D. | I2CSLEW: Set to 1. |
| 4:3 | MODE: set to 00. | Same as type D. | Not used, set to 00. |
| 2:0 | FUNC: Must select the correct function for this peripheral. | Same as type D. | Same as type D. |
| General comment | A good choice for an SCT output. | A reasonable choice for an SCT output. | Not recommended for SCT outputs. |

14.5 General description

The SCTimer/PWM, or in short SCT, is a powerful, flexible timer module capable of creating complex PWM waveforms and performing other advanced timing and control operations with minimal or no CPU intervention.

The SCT can operate as a single 32-bit counter or as two independent, 16-bit counters in uni-directional or bi-directional mode. As with most timers, the SCT supports a selection of match registers against which the count value can be compared, and capture registers where the current count value can be recorded when some pre-defined condition is detected.

An additional feature contributing to the versatility of the SCT is the concept of “events”. The SCT module supports multiple separate events that can be defined by the user based on some combination of parameters including a match on one of the match registers, and/or a transition on one of the SCT inputs or outputs, the direction of count, and other factors.

Every action that the SCT block can perform occurs in direct response to one if these user-defined events without any software overhead. Any event can be enabled to:

- Start, stop, or halt the counter.
- Limit the counter which means to clear the counter in unidirectional mode or change its direction in bi-directional mode.
- Set, clear, or toggle any SCT output.
- Force a capture of the count value into any capture registers.
- Generate an interrupt of DMA request.

The SCT allows the user to group and filter events, thereby selecting some events to be enabled together while others are disabled in a given context. A group of enabled and disabled events can be described as a state, and several states with different sets of enabled and disabled events are allowed. Changing from one state to another is event

driven as well and can therefore happen without software intervention. By defining these states, the SCTimer/PWM provides the means to run entire state machines in hardware with any desired level of complexity to accomplish complex waveform and timing tasks.

In a simple system such as a classical timer/counter with capture and match capabilities, all events that could cause the timer to capture the timer value or toggle a match output are enabled and remain enabled at all times while the counter is running. In this case, no events are filtered and the system is described by one state that does not change. This is the default configuration of the SCT.

In a more complex system, two states could be set up that allow some events in one state and not in the other. An event itself, enabled in both states, can then be used, to move from one state to the other and back while filtering out events in either state. In such a two-state system different waveforms at the SCT output can be created depending on the event history. Changing between states is event-driven and happens without any intervention by the CPU.

Formally, the SCTimer/PWM can be programmed as state machine generator. The ability to perform switching between groups of events provides the SCT the unique capability to be utilized as a highly complex State Machine engine. Events identify the occurrence of conditions that warrant state changes and determine the next state to move to. This provides an extremely powerful control tool - particularly when the SCT inputs and outputs are connected to other on-chip resources (comparators, ADC triggers, other timers etc.) in addition to general-purpose I/O.

In addition to events and states, the SCTimer/PWM provides other enhanced features:

- Four alternative clocking modes including a fully asynchronous mode.
- Selection of any SCT input as a clock source or a clock gate.
- Capability of selecting a “greater-than-or-equal-to” match condition for the purpose of event generation.

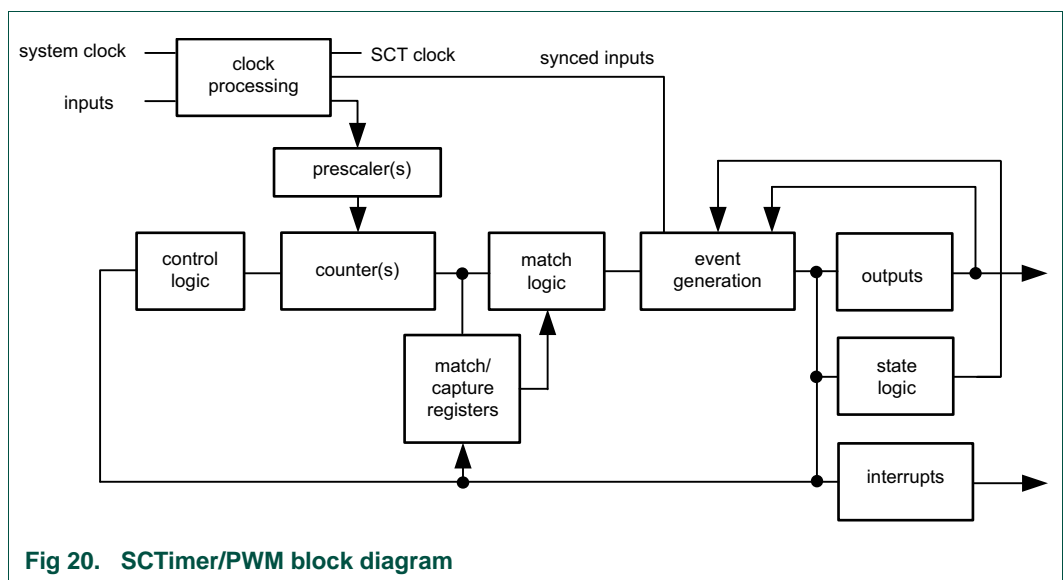


Fig 20. SCTimer/PWM block diagram

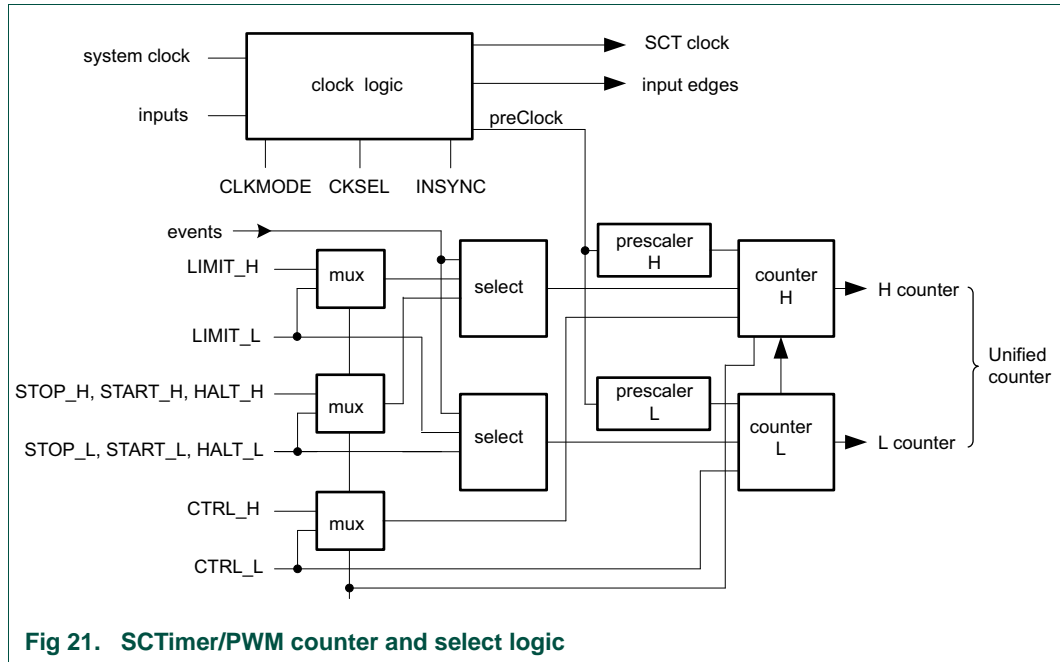


Fig 21. SCTimer/PWM counter and select logic

Remark: In this chapter, the term bus error indicates an SCT response that makes the processor take an exception.

14.6 Register description

The register addresses of the State Configurable Timer are shown in [Table 206](#). For most of the SCT registers, the register function depends on the setting of certain other register bits:

1. The UNIFY bit in the CONFIG register determines whether the SCT is used as one 32-bit register (for operation as one 32-bit counter/timer) or as two 16-bit counter/timers named L and H. The setting of the UNIFY bit is reflected in the register map:
 - UNIFY = 1: Only one register is used (for operation as one 32-bit counter/timer).
 - UNIFY = 0: Access the L and H registers by a 32-bit read or write operation or can be read or written to individually (for operation as two 16-bit counter/timers).

Typically, the UNIFY bit is configured by writing to the CONFIG register before any other registers are accessed.
2. The REGMODEn bits in the REGMODE register determine whether each set of Match/Capture registers uses the match or capture functionality:
 - REGMODEn = 0: Registers operate as match and reload registers.
 - REGMODEn = 1: Registers operate as capture and capture control registers.

Table 206. Register overview: State Configurable Timer SCT/PWM (base address 0x5000 4000)

| Name | Access | Offset | Description | Reset value | Reference |
|--------------------------|--------|-------------------|---|-------------|---------------------------|
| CONFIG | R/W | 0x000 | SCT configuration register | 0x0000 7E00 | Table 207 |
| CTRL | R/W | 0x004 | SCT control register | 0x0004 0004 | Table 208 |
| CTRL_L | R/W | 0x004 | SCT control register low counter 16-bit | 0x0004 0004 | Table 208 |
| CTRL_H | R/W | 0x006 | SCT control register high counter 16-bit | 0x0004 0004 | Table 208 |
| LIMIT | R/W | 0x008 | SCT limit event select register | 0x0000 0000 | Table 209 |
| LIMIT_L | R/W | 0x008 | SCT limit event select register low counter 16-bit | 0x0000 0000 | Table 209 |
| LIMIT_H | R/W | 0x00A | SCT limit event select register high counter 16-bit | 0x0000 0000 | Table 209 |
| HALT | R/W | 0x00C | SCT halt event select register | 0x0000 0000 | Table 210 |
| HALT_L | R/W | 0x00C | SCT halt event select register low counter 16-bit | 0x0000 0000 | Table 210 |
| HALT_H | R/W | 0x00E | SCT halt event select register high counter 16-bit | 0x0000 0000 | Table 210 |
| STOP | R/W | 0x010 | SCT stop event select register | 0x0000 0000 | Table 211 |
| STOP_L | R/W | 0x010 | SCT stop event select register low counter 16-bit | 0x0000 0000 | Table 211 |
| STOP_H | R/W | 0x012 | SCT stop event select register high counter 16-bit | 0x0000 0000 | Table 211 |
| START | R/W | 0x014 | SCT start event select register | 0x0000 0000 | Table 212 |
| START_L | R/W | 0x014 | SCT start event select register low counter 16-bit | 0x0000 0000 | Table 212 |
| START_H | R/W | 0x016 | SCT start event select register high counter 16-bit | 0x0000 0000 | Table 212 |
| COUNT | R/W | 0x040 | SCT counter register | 0x0000 0000 | Table 213 |
| COUNT_L | R/W | 0x040 | SCT counter register low counter 16-bit | 0x0000 0000 | Table 213 |
| COUNT_H | R/W | 0x042 | SCT counter register high counter 16-bit | 0x0000 0000 | Table 213 |
| STATE | R/W | 0x044 | SCT state register | 0x0000 0000 | Table 214 |
| STATE_L | R/W | 0x044 | SCT state register low counter 16-bit | 0x0000 0000 | Table 214 |
| STATE_H | R/W | 0x046 | SCT state register high counter 16-bit | 0x0000 0000 | Table 214 |
| INPUT | RO | 0x048 | SCT input register | 0x0000 0000 | Table 215 |
| REGMODE | R/W | 0x04C | SCT match/capture mode register | 0x0000 0000 | Table 216 |
| REGMODE_L | R/W | 0x04C | SCT match/capture mode register low counter 16-bit | 0x0000 0000 | Table 216 |
| REGMODE_H | R/W | 0x04E | SCT match/capture registers mode register high counter 16-bit | 0x0000 0000 | Table 216 |
| OUTPUT | R/W | 0x050 | SCT output register | 0x0000 0000 | Table 217 |
| OUTPUTDIRCTRL | R/W | 0x054 | SCT output counter direction control register | 0x0000 0000 | Table 218 |
| RES | R/W | 0x058 | SCT conflict resolution register | 0x0000 0000 | Table 219 |
| DMAREQ0 | R/W | 0x05C | SCT DMA request 0 register | 0x0000 0000 | Table 220 |
| DMAREQ1 | R/W | 0x060 | SCT DMA request 1 register | 0x0000 0000 | Table 221 |
| EVEN | R/W | 0x0F0 | SCT event interrupt enable register | 0x0000 0000 | Table 222 |
| EVFLAG | R/W | 0x0F4 | SCT event flag register | 0x0000 0000 | Table 223 |
| CONEN | R/W | 0x0F8 | SCT conflict interrupt enable register | 0x0000 0000 | Table 224 |
| CONFLAG | R/W | 0x0FC | SCT conflict flag register | 0x0000 0000 | Table 225 |
| MATCH0 to MATCH12 | R/W | 0x100 to 0x130 | SCT match value register of match channels 0 to 12; REGMODE0 to REGMODE12 = 0 | 0x0000 0000 | Table 225 |
| MATCH0_L to MATCH12_L | R/W | 0x100 to 0x130 | SCT match value register of match channels 0 to 12; low counter 16-bit; REGMODE0_L to REGMODE12_L = 0 | 0x0000 0000 | Table 225 |

Table 206. Register overview: State Configurable Timer SCT/PWM (base address 0x5000 4000) ...continued

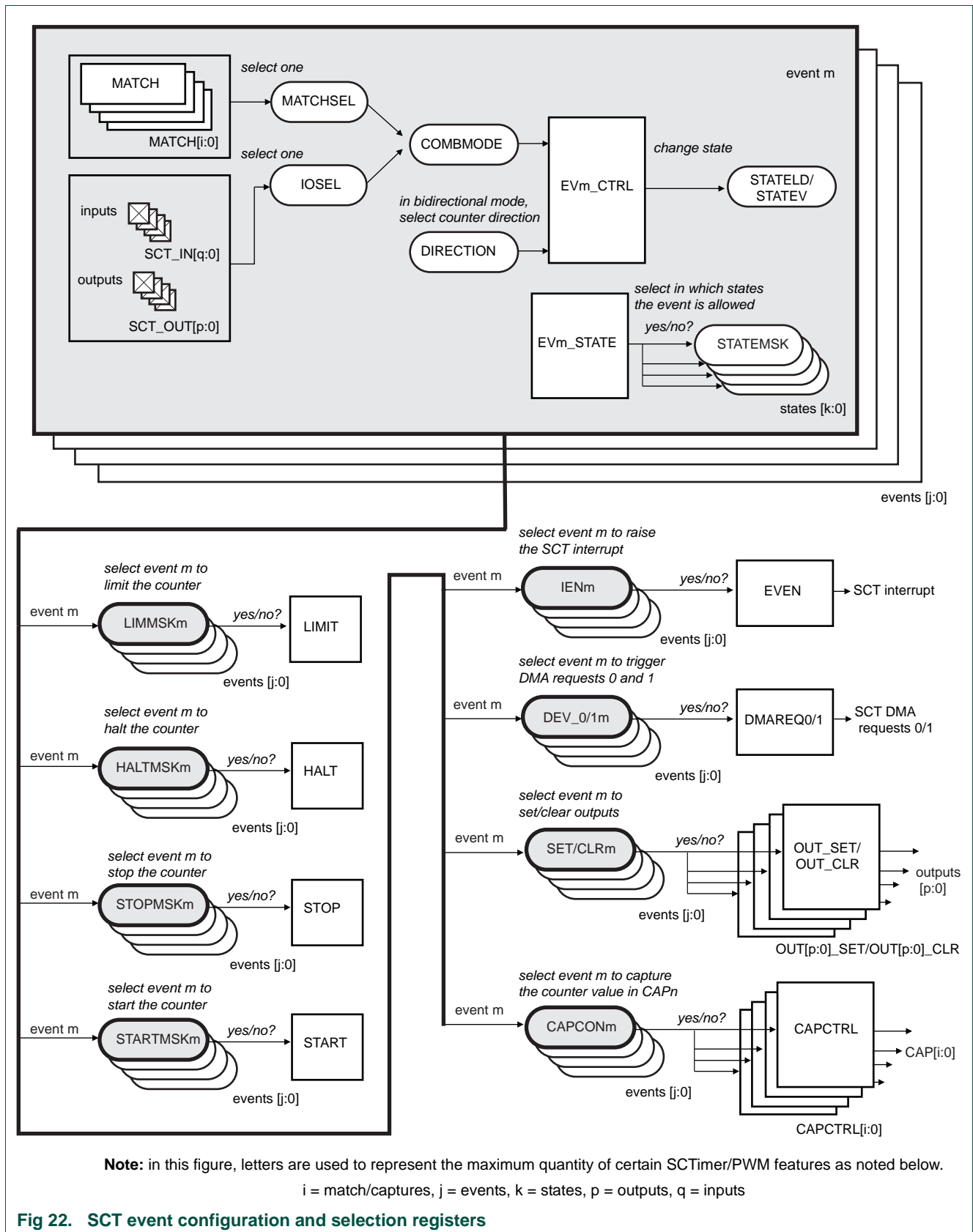
| Name | Access | Offset | Description | Reset value | Reference |
|-----------------------------|--------|----------------|--|-------------|---------------------------|
| MATCH0_H to MATCH12_H | R/W | 0x102 to 0x132 | SCT match value register of match channels 0 to 12; high counter 16-bit; REGMODE0_H to REGMODE12_H = 0 | 0x0000 0000 | Table 225 |
| CAP0 to CAP12 | R/W | 0x100 to 0x130 | SCT capture register of capture channel 0 to 12; REGMODE0 to REGMODE12 = 1 | 0x0000 0000 | Table 227 |
| CAP0_L to CAP12_L | R/W | 0x100 to 0x130 | SCT capture register of capture channel 0 to 12; low counter 16-bit; REGMODE0_L to REGMODE12_L = 1 | 0x0000 0000 | Table 227 |
| CAP0_H to CAP12_H | R/W | 0x102 to 0x132 | SCT capture register of capture channel 0 to 12; high counter 16-bit; REGMODE0_H to REGMODE12_H = 1 | 0x0000 0000 | Table 227 |
| MATCHREL0 to MATCHREL12 | R/W | 0x200 to 0x230 | SCT match reload value register 0 to 12; REGMODE0 = 0 to REGMODE12 = 0 | 0x0000 0000 | Table 228 |
| MATCHREL0_L to MATCHREL12_L | R/W | 0x200 to 0x230 | SCT match reload value register 0 to 12; low counter 16-bit; REGMODE0_L = 0 to REGMODE12_L = 0 | 0x0000 0000 | Table 228 |
| MATCHREL0_H to MATCHREL12_H | R/W | 0x202 to 0x232 | SCT match reload value register 0 to 12; high counter 16-bit; REGMODE0_H = 0 to REGMODE12_H = 0 | 0x0000 0000 | Table 228 |
| CAPCTRL0 to CAPCTRL12 | R/W | 0x200 to 0x230 | SCT capture control register 0 to 12; REGMODE0 = 1 to REGMODE12 = 1 | 0x0000 0000 | Table 229 |
| CAPCTRL0_L to CAPCTRL12_L | R/W | 0x200 to 0x230 | SCT capture control register 0 to 12; low counter 16-bit; REGMODE0_L = 1 to REGMODE12_L = 1 | 0x0000 0000 | Table 229 |
| CAPCTRL0_H to CAPCTRL12_H | R/W | 0x202 to 0x232 | SCT capture control register 0 to 12; high counter 16-bit; REGMODE0 = 1 to REGMODE12 = 1 | 0x0000 0000 | Table 229 |
| EV0_STATE | R/W | 0x300 | SCT event state register 0 | 0x0000 0000 | Table 230 |
| EV0_CTRL | R/W | 0x304 | SCT event control register 0 | 0x0000 0000 | Table 231 |
| EV1_STATE | R/W | 0x308 | SCT event state register 1 | 0x0000 0000 | Table 230 |
| EV1_CTRL | R/W | 0x30C | SCT event control register 1 | 0x0000 0000 | Table 231 |
| EV2_STATE | R/W | 0x310 | SCT event state register 2 | 0x0000 0000 | Table 230 |
| EV2_CTRL | R/W | 0x314 | SCT event control register 2 | 0x0000 0000 | Table 231 |
| EV3_STATE | R/W | 0x318 | SCT event state register 3 | 0x0000 0000 | Table 230 |
| EV3_CTRL | R/W | 0x31C | SCT event control register 3 | 0x0000 0000 | Table 231 |
| EV4_STATE | R/W | 0x320 | SCT event state register 4 | 0x0000 0000 | Table 230 |
| EV4_CTRL | R/W | 0x324 | SCT event control register 4 | 0x0000 0000 | Table 231 |
| EV5_STATE | R/W | 0x328 | SCT event state register 5 | 0x0000 0000 | Table 230 |
| EV5_CTRL | R/W | 0x32C | SCT event control register 5 | 0x0000 0000 | Table 231 |
| EV6_STATE | R/W | 0x330 | SCT event state register 6 | 0x0000 0000 | Table 230 |
| EV6_CTRL | R/W | 0x334 | SCT event control register 6 | 0x0000 0000 | Table 231 |
| EV7_STATE | R/W | 0x338 | SCT event state register 7 | 0x0000 0000 | Table 230 |
| EV7_CTRL | R/W | 0x33C | SCT event control register 7 | 0x0000 0000 | Table 231 |
| EV8_STATE | R/W | 0x340 | SCT event state register 8 | 0x0000 0000 | Table 230 |
| EV8_CTRL | R/W | 0x344 | SCT event control register 8 | 0x0000 0000 | Table 231 |
| EV9_STATE | R/W | 0x348 | SCT event state register 9 | 0x0000 0000 | Table 230 |
| EV9_CTRL | R/W | 0x34C | SCT event control register 9 | 0x0000 0000 | Table 231 |

Table 206. Register overview: State Configurable Timer SCT/PWM (base address 0x5000 4000) ...continued

| Name | Access | Offset | Description | Reset value | Reference |
|------------|--------|--------|-------------------------------|-------------|---------------------------|
| EV10_STATE | R/W | 0x350 | SCT event state register 10 | 0x0000 0000 | Table 230 |
| EV10_CTRL | R/W | 0x354 | SCT event control register 10 | 0x0000 0000 | Table 231 |
| EV11_STATE | R/W | 0x358 | SCT event state register 11 | 0x0000 0000 | Table 230 |
| EV11_CTRL | R/W | 0x35C | SCT event control register 11 | 0x0000 0000 | Table 231 |
| EV12_STATE | R/W | 0x360 | SCT event state register 12 | 0x0000 0000 | Table 230 |
| EV12_CTRL | R/W | 0x364 | SCT event control register 12 | 0x0000 0000 | Table 231 |
| OUT0_SET | R/W | 0x500 | SCT output 0 set register | 0x0000 0000 | Table 232 |
| OUT0_CLR | R/W | 0x504 | SCT output 0 clear register | 0x0000 0000 | Table 233 |
| OUT1_SET | R/W | 0x508 | SCT output 1 set register | 0x0000 0000 | Table 232 |
| OUT1_CLR | R/W | 0x50C | SCT output 1 clear register | 0x0000 0000 | Table 233 |
| OUT2_SET | R/W | 0x510 | SCT output 2 set register | 0x0000 0000 | Table 232 |
| OUT2_CLR | R/W | 0x514 | SCT output 2 clear register | 0x0000 0000 | Table 233 |
| OUT3_SET | R/W | 0x518 | SCT output 3 set register | 0x0000 0000 | Table 232 |
| OUT3_CLR | R/W | 0x51C | SCT output 3 clear register | 0x0000 0000 | Table 233 |
| OUT4_SET | R/W | 0x520 | SCT output 4 set register | 0x0000 0000 | Table 232 |
| OUT4_CLR | R/W | 0x524 | SCT output 4 clear register | 0x0000 0000 | Table 233 |
| OUT5_SET | R/W | 0x528 | SCT output 5 set register | 0x0000 0000 | Table 232 |
| OUT5_CLR | R/W | 0x52C | SCT output 5 clear register | 0x0000 0000 | Table 233 |
| OUT6_SET | R/W | 0x530 | SCT output 6 set register | 0x0000 0000 | Table 232 |
| OUT6_CLR | R/W | 0x534 | SCT output 6 clear register | 0x0000 0000 | Table 233 |
| OUT7_SET | R/W | 0x538 | SCT output 7 set register | 0x0000 0000 | Table 232 |
| OUT7_CLR | R/W | 0x53C | SCT output 7 clear register | 0x0000 0000 | Table 233 |

14.6.1 Register functional grouping

Most SCT registers either configure an event or select an event for a specific action of the counter (or counters) and outputs. [Figure 22](#) shows the registers and register bits that need to be configured for each event.



14.6.1.1 Counter configuration and control registers

The SCT contains two registers for configuring the SCT and monitor and control its operation by software.

- The configuration register (CONFIG) configures the SCT in single, 32-bit counter mode or in dual, 16-bit counter mode, configures the clocking and clock synchronization, and configures automatic limits and the use of reload registers.
- The control register (CTRL) allows to monitor and set the counter direction, and to clear, start, stop, or halt the 32-bit counter or each individual 16-bit counter if in dual-counter mode.

14.6.1.2 Event configuration registers

Each event is associated with two registers:

- One EVn_CTRL register per event to define what triggers the event.
- One EVn_STATE register per event to enable the event.

14.6.1.3 Match and capture registers

The SCT includes a set of registers to store the SCT match or capture values. Each match register is associated with a match reload register which automatically reloads the match register at the beginning of each counter cycle. This register group includes the following registers:

- One REGMODE register per match/capture register to configure each match/capture register for either storing a match value or a capture value.
- A set of match/capture registers with each register, depending on the setting of REGMODE, either storing a match value or a counter value.
- One reload register for each match register.

14.6.1.4 Event select registers for the counter operations

This group contains the registers that select the events which affect the counter. Counter actions are limit, halt, and start or stop and apply to the unified counter or to the two 16-bit counters. Also included is the counter register with the counter value, or values in the dual-counter set-up. This register group includes the following registers:

- LIMIT selects the events that limit the counter.
- START and STOP select events that start or stop the counter.
- HALT selects events that halt the counter: HALT
- COUNT contains the counter value.

The LIMIT, START, STOP, and HALT registers each contain one bit per event that selects for each event whether the event limits, stops, starts, or halts the counter, or counters in dual-counter mode.

In the dual-counter mode, the events can be selected independently for each counter.

14.6.1.5 Event select registers for setting or clearing the outputs

This group contains the registers that select the events which affect the level of each SCT output. Also included are registers to manage conflicts that occur when events try to set or clear the same output. This register group includes the following registers:

- One OUTn_SET register for each output to select the events which set the output.
- One OUTn_CLR register for each output to select the events which clear the output.
- The conflict resolution register which defines an action when more than one event try to control an output at the same time.
- The conflict flag and conflict interrupt enable registers that monitor interrupts arising from output set and clear conflicts.
- The output direction control register that interchanges the set and clear output operation caused by an event in bi-directional mode.

The OUTn_SET and OUTn_CLR registers each contain one bit per event that selects whether the event changes the state a given output n.

In the dual-counter mode, the events can be selected independently for each output.

14.6.1.6 Event select registers for capturing a counter value

This group contains registers that select events which capture the counter value and store it in one of the CAP registers. Each capture register m has one associated CAPCTRLm register which in turn selects the events to capture the counter value.

14.6.1.7 Event select register for initiating DMA transfers

One register is provided for each of the two DMA requests to select the events that can trigger a DMA request.

The DMAREQn register contain one bit for each event that selects whether this event triggers a DMA request. An additional bit enables the DMA trigger when the match registers are reloaded.

14.6.1.8 Interrupt handling registers

The following registers provide flags that are set by events and select the events that when they occur request an interrupt.

- The event flag register provides one flag for each event that is set when the event occurs.
- The event flag interrupt enable register provides one bit for each event to be enabled for the SCT interrupt.

14.6.1.9 Registers for controlling SCT inputs and outputs by software

Two registers are provided that allow software (as opposed to events) to set input and outputs of the SCT:

- The SCT input register to read the state of any of the SCT inputs.
- The SCT output register to set or clear any of the SCT outputs or to read the state of the outputs.

14.6.2 SCT configuration register

This register configures the overall operation of the SCT. Write to this register before any other registers. Only word-writes are permitted to this register. Attempting to write a half-word value results in a bus error.

Table 207. SCT configuration register (CONFIG, address 0x5000 4000) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|------------|-------|--|-------------|
| 0 | UNIFY | | SCT operation | 0 |
| | | 0 | The SCT operates as two 16-bit counters named COUNTER_L and COUNTER_H. | |
| | | 1 | The SCT operates as a unified 32-bit counter. | |
| 2:1 | CLKMODE | | SCT clock mode | 0 |
| | | 0x0 | System Clock Mode. The system clock clocks the entire SCT module including the counter(s) and counter prescalers. | |
| | | 0x1 | Sampled System Clock Mode. The system clock clocks the SCT module, but the counter and prescalers are only enabled to count when the designated edge is detected on the input selected by the CKSEL field. The minimum pulse width on the selected clock-gate input is 1 bus clock period. This mode is the high-performance, sampled-clock mode. | |
| | | 0x2 | SCT Input Clock Mode. The input/edge selected by the CKSEL field clocks the SCT module, including the counters and prescalers, after first being synchronized to the system clock. The minimum pulse width on the clock input is 1 bus clock period. This mode is the low-power, sampled-clock mode. | |
| | | 0x3 | Asynchronous Mode. The entire SCT module is clocked directly by the input/edge selected by the CKSEL field. In this mode, the SCT outputs are switched synchronously to the SCT input clock - not the system clock. The input clock rate must be at least half the system clock rate and can be the same or faster than the system clock. | |
| 6:3 | CKSEL | | SCT clock select. The specific functionality of the designated input/edge is dependent on the CLKMODE bit selection in this register. | 0 |
| | | 0x0 | Rising edges on input 0. | |
| | | 0x1 | Falling edges on input 0. | |
| | | 0x2 | Rising edges on input 1. | |
| | | 0x3 | Falling edges on input 1. | |
| | | 0x4 | Rising edges on input 2. | |
| | | 0x5 | Falling edges on input 2. | |
| | | 0x6 | Rising edges on input 3. | |
| | | 0x7 | Falling edges on input 3. | |
| 7 | NORELAOD_L | - | A 1 in this bit prevents the lower match registers from being reloaded from their respective reload registers. Setting this bit eliminates the need to write to the reload registers MATCHREL if the match values are fixed. Software can write to set or clear this bit at any time. This bit applies to both the higher and lower registers when the UNIFY bit is set. | 0 |
| 8 | NORELOAD_H | - | A 1 in this bit prevents the higher match registers from being reloaded from their respective reload registers. Setting this bit eliminates the need to write to the reload registers MATCHREL if the match values are fixed. Software can write to set or clear this bit at any time. This bit is not used when the UNIFY bit is set. | 0 |

Table 207. SCT configuration register (CONFIG, address 0x5000 4000) bit description ...continued

| Bit | Symbol | Value | Description | Reset value |
|-------|-------------|-------|--|-------------|
| 12:9 | INSYNC | - | Synchronization for input N (bit 9 = input 0, bit 10 = input 1, ..., bit 12 = input 3); all other bits are reserved. A 1 in one of these bits subjects the corresponding input to synchronization to the SCT clock, before it is used to create an event. If an input is known to already be synchronous to the SCT clock, this bit may be set to 0 for faster input response. (Note: The SCT clock is the system clock for CKMODEs 0-2. It is the selected, asynchronous SCT input clock for CKMODE3). Note that the INSYNC field only affects inputs used for event generation. It does not apply to the clock input specified in the CKSEL field. | 1 |
| 16:13 | - | - | Reserved. | - |
| 17 | AUTOLIMIT_L | - | A one in this bit causes a match on match register 0 to be treated as a de-facto LIMIT condition without the need to define an associated event. As with any LIMIT event, this automatic limit causes the counter to be cleared to zero in uni-directional mode or to change the direction of count in bi-directional mode. Software can write to set or clear this bit at any time. This bit applies to both the higher and lower registers when the UNIFY bit is set. | 0 |
| 18 | AUTOLIMIT_H | - | A one in this bit will cause a match on match register 0 to be treated as a de-facto LIMIT condition without the need to define an associated event. As with any LIMIT event, this automatic limit causes the counter to be cleared to zero in uni-directional mode or to change the direction of count in bi-directional mode. Software can write to set or clear this bit at any time. This bit is not used when the UNIFY bit is set. | 0 |
| 31:19 | - | - | Reserved | - |

14.6.3 SCT control register

If bit UNIFY = 1 in the CONFIG register, only the _L bits are used.

If bit UNIFY = 0 in the CONFIG register, this register can be written to as two registers CTRL_L and CTRL_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

All bits in this register can be written to when the counter is stopped or halted. When the counter is running, the only bits that can be written are STOP or HALT. (Other bits can be written in a subsequent write after HALT is set to 1.)

Remark: If CLKMODE = 0x3 is selected, wait at least 12 system clock cycles between a write access to the H, L or unified version of this register and the next write access. This restriction does not apply when writing to the HALT bit or bits and then writing to the CTRL register again to restart the counters - for example because software must update the MATCH register, which is only allowed when the counters are halted.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Table 208. SCT control register (CTRL, address 0x5000 4004) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|----------|-------|---|-------------|
| 0 | DOWN_L | - | This bit is 1 when the L or unified counter is counting down. Hardware sets this bit when the counter is counting up, counter limit occurs, and BIDIR = 1. Hardware clears this bit when the counter is counting down and a limit condition occurs or when the counter reaches 0. | 0 |
| 1 | STOP_L | - | When this bit is 1 and HALT is 0, the L or unified counter does not run, but I/O events related to the counter can occur. If a designated start event occurs, this bit is cleared and counting resumes. | 0 |
| 2 | HALT_L | - | When this bit is 1, the L or unified counter does not run and no events can occur. A reset sets this bit. When the HALT_L bit is one, the STOP_L bit is cleared. It is possible to remove the halt condition while keeping the SCT in the stop condition (not running) with a single write to this register to simultaneously clear the HALT bit and set the STOP bit. Remark: Once set, only software can clear this bit to restore counter operation. This bit is set on reset. | 1 |
| 3 | CLRCTR_L | - | Writing a 1 to this bit clears the L or unified counter. This bit always reads as 0. | 0 |
| 4 | BIDIR_L | - | L or unified counter direction select | 0 |
| | | 0 | Up. The counter counts up to a limit condition, then is cleared to zero. | |
| | | 1 | Up-down. The counter counts up to a limit, then counts down to a limit condition or to 0. | |
| 12:5 | PRE_L | - | Specifies the factor by which the SCT clock is prescaled to produce the L or unified counter clock. The counter clock is clocked at the rate of the SCT clock divided by PRE_L+1. Remark: Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value. | 0 |
| 15:13 | - | - | Reserved | - |
| 16 | DOWN_H | - | This bit is 1 when the H counter is counting down. Hardware sets this bit when the counter is counting, a counter limit condition occurs, and BIDIR is 1. Hardware clears this bit when the counter is counting down and a limit condition occurs or when the counter reaches 0. | 0 |
| 17 | STOP_H | - | When this bit is 1 and HALT is 0, the H counter does not, run but I/O events related to the counter can occur. If such an event matches the mask in the Start register, this bit is cleared and counting resumes. | 0 |
| 18 | HALT_H | - | When this bit is 1, the H counter does not run and no events can occur. A reset sets this bit. When the HALT_H bit is one, the STOP_H bit is cleared. It is possible to remove the halt condition while keeping the SCT in the stop condition (not running) with a single write to this register to simultaneously clear the HALT bit and set the STOP bit. Remark: Once set, this bit can only be cleared by software to restore counter operation. This bit is set on reset. | 1 |
| 19 | CLRCTR_H | - | Writing a 1 to this bit clears the H counter. This bit always reads as 0. | 0 |

Table 208. SCT control register (CTRL, address 0x5000 4004) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|---------|-------|--|-------------|
| 20 | BIDIR_H | | Direction select | 0 |
| | | 0 | The H counter counts up to its limit condition, then is cleared to zero. | |
| | | 1 | The H counter counts up to its limit, then counts down to a limit condition or to 0. | |
| 28:21 | PRE_H | - | Specifies the factor by which the SCT clock is prescaled to produce the H counter clock. The counter clock is clocked at the rate of the SCT clock divided by PRELH+1. Remark: Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value. | 0 |
| 31:29 | - | | Reserved | - |

14.6.4 SCT limit event select register

The running counter can be limited by an event. When any of the events selected in this register occur, the counter is cleared to zero from its current value or changes counting direction if in bi-directional mode.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit causes its associated event to serve as a LIMIT event. When any limit event occurs, the counter is reset to zero in uni-directional mode or changes its direction of count in bi-directional mode and keeps running. To define the actual limiting event (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: Counting up to all ones or counting down to zero is always equivalent to a limit event occurring.

Note that in addition to using this register to specify events that serve as limits, it is also possible to automatically cause a limit condition whenever a match register 0 match occurs. This eliminates the need to define an event for the sole purpose of creating a limit. The AUTOLIMITL and AUTOLIMITH bits in the configuration register enable/disable this feature (see [Table 207](#)).

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers LIMIT_L and LIMIT_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 209. SCT limit event select register (LIMIT, address 0x5000 4008) bit description

| Bit | Symbol | Description | Reset value |
|-------|----------|--|-------------|
| 15:0 | LIMMSK_L | If bit n is one, event n is used as a counter limit for the L or unified counter (event 0 = bit 0, event 1 = bit 1, ...). The number of bits = number of events in this SCT. | 0 |
| 31:16 | LIMMSK_H | If bit n is one, event n is used as a counter limit for the H counter (event 0 = bit 16, event 1 = bit 17, ...). The number of bits = number of events in this SCT. | 0 |

14.6.5 SCT halt event select register

The running counter can be disabled (halted) by an event. When any of the events selected in this register occur, the counter stops running and all further events are disabled.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a HALT event. To define the actual events that cause the counter to halt (a match, an I/O pin toggle, etc.), see the EVn_CTRL registers.

Remark: A HALT condition can only be removed when software clears the HALT bit in the CTRL register (Table 208).

If UNIFY = 1 in the CONFIG register, only the L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers HALT_L and HALT_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 210. SCT halt event select register (HALT, address 0x5000 400C) bit description

| Bit | Symbol | Description | Reset value |
|-------|-----------|---|-------------|
| 15:0 | HALTMSK_L | If bit n is one, event n sets the HALT_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, ...). The number of bits = number of events in this SCT. | 0 |
| 31:16 | HALTMSK_H | If bit n is one, event n sets the HALT_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, ...). The number of bits = number of events in this SCT. | 0 |

14.6.6 SCT stop event select register

The running counter can be stopped by an event. When any of the events selected in this register occur, counting is suspended, that is the counter stops running and remains at its current value. Event generation remains enabled, and any event selected in the START register such as an I/O event or an event generated by the other counter can restart the counter.

This register specifies which events stop the counter. Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a STOP event. To define the actual event that causes the counter to stop (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: Software can stop and restart the counter by writing to the CTRL register.

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers STOPT_L and STOP_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 211. SCT stop event select register (STOP, address 0x5000 4010) bit description

| Bit | Symbol | Description | Reset value |
|-------|-----------|---|-------------|
| 15:0 | STOPMSK_L | If bit n is one, event n sets the STOP_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, ...). The number of bits = number of events in this SCT. | 0 |
| 31:16 | STOPMSK_H | If bit n is one, event n sets the STOP_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, ...). The number of bits = number of events in this SCT. | 0 |

14.6.7 SCT start event select register

The stopped counter can be re-started by an event. When any of the events selected in this register occur, counting is restarted from the current counter value.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a START event. When any START event occurs, hardware will clear the STOP bit in the Control Register CTRL. Note that a START event has no effect on the HALT bit. Only software can remove a HALT condition. To define the actual event that starts the counter (an I/O pin toggle or an event generated by the other running counter in dual-counter mode), see the EVn_CTRL register.

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers START_L and START_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 212. SCT start event select register (START, address 0x5000 4014) bit description

| Bit | Symbol | Description | Reset value |
|-------|------------|---|-------------|
| 15:0 | STARTMSK_L | If bit n is one, event n clears the STOP_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, ...). The number of bits = number of events in this SCT. | 0 |
| 31:16 | STARTMSK_H | If bit n is one, event n clears the STOP_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, ...). The number of bits = number of events in this SCT. | 0 |

14.6.8 SCT counter register

If UNIFY = 1 in the CONFIG register, the counter is a unified 32-bit register and both the _L and _H bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers COUNT_L and COUNT_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation. In this case, the L and H registers count independently under the control of the other registers.

Writing to the COUNT_L, COUNT_H, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register). Attempting to write to the counter when it is not halted causes a bus error. Software can read the counter registers at any time.

Table 213. SCT counter register (COUNT, address 0x5000 4040) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 15:0 | CTR_L | When UNIFY = 0, read or write the 16-bit L counter value. When UNIFY = 1, read or write the lower 16 bits of the 32-bit unified counter. | 0 |
| 31:16 | CTR_H | When UNIFY = 0, read or write the 16-bit H counter value. When UNIFY = 1, read or write the upper 16 bits of the 32-bit unified counter. | 0 |

14.6.9 SCT state register

Each group of enabled and disabled events is assigned a number called the state variable. For example, a state variable with a value of 0 could have events 0, 2, and 3 enabled and all other events disabled. A state variable with the value of 1 could have events 1, 4, and 5 enabled and all others disabled.

Remark: The EVm_STATE registers define which event is enabled in each group.

Software can read the state associated with a counter at any time. Writing to the STATE_L, STATE_H, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register).

The state variable is the main feature that distinguishes the SCTimer/PWM from other counter/timer/ PWM blocks. Events can be made to occur only in certain states. Events, in turn, can perform the following actions:

- set and clear outputs
- limit, stop, and start the counter
- cause interrupts and DMA requests
- modify the state variable

The value of a state variable is completely under the control of the application. If an application does not use states, the value of the state variable remains zero, which is the default value.

A state variable can be used to track and control multiple cycles of the associated counter in any desired operational sequence. The state variable is logically associated with a state machine diagram which represents the SCT configuration. See [Section 14.6.24](#) and [14.6.25](#) for more about the relationship between states and events.

The STATELD/STADEV fields in the event control registers of all defined events set all possible values for the state variable. The change of the state variable during multiple counter cycles reflects how the associated state machine moves from one state to the next.

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers STATE_L and STATE_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 214. SCT state register (STATE, address 0x5000 4044) bit description

| Bit | Symbol | Description | Reset value |
|-------|---------|-----------------|-------------|
| 4:0 | STATE_L | State variable. | 0 |
| 15:5 | - | Reserved. | - |
| 20:16 | STATE_H | State variable. | 0 |
| 31:21 | - | Reserved. | - |

14.6.10 SCT input register

Software can read the state of the SCT inputs in this read-only register in slightly different forms.

1. The AIN bit displays the state of the input captured on each rising edge of the SCT clock. This corresponds to a nearly direct read-out of the input but can cause spurious fluctuations in case of an asynchronous input signal.
2. The SIN bit displays the form of the input as it is used for event detection. This may include additional stages of synchronization, depending on what is specified for that input in the INSYNC field in the CONFIG register:
 - If the INSYNC bit is set for the input, the input is triple-synchronized to the SCT clock resulting in a stable signal that is delayed by three SCT clock cycles.
 - If the INSYNC bit is not set, the SIN bit value is identical to the AIN bit value.

Table 215. SCT input register (INPUT, address 0x5000 4048) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 0 | AIN0 | Input 0 state. Input 0 state on the last SCT clock edge. | - |
| 1 | AIN1 | Input 1 state. Input 1 state on the last SCT clock edge. | - |
| 2 | AIN2 | Input 2 state. Input 2 state on the last SCT clock edge. | - |
| 3 | AIN3 | Input 3 state. Input 3 state on the last SCT clock edge. | - |
| 15:4 | AIN... | Input state for the remainder of states implemented in this SCT. | - |
| 16 | SIN0 | Input 0 state. Input 0 state following the synchronization specified by INSYNC0. | - |
| 17 | SIN1 | Input 1 state. Input 1 state following the synchronization specified by INSYNC0. | - |
| 18 | SIN2 | Input 2 state. Input 2 state following the synchronization specified by INSYNC0. | - |
| 19 | SIN3 | Input 3 state. Input 3 state following the synchronization specified by INSYNC0. | - |
| 31:20 | SIN... | Input state for the remainder of states implemented in this SCT. | - |

14.6.11 SCT match/capture mode register

If UNIFY = 1 in the CONFIG register, only the _L bits of this register are used. In this case, REGMODE_H is not used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers REGMODE_L and REGMODE_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation. The _L bits/register control the L match/capture registers, and the _H bits/register control the H match/capture registers.

The SCT contains multiple Match/Capture registers. The Register Mode register selects whether each register acts as a Match register (see [Section 14.6.20](#)) or as a Capture register (see [Section 14.6.21](#)). Each Match/Capture register has an accompanying register which functions as a Reload register when the primary register is used as a Match register ([Section 14.6.22](#)) or as a Capture-Control register when the register is used as a capture register ([Section 14.6.23](#)). REGMODE_H is used only when the UNIFY bit is 0.

Table 216. SCT match/capture mode register (REGMODE, address 0x5000 404C) bit description

| Bit | Symbol | Description | Reset value |
|-------|----------|--|-------------|
| 15:0 | REGMOD_L | Each bit controls one match/capture register (register 0 = bit 0, register 1 = bit 1, ...). The number of bits = number of match/captures in this SCT. 0 = register operates as match register. 1 = register operates as capture register. | 0 |
| 31:16 | REGMOD_H | Each bit controls one match/capture register (register 0 = bit 16, register 1 = bit 17, ...). The number of bits = number of match/captures in this SCT. 0 = register operates as match registers. 1 = register operates as capture registers. | 0 |

14.6.12 SCT output register

Each SCT output has a corresponding bit in this register to allow software to control the output state directly or read its current state.

While the counter is running, outputs are set, cleared, or toggled only by events. However, using this register, software can write to any of the output registers when both counters are halted to control the outputs directly. Writing to the OUT register is only allowed when

all counters (L-counter, H-counter, or unified counter) are halted (HALT bits are set to 1 in the CTRL register).

Software can read this register at any time to sense the state of the outputs.

Table 217. SCT output register (OUTPUT, address 0x5000 4050) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 15:0 | OUT | Writing a 1 to bit n forces the corresponding output HIGH. Writing a 0 forces the corresponding output LOW (output 0 = bit 0, output 1 = bit 1, ...). The number of bits = number of outputs in this SCT. | 0 |
| 31:16 | - | Reserved | - |

14.6.13 SCT bi-directional output control register

For bi-directional mode, this register specifies (for each output) the impact of the counting direction on the meaning of set and clear operations on the output (see [Section 14.6.26](#) and [Section 14.6.27](#)). The purpose of this register is to facilitate the creation of center-aligned output waveforms without the need to define additional events.

Table 218. SCT bidirectional output control register (OUTPUTDIRCTRL, address 0x5000 4054) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|-----------|-------|--|-------------|
| 1:0 | SETCLR0 | | Set/clear operation on output 0. Value 0x3 is reserved. Do not program this value. | 0 |
| | | 0x0 | Set and clear do not depend on the direction of any counter. | |
| | | 0x1 | Set and clear are reversed when counter L or the unified counter is counting down. | |
| | | 0x2 | Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1. | |
| 3:2 | SETCLR1 | | Set/clear operation on output 1. Value 0x3 is reserved. Do not program this value. | 0 |
| | | 0x0 | Set and clear do not depend on the direction of any counter. | |
| | | 0x1 | Set and clear are reversed when counter L or the unified counter is counting down. | |
| | | 0x2 | Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1. | |
| 5:4 | SETCLR2 | | Set/clear operation on output 2. Value 0x3 is reserved. Do not program this value. | 0 |
| | | 0x0 | Set and clear do not depend on the direction of any counter. | |
| | | 0x1 | Set and clear are reversed when counter L or the unified counter is counting down. | |
| | | 0x2 | Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1. | |
| 7:6 | SETCLR3 | | Set/clear operation on output 3. Value 0x3 is reserved. Do not program this value. | 0 |
| | | 0x0 | Set and clear do not depend on the direction of any counter. | |
| | | 0x1 | Set and clear are reversed when counter L or the unified counter is counting down. | |
| | | 0x2 | Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1. | |
| 9:8 | SETCLR4 | | Set/clear operation on output 4. Value 0x3 is reserved. Do not program this value. | 0 |
| | | 0x0 | Set and clear do not depend on the direction of any counter. | |
| | | 0x1 | Set and clear are reversed when counter L or the unified counter is counting down. | |
| | | 0x2 | Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1. | |
| 31:10 | SETCLR... | | Set/clear operation controls for the remainder of outputs on this SCT. | 0 |

14.6.14 SCT conflict resolution register

The output conflict resolution register specifies what action should be taken if multiple events (or even the same event) dictate that a given output should be both set and cleared at the same time.

To enable an event to toggle an output each time the event occurs, set the bits for that event in both the OUTn_SET and OUTn_CLR registers and set the On_RES value to 0x3 in this register.

Table 219. SCT conflict resolution register (RES, address 0x5000 4058) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|---------|-------|---|-------------|
| 1:0 | O0RES | | Effect of simultaneous set and clear on output 0. | 0 |
| | | 0x0 | No change. | |
| | | 0x1 | Set output (or clear based on the SETCLR0 field in the OUTPUTDIRCTRL register). | |
| | | 0x2 | Clear output (or set based on the SETCLR0 field). | |
| | | 0x3 | Toggle output. | |
| 3:2 | O1RES | | Effect of simultaneous set and clear on output 1. | 0 |
| | | 0x0 | No change. | |
| | | 0x1 | Set output (or clear based on the SETCLR1 field in the OUTPUTDIRCTRL register). | |
| | | 0x2 | Clear output (or set based on the SETCLR1 field). | |
| | | 0x3 | Toggle output. | |
| 5:4 | O2RES | | Effect of simultaneous set and clear on output 2. | 0 |
| | | 0x0 | No change. | |
| | | 0x1 | Set output (or clear based on the SETCLR2 field in the OUTPUTDIRCTRL register). | |
| | | 0x2 | Clear output n (or set based on the SETCLR2 field). | |
| | | 0x3 | Toggle output. | |
| 7:6 | O3RES | | Effect of simultaneous set and clear on output 3. | 0 |
| | | 0x0 | No change. | |
| | | 0x1 | Set output (or clear based on the SETCLR3 field in the OUTPUTDIRCTRL register). | |
| | | 0x2 | Clear output (or set based on the SETCLR3 field). | |
| | | 0x3 | Toggle output. | |
| 9:8 | O4RES | | Effect of simultaneous set and clear on output 4. | 0 |
| | | 0x0 | No change. | |
| | | 0x1 | Set output (or clear based on the SETCLR4 field in the OUTPUTDIRCTRL register). | |
| | | 0x2 | Clear output (or set based on the SETCLR4 field). | |
| | | 0x3 | Toggle output. | |
| 31:10 | O...RES | | Resolution controls for the remainder of outputs on this SCT. | 0 |

14.6.15 SCT DMA request 0 and 1 registers

The SCT includes two DMA request outputs. These registers enable the DMA requests to be triggered when a particular event occurs or when counter Match registers are loaded from its Reload registers. The DMA request registers are word-write only. Attempting to write a half-word value to these registers result in a bus error.

Event-triggered DMA requests are particularly useful for launching DMA activity to or from other peripherals under the control of the SCT.

Table 220. SCT DMA 0 request register (DMAREQ0, address 0x5000 405C) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 15:0 | DEV_0 | If bit n is one, event n triggers DMA request 0 (event 0 = bit 0, event 1 = bit 1, ...). The number of bits = number of events in this SCT. | 0 |
| 29:16 | - | Reserved | - |
| 30 | DRL0 | A 1 in this bit triggers DMA request 0 when it loads the MATCH_L/Unified registers from the RELOAD_L/Unified registers. | 0 |
| 31 | DRQ0 | This read-only bit indicates the state of DMA Request 0 | 0 |

Table 221. SCT DMA 1 request register (DMAREQ1, address 0x5000 4060) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 15:0 | DEV_1 | If bit n is one, event n triggers DMA request 1 (event 0 = bit 0, event 1 = bit 1, ...). The number of bits = number of events in this SCT. | 0 |
| 29:16 | - | Reserved | - |
| 30 | DRL1 | A 1 in this bit triggers DMA request 1 when it loads the Match L/Unified registers from the Reload L/Unified registers. | 0 |
| 31 | DRQ1 | This read-only bit indicates the state of DMA Request 1. | 0 |

14.6.16 SCT event interrupt enable register

This register enables flags to request an interrupt if the FLAGn bit in the SCT event flag register ([Section 14.6.17](#)) is also set.

Table 222. SCT event interrupt enable register (EVEN, address 0x5000 40F0) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 15:0 | IEN | The SCT requests an interrupt when bit n of this register and the event flag register are both one (event 0 = bit 0, event 1 = bit 1, ...). The number of bits = number of events in this SCT. | 0 |
| 31:16 | - | Reserved | - |

14.6.17 SCT event flag register

This register records events. Writing ones to this register clears the corresponding flags and negates the SCT interrupt request if all enabled flag register bits are zero.

Table 223. SCT event flag register (EVFLAG, address 0x5000 40F4) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 15:0 | FLAG | Bit n is one if event n has occurred since reset or a 1 was last written to this bit (event 0 = bit 0, event 1 = bit 1, ...). The number of bits = number of events in this SCT. | 0 |
| 31:16 | - | Reserved | - |

14.6.18 SCT conflict interrupt enable register

This register enables the no-change conflict events specified in the SCT conflict resolution register to generate an interrupt request.

Table 224. SCT conflict interrupt enable register (CONEN, address 0x5000 40F8) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 15:0 | NCEN | The SCT requests an interrupt when bit n of this register and the SCT conflict flag register are both one (output 0 = bit 0, output 1 = bit 1, ...). The number of bits = number of outputs in this SCT. | 0 |
| 31:16 | - | Reserved | |

14.6.19 SCT conflict flag register

This register records a no-change conflict occurrence and provides details of a bus error. Writing ones to the NCFLAG bits clears the corresponding read bits and negates the SCT interrupt request if all enabled Flag bits are zero.

Table 225. SCT conflict flag register (CONFLAG, address 0x5000 40FC) bit description

| Bit | Symbol | Description | Reset value |
|-------|---------|---|-------------|
| 15:0 | NCFLAG | Bit n is one if a no-change conflict event occurred on output n since reset or a 1 was last written to this bit (output 0 = bit 0, output 1 = bit 1, ...). The number of bits = number of outputs in this SCT. | 0 |
| 29:16 | - | Reserved. | - |
| 30 | BUSERRL | The most recent bus error from this SCT involved writing CTR L/Unified, STATE L/Unified, MATCH L/Unified, or the Output register when the L/U counter was not halted. A word write to certain L and H registers can be half successful and half unsuccessful. | 0 |
| 31 | BUSERRH | The most recent bus error from this SCT involved writing CTR H, STATE H, MATCH H, or the Output register when the H counter was not halted. | 0 |

14.6.20 SCT match registers 0 to 12 (REGMODEn bit = 0)

Match registers are compared to the counters to help create events. When the UNIFY bit is 0, the L and H registers are independently compared to the L and H counters. When UNIFY is 1, the combined L and H registers hold a 32-bit value that is compared to the unified counter. A Match can only occur in a clock in which the counter is running (STOP and HALT are both 0).

Match registers can be read at any time. Writing to the MATCH_L, MATCH_H, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register). Match events occur in the SCT clock in which the counter is (or would be) incremented to the next value. When a Match event limits its counter as described in [Section 14.6.4](#), the value in the Match register is the last value of the counter before it is cleared to zero (or decremented if BIDIR is 1).

There is no “write-through” from Reload registers to Match registers. Before starting a counter, software can write one value to the Match register used in the first cycle of the counter and a different value to the corresponding Match Reload register used in the second cycle.

Table 226. SCT match registers 0 to 12 (MATCH[0:12], address 0x5000 4100 (MATCH0) to 0x5000 4130 (MATCH12)) bit description (REGMODEn bit = 0)

| Bit | Symbol | Description | Reset value |
|-------|----------|--|-------------|
| 15:0 | MATCHn_L | When UNIFY = 0, read or write the 16-bit value to be compared to the L counter. When UNIFY = 1, read or write the lower 16 bits of the 32-bit value to be compared to the unified counter. | 0 |
| 31:16 | MATCHn_H | When UNIFY = 0, read or write the 16-bit value to be compared to the H counter. When UNIFY = 1, read or write the upper 16 bits of the 32-bit value to be compared to the unified counter. | 0 |

14.6.21 SCT capture registers 0 to 12 (REGMODEn bit = 1)

These registers allow software to record the counter values upon occurrence of the events selected by the corresponding Capture Control registers occurred.

Table 227. SCT capture registers 0 to 12 (CAP[0:12], address 0x5000 4100 (CAP0) to 0x5000 4130 (CAP12)) bit description (REGMODEn bit = 1)

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 15:0 | CAPn_L | When UNIFY = 0, read the 16-bit counter value at which this register was last captured. When UNIFY = 1, read the lower 16 bits of the 32-bit value at which this register was last captured. | 0 |
| 31:16 | CAPn_H | When UNIFY = 0, read the 16-bit counter value at which this register was last captured. When UNIFY = 1, read the upper 16 bits of the 32-bit value at which this register was last captured. | 0 |

14.6.22 SCT match reload registers 0 to 12 (REGMODEn bit = 0)

A Match register (L, H, or unified 32-bit) is loaded from its corresponding Reload register at the start of each new counter cycle, that is

- when BIDIR = 0 and the counter is cleared to zero upon reaching its limit condition.
- when BIDIR = 1 and the counter counts down to 0, unless the appropriate NORELOAD bit is set in the CFG register.

Table 228. SCT match reload registers 0 to 12 (MATCHREL[0:12], address 0x5000 4200 (MATCHRELO) to 0x5000 4230 (MATCHREL12)) bit description (REGMODEn bit = 0)

| Bit | Symbol | Description | Reset value |
|-------|-----------|--|-------------|
| 15:0 | RELOADn_L | When UNIFY = 0, specifies the 16-bit value to be loaded into the MATCHn_L register. When UNIFY = 1, specifies the lower 16 bits of the 32-bit value to be loaded into the MATCHn register. | 0 |
| 31:16 | RELOADn_H | When UNIFY = 0, specifies the 16-bit to be loaded into the MATCHn_H register. When UNIFY = 1, specifies the upper 16 bits of the 32-bit value to be loaded into the MATCHn register. | 0 |

14.6.23 SCT capture control registers 0 to 12 (REGMODEn bit = 1)

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers CAPCTRLn_L and CAPCTRLn_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Based on a selected event, the capture registers can be loaded with the current counter value when the event occurs.

Each Capture Control register (L, H, or unified 32-bit) controls which events cause the load of corresponding Capture register from the counter.

Table 229. SCT capture control registers 0 to 12 (CAPCTRL[0:12], address 0x5000 4200 (CAPCTRL0) to 0x5000 4230 (CAPCTRL12)) bit description (REGMODEn bit = 1)

| Bit | Symbol | Description | Reset value |
|-------|-----------|--|-------------|
| 15:0 | CAPCONn_L | If bit m is one, event m causes the CAPn_L (UNIFY = 0) or the CAPn (UNIFY = 1) register to be loaded (event 0 = bit 0, event 1 = bit 1, ...). The number of bits = number of match/captures in this SCT. | 0 |
| 31:16 | CAPCONn_H | If bit m is one, event m causes the CAPn_H (UNIFY = 0) register to be loaded (event 0 = bit 16, event 1 = bit 17, ...). The number of bits = number of match/captures in this SCT. | 0 |

14.6.24 SCT event enable registers 0 to 12

Each event can be enabled in some contexts (or states) and disabled in others. Each event defined in the EV_CTRL register has one associated event enable register that can enable or disable the event for each available state.

Each event has one associated SCT event state mask register that allow this event to happen in one or more states of the counter selected by the HEVENT bit in the corresponding EVn_CTRL register.

An event n is disabled when its EVn_STATE register contains all zeros, since it is masked regardless of the current state.

In simple applications that do not use states, write 0x01 to this register to enable each event in exactly one state. Since the state doesn't change (that is, the state variable always remains at its reset value of 0), writing 0x01 permanently enables this event.

Table 230. SCT event state mask registers 0 to 12 (EV[0:12]_STATE, addresses 0x5000 4300 (EV0_STATE) to 0x5000 4360 (EV12_STATE)) bit description

| Bit | Symbol | Description | Reset value |
|-------|-----------|---|-------------|
| 15:0 | STATEMSKn | If bit m is one, event n happens in state m of the counter selected by the HEVENT bit (n = event number, m = state number; state 0 = bit 0, state 1 = bit 1, ...). The number of bits = number of states in this SCT. | 0 |
| 31:16 | - | Reserved. | - |

14.6.25 SCT event control registers 0 to 12

This register defines the conditions for an event to occur based on the counter values or input and output states. Once the event is configured, it can be selected to trigger multiple actions (for example stop the counter and toggle an output) unless the event is blocked in the current state of the SCT or the counter is halted. To block a particular event from occurring, use the EV_STATE register. To block all events for a given counter, set the HALT bit in the CTRL register or select an event to halt the counter.

An event can be programmed to occur based on a selected input or output edge or level and/or based on its counter value matching a selected match register. In bi-directional mode, events can also be enabled based on the direction of count.

When the UNIFY bit is 0, each event is associated with a particular counter by the HEVENT bit in its event control register. An event is permanently disabled when its event state mask register contains all 0s.

Each event can modify its counter STATE value. If more than one event associated with the same counter occurs in a given clock cycle, only the state change specified for the highest-numbered event among them takes place. Other actions dictated by any simultaneously occurring events all take place.

Table 231. SCT event control register 0 to 12 (EV[0:12]_CTRL, address 0x5000 4304 (EV0_CTRL) to 0x5000 4364 (EV12_CTRL)) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|----------|-------|--|-------------|
| 3:0 | MATCHSEL | - | Selects the Match register associated with this event (if any). A match can occur only when the counter selected by the HEVENT bit is running. | 0 |
| 4 | HEVENT | - | Select L/H counter. Do not set this bit if UNIFY = 1. | 0 |
| | | 0 | Selects the L state and the L match register selected by MATCHSEL. | |
| | | 1 | Selects the H state and the H match register selected by MATCHSEL. | |
| 5 | OUTSEL | - | Input/output select | 0 |
| | | 0 | Selects the inputs elected by IOSEL. | |
| | | 1 | Selects the outputs selected by IOSEL. | |
| 9:6 | IOSEL | - | Selects the input or output signal number (0 to 3 for inputs or 0 to 5 for outputs) associated with this event (if any). Do not select an input in this register, if CKMODE is 1x. In this case the clock input is an implicit ingredient of every event. | 0 |
| 11:10 | IOCOND | - | Selects the I/O condition for event n. (The detection of edges on outputs lag the conditions that switch the outputs by one SCT clock). In order to guarantee proper edge/state detection, an input must have a minimum pulse width of at least one SCT clock period . | 0 |
| | | 0x0 | LOW | |
| | | 0x1 | Rise | |
| | | 0x2 | Fall | |
| | | 0x3 | HIGH | |
| 13:12 | COMBMODE | - | Selects how the specified match and I/O condition are used and combined. | 0 |
| | | 0x0 | OR. The event occurs when either the specified match or I/O condition occurs. | |
| | | 0x1 | MATCH. Uses the specified match only. | |
| | | 0x2 | IO. Uses the specified I/O condition only. | |
| | | 0x3 | AND. The event occurs when the specified match and I/O condition occur simultaneously. | |
| 14 | STATELD | - | This bit controls how the STATEV value modifies the state selected by HEVENT when this event is the highest-numbered event occurring for that state. | 0 |
| | | 0 | STATEV value is added into STATE (the carry-out is ignored). | |
| | | 1 | STATEV value is loaded into STATE. | |
| 19:15 | STATEV | - | This value is loaded into or added to the state selected by HEVENT, depending on STATELD, when this event is the highest-numbered event occurring for that state. If STATELD and STATEV are both zero, there is no change to the STATE value. | 0 |

Table 231. SCT event control register 0 to 12 (EV[0:12]_CTRL, address 0x5000 4304 (EV0_CTRL) to 0x5000 4364 (EV12_CTRL)) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|-----------|-------|--|-------------|
| 20 | MATCHMEM | | If this bit is one and the COMBMODE field specifies a match component to the triggering of this event, then a match is considered to be active whenever the counter value is GREATER THAN OR EQUAL TO the value specified in the match register when counting up, LESS THEN OR EQUAL TO the match value when counting down. If this bit is zero, a match is only be active during the cycle when the counter is equal to the match value. | 0 |
| 22:21 | DIRECTION | | Direction qualifier for event generation. This field only applies when the counters are operating in BIDIR mode. If BIDIR = 0, the SCT ignores this field. Value 0x3 is reserved. | 0 |
| | | 0x0 | Direction independent. This event is triggered regardless of the count direction. | |
| | | 0x1 | Counting up. This event is triggered only during up-counting when BIDIR = 1. | |
| | | 0x2 | Counting down. This event is triggered only during down-counting when BIDIR = 1. | |
| 31:23 | - | | Reserved | - |

14.6.26 SCT output set registers 0 to 7

Based on a selected event, each SCT output can be set.

There is one output set register for each SCT output which selects which events can set that output. Each bit of an output set register is associated with a different event (bit 0 with event 0, etc.). A selected event can set or clear the output depending on the setting of the SETCLRn field in the OUTPUTDIRCTRL register. To define the actual event that sets the output (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Table 232. SCT output set register (OUT[0:7]_SET, address 0x5000 4500 (OUT0_SET) to 0x5000 4538 (OUT7_SET) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 15:0 | SET | A 1 in bit m selects event m to set output n (or clear it if SETCLRn = 0x1 or 0x2) event 0 = bit 0, event 1 = bit 1, ...up to the number of bits = number of events in this SCT. When the counter is used in bi-directional mode, it is possible to reverse the action specified by the output set and clear registers when counting down, See the OUTPUTCTRL register. | 0 |
| 31:16 | - | Reserved | - |

14.6.27 SCT output clear registers 0 to 7

Based on a selected event, each SCT output can be cleared.

There is one register for each SCT output which selects which events can clear that output. Each bit of an output clear register is associated with a different event (bit 0 with event 0, etc.). A selected event can clear or set the output depending on the setting of the SETCLRn field in the OUTPUTDIRCTRL register. To define the actual event that clears the output (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

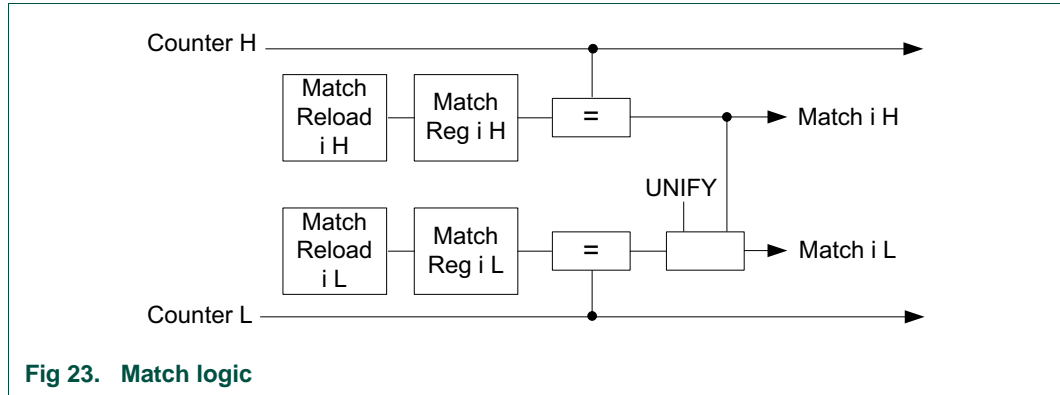
Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Table 233. SCT output clear register (OUT[0:7]_CLR, address 0x5000 4504 (OUT0_CLR) to 0x5000 453C (OUT7_CLR)) bit description

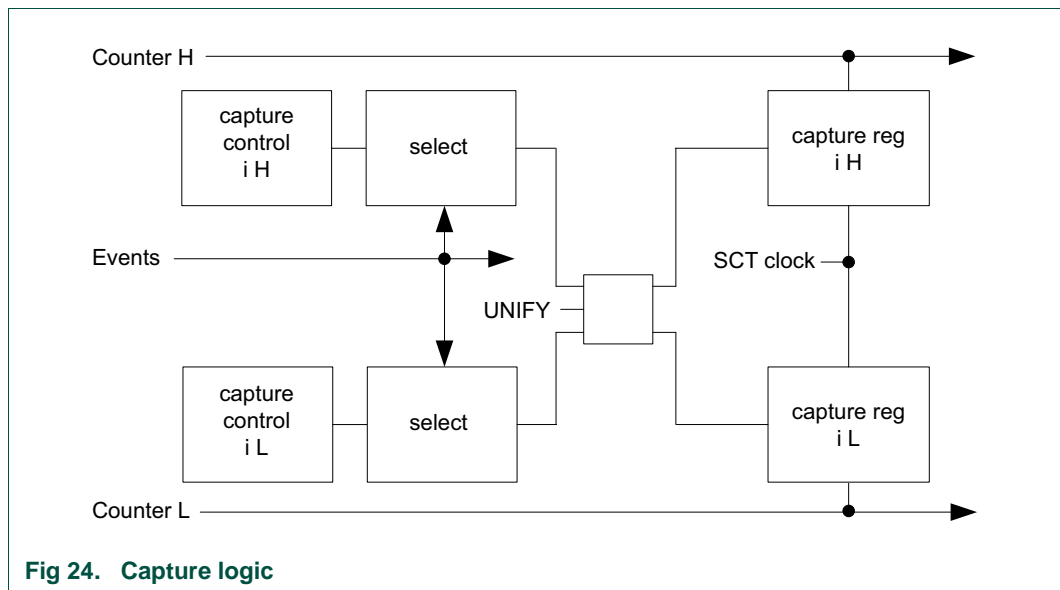
| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 15:0 | CLR | A 1 in bit m selects event m to clear output n (or set it if SETCLRn = 0x1 or 0x2) event 0 = bit 0, event 1 = bit 1, ... up to the number of bits = number of events in this SCT. When the counter is used in bi-directional mode, it is possible to reverse the action specified by the output set and clear registers when counting down, See the OUTPUTCTRL register. | 0 |
| 31:16 | - | Reserved | - |

14.7 Functional description

14.7.1 Match logic



14.7.2 Capture logic



14.7.3 Event selection

State variables allow control of the SCT across more than one cycle of the counter. Counter matches, input/output edges, and state values are combined into a set of general-purpose events that can switch outputs, request interrupts, and change state values.

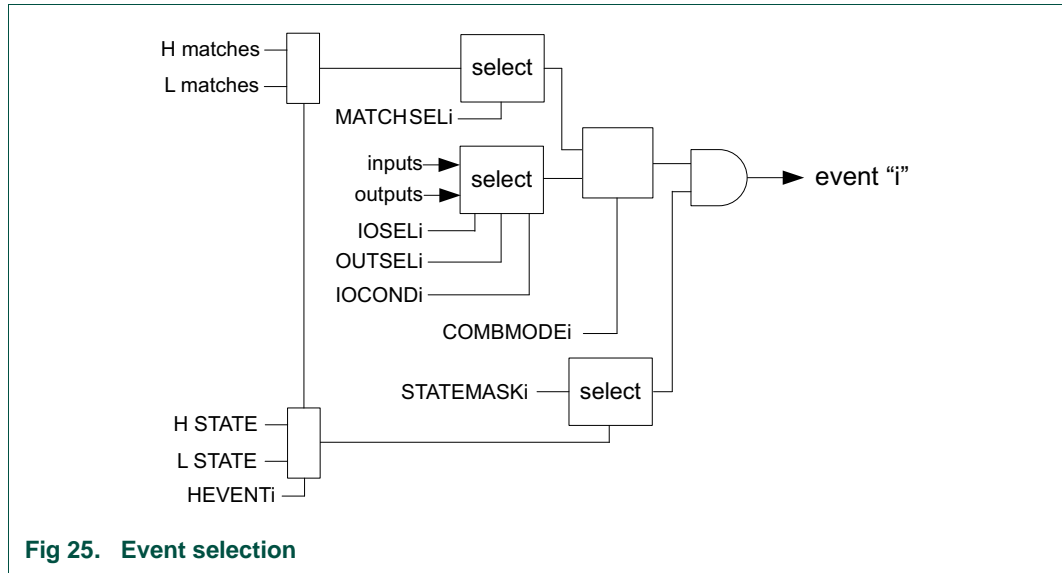


Fig 25. Event selection

14.7.4 Output generation

Figure 26 shows one output slice of the SCT.

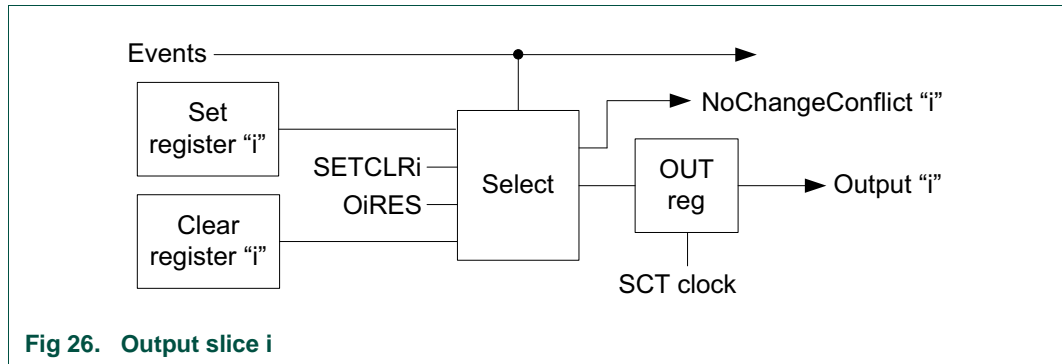


Fig 26. Output slice i

14.7.5 State logic

The SCT can be configured as a timer/counter with multiple programmable states. The states are user-defined through the events that can be captured in each particular state. In a multi-state SCT, the SCT can change from one state to another state when a user-defined event triggers a state change. The state change is triggered through each event's EV_CTRL register in one of the following ways:

- The event can increment the current state number by a new value.
- The event can write a new state value.

If an event increments the state number beyond the number of available states, the SCT enters a locked state in which all further events are ignored while the counter is still running. Software must interfere to change out of this state.

Software can capture the counter value (and potentially create an interrupt and write to all outputs) when the event moving the SCT into a locked state occurs. Later, while the SCT is in the locked state, software can read the counter again to record the time passed since the locking event and can also read the state variable to obtain the current state number.

If the SCT registers an event that forces an abort, putting the SCT in a locked state can be a safe way to record the time that has passed since the abort event while no new events are allowed to occur. Since multiple states (any state number between the maximum implemented state and 31) are locked states, multiple abort or error events can be defined each incrementing the state number by a different value.

14.7.6 Interrupt generation

The SCT generates one interrupt to the NVIC.

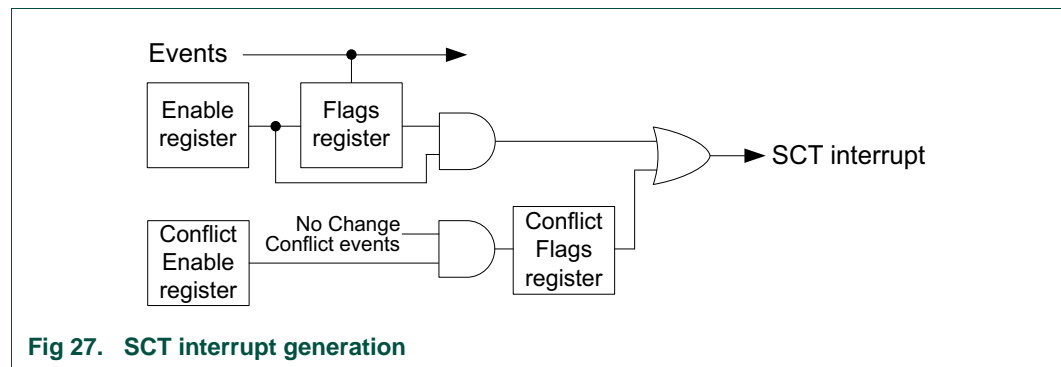


Fig 27. SCT interrupt generation

14.7.7 Clearing the prescaler

When enabled by a non-zero PRE field in the Control register, the prescaler acts as a clock divider for the counter, like a fractional part of the counter value. The prescaler is cleared whenever the counter is cleared or loaded for any of the following reasons:

- Hardware reset
- Software writing to the counter register
- Software writing a 1 to the CLRCTR bit in the control register
- an event selected by a 1 in the counter limit register when BIDIR = 0

When BIDIR is 0, a limit event caused by an I/O signal can clear a non-zero prescaler. However, a limit event caused by a Match only clears a non-zero prescaler in one special case as described [Section 14.7.8](#).

A limit event when BIDIR is 1 does not clear the prescaler. Rather it clears the DOWN bit in the Control register, and decrements the counter on the same clock if the counter is enabled in that clock.

14.7.8 Match vs. I/O events

Counter operation is complicated by the prescaler and by clock mode 01 in which the SCT clock is the bus clock. However, the prescaler and counter are enabled to count only when a selected edge is detected on a clock input.

- The prescaler is enabled when the clock mode is not 01, or when the input edge selected by the CLKSEL field is detected.
- The counter is enabled when the prescaler is enabled, and (PRELIM=0 or the prescaler is equal to the value in PRELIM).

An I/O component of an event can occur in any SCT clock when its counter HALT bit is 0. In general, a Match component of an event can only occur in a UT clock when its counter HALT and STOP bits are both 0 and the counter is enabled.

[Table 234](#) shows when the various kinds of events can occur.

Table 234. Event conditions

| COMBMODE | IOMODE | Event can occur on clock: |
|----------|--------------|---|
| IO | Any | Event can occur whenever HALT = 0 (type A). |
| MATCH | Any | Event can occur when HALT = 0 and STOP = 0 and the counter is enabled (type C). |
| OR | Any | From the IO component: Event can occur whenever HALT = 0 (A). From the match component: Event can occur when HALT = 0 and STOP = 0 and the counter is enabled (C). |
| AND | LOW or HIGH | Event can occur when HALT = 0 and STOP = 0 and the counter is enabled (C). |
| AND | RISE or FALL | Event can occur whenever HALT = 0 (A). |

14.7.9 SCT operation

In its simplest, single-state configuration, the SCT operates as an event controlled one- or bidirectional counter. Events can be configured to be counter match events, an input or output level, transitions on an input or output pin, or a combination of match and input/output behavior. In response to an event, the SCT output or outputs can transition, or the SCT can perform other actions such as creating an interrupt or starting, stopping, or resetting the counter. Multiple simultaneous actions are allowed for each event. Furthermore, any number of events can trigger one specific action of the SCT.

An action or multiple actions of the SCT uniquely define an event. A state is defined by which events are enabled to trigger an SCT action or actions in any stage of the counter. Events not selected for this state are ignored.

In a multi-state configuration, states change in response to events. A state change is an additional action that the SCT can perform when the event occurs. When an event is configured to change the state, the new state defines a new set of events resulting in different actions of the SCT. Through multiple cycles of the counter, events can change the state multiple times and thus create a large variety of event controlled transitions on the SCT outputs and/or interrupts.

Once configured, the SCT can run continuously without software intervention and can generate multiple output patterns entirely under the control of events.

- To configure the SCT, see [Section 14.7.10](#).
- To start, run, and stop the SCT, see [Section 14.7.11](#).
- To configure the SCT as simple event controlled counter/timer, see [Section 14.7.12](#).

14.7.10 Configure the SCT

To set up the SCT for multiple events and states, perform the following configuration steps:

14.7.10.1 Configure the counter

1. Configure the L and H counters in the CONFIG register by selecting two independent 16-bit counters (L counter and H counter) or one combined 32-bit counter in the UNIFY field.
2. Select the SCT clock source in the CONFIG register (fields CLKMODE and CLKSEL) from any of the inputs or an internal clock.

14.7.10.2 Configure the match and capture registers

1. Select how many match and capture registers the application uses (not more than what is available on this device):
 - In the REGMODE register, select for each of the match/capture register pairs whether the register is used as a match register or capture register.
2. Define match conditions for each match register selected:
 - Each match register MATCH sets one match value, if a 32-bit counter is used, or two match values, if the L and H 16-bit counters are used.
 - Each match reload register MATCHRELOAD sets a reload value that is loaded into the match register when the counter reaches a limit condition or the value 0.

14.7.10.3 Configure events and event responses

1. Define when each event can occur in the following way in the EVn_CTRL registers (up to 6, one register per event):
 - Select whether the event occurs on an input or output changing, on an input or output level, a match condition of the counter, or a combination of match and input/output conditions in field COMBMODE.
 - For a match condition:

Select the match register that contains the match condition for the event to occur. Enter the number of the selected match register in field MATCHSEL.

If using L and H counters, define whether the event occurs on matching the L or the H counter in field HEVENT.
 - For an SCT input or output level or transition:

Select the input number or the output number that is associated with this event in fields IOSEL and OUTSEL.

Define how the selected input or output triggers the event (edge or level sensitive) in field IOCOND.
2. Define what the effect of each event is on the SCT outputs in the OUTn_SET or OUTn_CLR registers (up to the maximum number of outputs on this device, one register per output):
 - For each SCT output, select which events set or clear this output. More than one event can change the output, and each event can change multiple outputs.
3. Define how each event affects the counter:

- Set the corresponding event bit in the LIMIT register for the event to set an upper limit for the counter.
When a limit event occurs in unidirectional mode, the counter is cleared to zero and begins counting up on the next clock edge.
When a limit event occurs in bidirectional mode, the counter begins to count down from the current value on the next clock edge.
 - Set the corresponding event bit in the HALT register for the event to halt the counter. If the counter is halted, it stops counting and no new events can occur. The counter operation can only be restored by clearing the HALT_L and/or the HALT_H bits in the CTRL register.
 - Set the corresponding event bit in the STOP register for the event to stop the counter. If the counter is stopped, it stops counting. However, an event that is configured as a transition on an input/output can restart the counter.
 - Set the corresponding event bit in the START register for the event to restart the counting. Only events that are defined by an input changing can be used to restart the counter.
4. Define which events contribute to the SCT interrupt:
- Set the corresponding event bit in the EVEN and the EVFLAG registers to enable the event to contribute to the SCT interrupt.

14.7.10.4 Configure multiple states

1. In the EVn_STATE register for each event (up to the maximum number of events on this device, one register per event), select the state or states (up to 2) in which this event is allowed to occur. Each state can be selected for more than one event.
2. Determine how the event affects the system state:
In the EVn_CTRL registers (up to the maximum number of events on this device, one register per event), set the new state value in the STATEV field for this event. If the event is the highest numbered in the current state, this value is either added to the existing state value or replaces the existing state value, depending on the field STATELD.

Remark: If there are higher numbered events in the current state, this event cannot change the state.

If the STATEV and STATELD values are set to zero, the state does not change.

14.7.10.5 Miscellaneous options

- There are a certain (selectable) number of capture registers. Each capture register can be programmed to capture the counter contents when one or more events occur.
- If the counter is in bidirectional mode, the effect of set and clear of an output can be made to depend on whether the counter is counting up or down by writing to the OUTPUTDIRCTRL register.

14.7.11 Run the SCT

1. Configure the SCT (see [Section 14.7.10 “Configure the SCT”](#)).
2. Write to the STATE register to define the initial state. By default the initial state is state 0.

3. To start the SCT, write to the CTRL register:
 - Clear the counters.
 - Clear or set the STOP_L and/or STOP_H bits.
Remark: The counter starts counting once the STOP bit is cleared as well. If the STOP bit is set, the SCT waits instead for an event to occur that is configured to start the counter.
 - For each counter, select unidirectional or bidirectional counting mode (field BIDIR_L and/or BIDIR_H).
 - Select the prescale factor for the counter clock (CTRL register).
 - Clear the HALT_L and/or HALT_H bit. By default, the counters are halted and no events can occur.
4. To stop the counters by software at any time, stop or halt the counter (write to STOP_L and/or STOP_H bits or HALT_L and/or HALT_H bits in the CTRL register).
 - When the counters are stopped, both an event configured to clear the STOP bit or software writing a zero to the STOP bit can start the counter again.
 - When the counter are halted, only a software write to clear the HALT bit can start the counter again. No events can occur.
 - When the counters are halted, software can set any SCT output HIGH or LOW directly by writing to the OUT register.

The current state can be read at any time by reading the STATE register.

To change the current state by software (that is independently of any event occurring), set the HALT bit and write to the STATE register to change the state value. Writing to the STATE register is only allowed when the counter is halted (the HALT_L and/or HALT_H bits are set) and no events can occur.

14.7.12 Configure the SCT without using states

The SCT can be used as standard counter/timer with external capture inputs and match outputs without using the state logic. To operate the SCT without states, configure the SCT as follows:

- Write zero to the STATE register (zero is the default).
- Write zero to the STATELD and STATEV fields in the EVCTRL registers for each event.
- Write 0x1 to the EVn_STATE register of each event. Writing 0x1 enables the event.
In effect, the event is allowed to occur in a single state which never changes while the counter is running.

14.7.13 SCT PWM Example

[Figure 28](#) shows a simple application of the SCT using two sets of match events (EV0/1 and EV3/4) to set/clear SCT output 0. The timer is automatically reset whenever it reaches the MAT0 match value.

In the initial state 0, match event EV0 sets output 0 to HIGH and match event EV1 clears output 0. The SCT input 0 is monitored: If input0 is found LOW by the next time the timer is reset(EV2), the state is changed to state 1, and EV3/4 are enabled, which create the

same output but triggered by different match values. If input 0 is found HIGH by the next time the timer is reset, the associated event (EV5) causes the state to change back to state 0 where the events EV0 and EV1 are enabled.

The example uses the following SCT configuration:

- 1 input
- 1 output
- 5 match registers
- 6 events and match 0 used with autolimit function
- 2 states

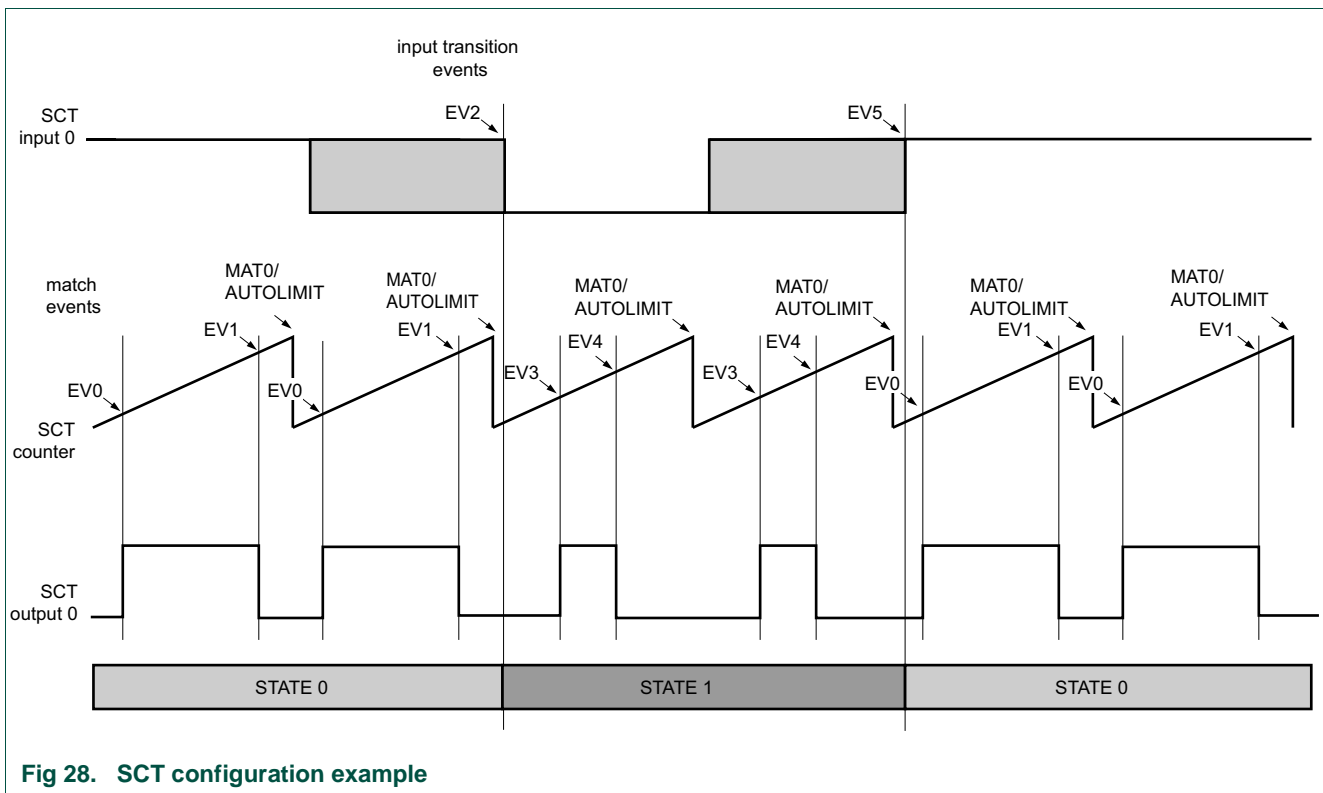


Fig 28. SCT configuration example

This application of the SCT uses the following configuration (all register values not listed in [Table 235](#) are set to their default values):

Table 235. SCT configuration example

| Configuration | Registers | Setting |
|-------------------------|-----------|---|
| Counter | CONFIG | Uses one counter (UNIFY = 1). |
| | CONFIG | Enable the autolimit for MAT0. (AUTOLIMIT = 1.) |
| | CTRL | Uses unidirectional counter (BIDIR_L = 0). |
| Clock base | CONFIG | Uses default values for clock configuration. |
| Match/Capture registers | REGMODE | Configure one match register for each match event by setting REGMODE_L bits 0,1, 2, 3, 4 to 0. This is the default. |

Table 235. SCT configuration example

| Configuration | Registers | Setting |
|--|-------------------|---|
| Define match values | MATCH0/1/2/3/4 | Set a match value MATCH0/1/2/4/5_L in each register. The match 0 register serves as an automatic limit event that resets the counter. without using an event. To enable the automatic limit, set the AUTOLIMIT bit in the CONFIG register. |
| Define match reload values | MATCHREL0/1/2/3/4 | Set a match reload value RELOAD0/1/2/3/4_L in each register (same as the match value in this example). |
| Define when event 0 occurs | EV0_CTRL | <ul style="list-style-type: none"> Set COMBMODE = 0x1. Event 0 uses match condition only. Set MATCHSEL = 1. Select match value of match register 1. The match value of MAT1 is associated with event 0. |
| Define when event 1 occurs | EV1_CTRL | <ul style="list-style-type: none"> Set COMBMODE = 0x1. Event 1 uses match condition only. Set MATCHSEL = 2 Select match value of match register 2. The match value of MAT2 is associated with event 1. |
| Define when event 2 occurs | EV2_CTRL | <ul style="list-style-type: none"> Set COMBMODE = 0x3. Event 2 uses match condition and I/O condition. Set IOSEL = 0. Select input 0. Set IOCOND = 0x0. Input 0 is LOW. Set MATCHSEL = 0. Chooses match register 0 to qualify the event. |
| Define how event 2 changes the state | EV2_CTRL | Set STATEV bits to 1 and the STATED bit to 1. Event 2 changes the state to state 1. |
| Define when event 3 occurs | EV3_CTRL | <ul style="list-style-type: none"> Set COMBMODE = 0x1. Event 3 uses match condition only. Set MATCHSEL = 0x3. Select match value of match register 3. The match value of MAT3 is associated with event 3.. |
| Define when event 4 occurs | EV4_CTRL | <ul style="list-style-type: none"> Set COMBMODE = 0x1. Event 4 uses match condition only. Set MATCHSEL = 0x4. Select match value of match register 4.The match value of MAT4 is associated with event 4. |
| Define when event 5 occurs | EV5_CTRL | <ul style="list-style-type: none"> Set COMBMODE = 0x3. Event 5 uses match condition and I/O condition. Set IOSEL = 0. Select input 0. Set IOCOND = 0x3. Input 0 is HIGH. Set MATCHSEL = 0. Chooses match register 0 to qualify the event. |
| Define how event 5 changes the state | EV5_CTRL | Set STATEV bits to 0 and the STATED bit to 1. Event 5 changes the state to state 0. |
| Define by which events output 0 is set | OUT0_SET | Set SET0 bits 0 (for event 0) and 3 (for event 3) to one to set the output when these events 0 and 3 occur. |
| Define by which events output 0 is cleared | OUT0_CLR | Set CLR0 bits 1 (for events 1) and 4 (for event 4) to one to clear the output when events 1 and 4 occur. |
| Configure states in which event 0 is enabled | EV0_STATE | Set STATEMSK0 bit 0 to 1. Set all other bits to 0. Event 0 is enabled in state 0. |
| Configure states in which event 1 is enabled | EV1_STATE | Set STATEMSK1 bit 0 to 1. Set all other bits to 0. Event 1 is enabled in state 0. |
| Configure states in which event 2 is enabled | EV2_STATE | Set STATEMSK2 bit 0 to 1. Set all other bits to 0. Event 2 is enabled in state 0. |
| Configure states in which event 3 is enabled | EV3_STATE | Set STATEMSK3 bit 1 to 1. Set all other bits to 0. Event 3 is enabled in state 1. |
| Configure states in which event 4 is enabled | EV4_STATE | Set STATEMSK4 bit 1 to 1. Set all other bits to 0. Event 4 is enabled in state 1. |
| Configure states in which event 5 is enabled | EV5_STATE | Set STATEMSK5 bit 1 to 1. Set all other bits to 0. Event 5 is enabled in state 1. |

15.1 How to read this chapter

These five standard timers are available on all LPC5410x parts.

15.2 Basic configuration

- Set the appropriate bits to enable clocks to timers that will be used: CT32B0 and CT32B1 in the ASYNCAPBCLKCTRL register, CT32B2, CT32B3, and CT32B4 in the AHBCLKCTRL1 register.
- Clear the timer reset using the ASYNCPRESETCTRL register ([Table 89](#) for CT32B0 and 1) and the PRESETCTRL1 register ([Table 36](#) for CT32B2, 3, and 4).
- Pins: Select timer pins and pin modes as needed through the relevant IOCON registers ([Chapter 8](#)).
- Interrupts: See register MCR ([Table 250](#)) and CCR ([Table 254](#)) for match and capture events. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
- DMA: Some timer match conditions can be used to generate timed DMA requests, see [Table 172](#).

15.3 Features

- Each is a 32-bit counter/timer with a programmable 32-bit prescaler. Four of the timers include external capture and match pin connections.
- Counter or timer operation.
- For each timer with pin connections, up to 4 32-bit capture channels that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.
- The timer and prescaler may be configured to be cleared on a designated capture event. This feature permits easy pulse-width measurement by clearing the timer on the leading edge of an input pulse and capturing the timer value on the trailing edge.
- Four 32-bit match registers that allow:
 - Continuous operation with optional interrupt generation on match.
 - Stop timer on match with optional interrupt generation.
 - Reset timer on match with optional interrupt generation.
- For each timer with pin connections, up to 4 external outputs corresponding to match registers with the following capabilities:
 - Set LOW on match.
 - Set HIGH on match.
 - Toggle on match.
 - Do nothing on match.

- PWM: for each timer with pin connections, up to 3 match outputs can be used as single edge controlled PWM outputs.

15.4 Applications

- Interval Timer for counting internal events.
- PWM outputs
- Pulse Width Demodulator via Capture inputs.
- Free running timer.

15.5 General description

Each Counter/timer is designed to count cycles of the peripheral clock (PCLK) or an externally supplied clock and can optionally generate interrupts or perform other actions at specified timer values based on four match registers. Each counter/timer also includes one capture input to trap the timer value when an input signal transitions, optionally generating an interrupt.

In PWM mode, three match registers can be used to provide a single-edge controlled PWM output on the match output pins. One match register is used to control the PWM cycle length.

15.5.1 Capture inputs

The capture signal can be configured to load the Capture Register with the value in the counter/timer and optionally generate an interrupt. The capture signal is generated by one of the pins with a capture function. Each capture signal is connected to one capture channel of the timer.

The Counter/Timer block can select a capture signal as a clock source instead of the PCLK derived clock. For more details see [Section 15.7.11](#).

15.5.2 Match outputs

When a match register equals the timer counter (TC), the corresponding match output can either toggle, go LOW, go HIGH, or do nothing. The External Match Register (EMR) and the PWM Control Register (PWMCON) control the functionality of this output.

15.5.3 Applications

- Interval timer for counting internal events
- Pulse Width Modulator via match outputs
- Pulse Width Demodulator via capture input
- Free running timer

15.5.4 Architecture

The block diagram for the timers is shown in [Figure 29](#).

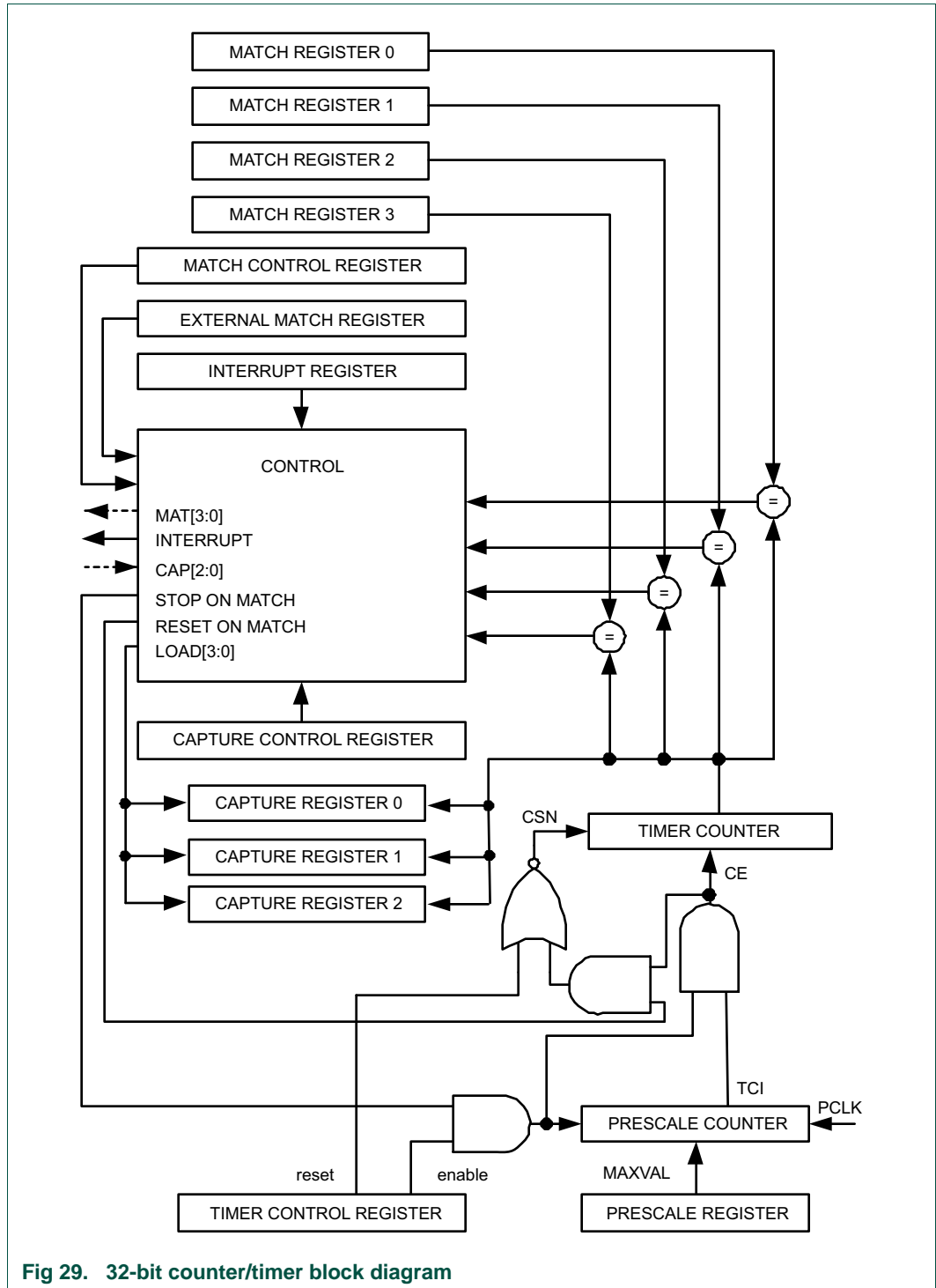


Fig 29. 32-bit counter/timer block diagram

15.6 Pin description

[Table 236](#) gives a brief summary of each of the Timer/Counter related pins. Recommended IOCON settings are shown in [Table 237](#).

Table 236. Timer/Counter pin description

| Pin | Type | Description |
|--|--------|---|
| CT32B0_CAP3:0 CT32B1_CAP3:0 CT32B2_CAP3:0 CT32B3_CAP3:0 | Input | Capture Signals- A transition on a capture pin can be configured to load one of the Capture Registers with the value in the Timer Counter and optionally generate an interrupt. Capture functionality can be selected from a number of pins. When more than one pin is selected for a Capture input on a single timer channel, the pin with the lowest Port number is used Timer/Counter block can select a capture signal as a clock source instead of the PCLK derived clock. For more details see Section 15.7.11 . |
| CT32B0_MAT1:0 CT32B1_MAT1:0 CT32B2_MAT3:0 CT32B3_MAT1:0 | Output | External Match Output - When a match register (MR3:0) equals the timer counter (TC) this output can either toggle, go low, go high, or do nothing. The External Match Register (EMR) controls the functionality of this output. Match Output functionality can be selected on a number of pins in parallel. |

Table 237: Suggested CT32B timer pin settings

| IOCON bit(s) | Type D pin | Type A pin | Type I pin |
|-----------------|---|--|--|
| 10 | OD: Set to 0 unless open-drain output is desired. | Same as type D. | I2CFILTER: Set to 1 |
| 9 | SLEW: Generally set to 0. | Not used, set to 0 | I2CDRIVE: Set to 0. |
| 8 | FILTEROFF: Generally set to 1. | Same as type D. | Same as type D. |
| 7 | DIGIMODE: Set to 1. | Same as type D. | Same as type D. |
| 6 | INVERT: Set to 0. | Same as type D. | Same as type D. |
| 5 | Not used, set to 0. | Same as type D. | I2CSLEW: Set to 1. |
| 4:3 | MODE: Set to 00 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input). | Same as type D. | Not used, set to 0. |
| 2:0 | FUNC: Must select the correct function for this peripheral. | Same as type D. | Same as type D. |
| General comment | A good choice for timer capture input or match output. | A reasonable choice for timer capture input or match output. | Not recommended for timer match outputs. |

15.6.1 Multiple CAP and MAT pins

Software can select from multiple pins for the CAP or MAT functions in the IOCON registers, which are described in [Chapter 8](#). Note that match conditions may be used internally without the use of a device pin.

15.7 Register description

Each Timer/Counter contains the registers shown in [Table 238](#) ("Reset Value" refers to the data stored in used bits only; it does not include reserved bits content). More detailed descriptions follow.

Table 238. Register overview: CT32B0/1/2/3 (register base addresses 0x400B 4000 (CT32B0), 0x400B 8000 (CT32B1), 0x4000 4000 (CT32B2), 0x4000 8000 (CT32B3), 0x4000 C000 (CT32B4))

| Name | Access | Address offset | Description | Reset value ^[1] | Reference |
|------|--------|----------------|---|----------------------------|---------------------------|
| IR | R/W | 0x00 | Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending. | 0 | Table 240 |
| TCR | R/W | 0x04 | Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR. | 0 | Table 242 |
| TC | R/W | 0x08 | Timer Counter. The 32 bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR. | 0 | Table 244 |
| PR | R/W | 0x0C | Prescale Register. When the Prescale Counter (PC) is equal to this value, the next clock increments the TC and clears the PC. | 0 | Table 246 |
| PC | R/W | 0x10 | Prescale Counter. The 32 bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface. | 0 | Table 248 |
| MCR | R/W | 0x14 | Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs. | 0 | Table 250 |
| MR0 | R/W | 0x18 | Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC. | 0 | Table 252 |
| MR1 | R/W | 0x1C | Match Register 1. See MR0 description. | 0 | Table 252 |
| MR2 | R/W | 0x20 | Match Register 2. See MR0 description. | 0 | Table 252 |
| MR3 | R/W | 0x24 | Match Register 3. See MR0 description. | 0 | Table 252 |
| CCR | R/W | 0x28 | Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place. | 0 | Table 254 |
| CR0 | RO | 0x2C | Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CAPn.0 input. | 0 | Table 256 |
| CR1 | RO | 0x30 | Capture Register 1. See CR0 description. | 0 | Table 256 |
| CR2 | RO | 0x34 | Capture Register 2. See CR0 description. | 0 | Table 256 |
| CR3 | RO | 0x38 | Capture Register 3. See CR0 description. | 0 | Table 256 |
| EMR | R/W | 0x3C | External Match Register. The EMR controls the match function and the external match pins. | 0 | Table 258 |
| CTCR | R/W | 0x70 | Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting. | 0 | Table 260 |
| PWMC | R/W | 0x74 | PWM Control Register. The PWMCON enables PWM mode for the external match pins. | 0 | Table 262 |

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

15.7.1 Interrupt Register

The Interrupt Register consists of 4 bits for the match interrupts and 4 bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be high. Otherwise, the bit will be low. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect. The act of clearing an interrupt for a timer match also clears any corresponding DMA request. Writing a zero has no effect.

Table 239. Address map IR register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| CT32B0 | 0x400B 4000 | 0x000 | - | 1 |
| CT32B1 | 0x400B 8000 | 0x000 | - | 1 |
| CT32B2 | 0x4000 4000 | 0x000 | - | 1 |
| CT32B3 | 0x4000 8000 | 0x000 | - | 1 |
| CT32B4 | 0x4000 C000 | 0x000 | - | 1 |

Table 240. Interrupt Register (IR, address offset 0x000) bit description

| Bit | Symbol | Description | Reset Value |
|------|--------|---|-------------|
| 0 | MR0INT | Interrupt flag for match channel 0. | 0 |
| 1 | MR1INT | Interrupt flag for match channel 1. | 0 |
| 2 | MR2INT | Interrupt flag for match channel 2. | 0 |
| 3 | MR3INT | Interrupt flag for match channel 3. | 0 |
| 4 | CR0INT | Interrupt flag for capture channel 0 event. | 0 |
| 5 | CR1INT | Interrupt flag for capture channel 1 event. | 0 |
| 6 | CR2INT | Interrupt flag for capture channel 2 event. | 0 |
| 7 | CR3INT | Interrupt flag for capture channel 3 event. | 0 |
| 31:6 | - | Reserved. Read value is undefined, only zero should be written. | - |

15.7.2 Timer Control Register

The Timer Control Register (TCR) is used to control the operation of the Timer/Counter.

Table 241. Address map TCR register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| CT32B0 | 0x400B 4000 | 0x004 | - | 1 |
| CT32B1 | 0x400B 8000 | 0x004 | - | 1 |
| CT32B2 | 0x4000 4000 | 0x004 | - | 1 |
| CT32B3 | 0x4000 8000 | 0x004 | - | 1 |
| CT32B4 | 0x4000 C000 | 0x004 | - | 1 |

Table 242. Timer Control Register (TCR, address offset 0x004) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|--------|-------|--|-------------|
| 0 | CEN | | Counter enable. | 0 |
| | | 0 | Disabled. The counters are disabled. | |
| | | 1 | Enabled. The Timer Counter and Prescale Counter are enabled. | |

Table 242. Timer Control Register (TCR, address offset 0x004) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|--|-------------|
| 1 | CRST | | Counter reset. | 0 |
| | | 0 | Disabled. Do nothing. | |
| | | 1 | Enabled. The Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until TCR[1] is returned to zero. | |
| 31:2 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

15.7.3 Timer Counter registers

The 32-bit Timer Counter register is incremented when the prescale counter reaches its terminal count. Unless it is reset before reaching its upper limit, the Timer Counter will count up through the value 0xFFFF FFFF and then wrap back to the value 0x0000 0000. This event does not cause an interrupt, but a match register can be used to detect an overflow if needed.

Table 243. Address map TC register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| CT32B0 | 0x400B 4000 | 0x008 | - | 1 |
| CT32B1 | 0x400B 8000 | 0x008 | - | 1 |
| CT32B2 | 0x4000 4000 | 0x008 | - | 1 |
| CT32B3 | 0x4000 8000 | 0x008 | - | 1 |
| CT32B4 | 0x4000 C000 | 0x004 | - | 1 |

Table 244. Timer counter registers (TC, address offset 0x08) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|----------------------|-------------|
| 31:0 | TCVAL | Timer counter value. | 0 |

15.7.4 Prescale register

The 32-bit Prescale register specifies the maximum value for the Prescale Counter.

Table 245. Address map PR register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| CT32B0 | 0x400B 4000 | 0x00C | - | 1 |
| CT32B1 | 0x400B 8000 | 0x00C | - | 1 |
| CT32B2 | 0x4000 4000 | 0x00C | - | 1 |
| CT32B3 | 0x4000 8000 | 0x00C | - | 1 |
| CT32B4 | 0x4000 C000 | 0x00C | - | 1 |

Table 246. Timer prescale registers (PR, address offset 0x00C) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|-------------------------|-------------|
| 31:0 | PRVAL | Prescale counter value. | 0 |

15.7.5 Prescale Counter register

The 32-bit Prescale Counter controls division of PCLK by some constant value before it is applied to the Timer Counter. This allows control of the relationship of the resolution of the timer versus the maximum time before the timer overflows. The Prescale Counter is incremented on every PCLK. When it reaches the value stored in the Prescale register, the Timer Counter is incremented and the Prescale Counter is reset on the next PCLK. This causes the Timer Counter to increment on every PCLK when PR = 0, every 2 pclks when PR = 1, etc.

Table 247. Address map PC register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| CT32B0 | 0x400B 4000 | 0x010 | - | 1 |
| CT32B1 | 0x400B 8000 | 0x010 | - | 1 |
| CT32B2 | 0x4000 4000 | 0x010 | - | 1 |
| CT32B3 | 0x4000 8000 | 0x010 | - | 1 |
| CT32B4 | 0x4000 C000 | 0x010 | - | 1 |

Table 248. Timer prescale counter registers (PC, address offset 0x010) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|-------------------------|-------------|
| 31:0 | PCVAL | Prescale counter value. | 0 |

15.7.6 Match Control Register

The Match Control Register is used to control what operations are performed when one of the Match Registers matches the Timer Counter.

Table 249. Address map MCR register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| CT32B0 | 0x400B 4000 | 0x014 | - | 1 |
| CT32B1 | 0x400B 8000 | 0x014 | - | 1 |

Table 249. Address map MCR register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| CT32B2 | 0x4000 4000 | 0x014 | - | 1 |
| CT32B3 | 0x4000 8000 | 0x014 | - | 1 |
| CT32B4 | 0x4000 C000 | 0x014 | - | 1 |

Table 250. Match Control Register (MCR, address offset 0x014) bit description

| Bit | Symbol | Description | Reset Value |
|-------|--------|--|-------------|
| 0 | MR0I | Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC. 0 = disabled. 1 = enabled. | 0 |
| 1 | MR0R | Reset on MR0: the TC will be reset if MR0 matches it. 0 = disabled. 1 = enabled. | 0 |
| 2 | MR0S | Stop on MR0: the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC. 0 = disabled. 1 = enabled. | 0 |
| 3 | MR1I | Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC. 0 = disabled. 1 = enabled. | 0 |
| 4 | MR1R | Reset on MR1: the TC will be reset if MR1 matches it. 0 = disabled. 1 = enabled. | 0 |
| 5 | MR1S | Stop on MR1: the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC. 0 = disabled. 1 = enabled. | 0 |
| 6 | MR2I | Interrupt on MR2: an interrupt is generated when MR2 matches the value in the TC. 0 = disabled. 1 = enabled. | 0 |
| 7 | MR2R | Reset on MR2: the TC will be reset if MR2 matches it. 0 = disabled. 1 = enabled. | 0 |
| 8 | MR2S | Stop on MR2: the TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC. 0 = disabled. 1 = enabled. | 0 |
| 9 | MR3I | Interrupt on MR3: an interrupt is generated when MR3 matches the value in the TC. 0 = disabled. 1 = enabled. | 0 |
| 10 | MR3R | Reset on MR3: the TC will be reset if MR3 matches it. 0 = disabled. 1 = enabled. | 0 |
| 11 | MR3S | Stop on MR3: the TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC. 0 = disabled. 1 = enabled. | 0 |
| 31:12 | - | Reserved. Read value is undefined, only zero should be written. | NA |

15.7.7 Match Registers

The Match register values are continuously compared to the Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

Table 251. Address map MR[0:3] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| CT32B0 | 0x400B 4000 | [0x018:0x024] | 0x4 | 4 |
| CT32B1 | 0x400B 8000 | [0x018:0x024] | 0x4 | 4 |
| CT32B2 | 0x4000 4000 | [0x018:0x024] | 0x4 | 4 |
| CT32B3 | 0x4000 8000 | [0x018:0x024] | 0x4 | 4 |
| CT32B4 | 0x4000 C000 | [0x018:0x024] | 0x4 | 4 |

Table 252. Timer match registers (MR[0:3], address offset [0x018:0x024]) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|----------------------------|-------------|
| 31:0 | MATCH | Timer counter match value. | 0 |

15.7.8 Capture Control Register

The Capture Control Register is used to control whether one of the four Capture Registers is loaded with the value in the Timer Counter when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges. In the description below, "n" represents the timer number, 0 or 1.

Note: If Counter mode is selected for a particular CAP input in the CTCR, the 3 bits for that input in this register should be programmed as 000, but capture and/or interrupt can be selected for the other 3 CAP inputs.

Table 253. Address map CCR register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| CT32B0 | 0x400B 4000 | 0x028 | - | 1 |
| CT32B1 | 0x400B 8000 | 0x028 | - | 1 |
| CT32B2 | 0x4000 4000 | 0x028 | - | 1 |
| CT32B3 | 0x4000 8000 | 0x028 | - | 1 |
| CT32B4 | 0x4000 C000 | 0x028 | - | 1 |

Table 254. Capture Control Register (CCR, address offset 0x028) bit description

| Bit | Symbol | Description | Reset Value |
|-------|--------|---|-------------|
| 0 | CAP0RE | Rising edge of capture channel 0: a sequence of 0 then 1 causes CR0 to be loaded with the contents of TC. 0 = disabled. 1 = enabled. | 0 |
| 1 | CAP0FE | Falling edge of capture channel 0: a sequence of 1 then 0 causes CR0 to be loaded with the contents of TC. 0 = disabled. 1 = enabled. | 0 |
| 2 | CAP0I | Generate interrupt on channel 0 capture event: a CR0 load generates an interrupt. | 0 |
| 3 | CAP1RE | Rising edge of capture channel 1: a sequence of 0 then 1 causes CR1 to be loaded with the contents of TC. 0 = disabled. 1 = enabled. | 0 |
| 4 | CAP1FE | Falling edge of capture channel 1: a sequence of 1 then 0 causes CR1 to be loaded with the contents of TC. 0 = disabled. 1 = enabled. | 0 |
| 5 | CAP1I | Generate interrupt on channel 1 capture event: a CR1 load generates an interrupt. | 0 |
| 6 | CAP2RE | Rising edge of capture channel 2: a sequence of 0 then 1 causes CR2 to be loaded with the contents of TC. 0 = disabled. 1 = enabled. | 0 |
| 7 | CAP2FE | Falling edge of capture channel 2: a sequence of 1 then 0 causes CR2 to be loaded with the contents of TC. 0 = disabled. 1 = enabled. | 0 |
| 8 | CAP2I | Generate interrupt on channel 2 capture event: a CR2 load generates an interrupt. | 0 |
| 9 | CAP3RE | Rising edge of capture channel 3: a sequence of 0 then 1 causes CR3 to be loaded with the contents of TC. 0 = disabled. 1 = enabled. | 0 |
| 10 | CAP3FE | Falling edge of capture channel 3: a sequence of 1 then 0 causes CR3 to be loaded with the contents of TC. 0 = disabled. 1 = enabled. | 0 |
| 11 | CAP3I | Generate interrupt on channel 3 capture event: a CR3 load generates an interrupt. | 0 |
| 31:12 | - | Reserved. Read value is undefined, only zero should be written. | NA |

15.7.9 Capture Registers

Each Capture register is associated with one capture channel and may be loaded with the counter/timer value when a specified event occurs on the signal defined for that capture channel. The signal could originate from an external pin or from an internal source. The settings in the Capture Control Register register determine whether the capture function is enabled, and whether a capture event happens on the rising edge of the associated signal, the falling edge, or on both edges.

Table 255. Address map CR[0:3] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| CT32B0 | 0x400B 4000 | [0x02C:0x038] | 0x4 | 4 |
| CT32B1 | 0x400B 8000 | [0x02C:0x038] | 0x4 | 4 |
| CT32B2 | 0x4000 4000 | [0x02C:0x038] | 0x4 | 4 |
| CT32B3 | 0x4000 8000 | [0x02C:0x038] | 0x4 | 4 |
| CT32B4 | 0x4000 C000 | [0x02C:0x038] | 0x4 | 4 |

Table 256. Timer capture registers (CR[0:3], address offsets [0x02C:0x038]) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|------------------------------|-------------|
| 31:0 | CAP | Timer counter capture value. | 0 |

15.7.10 External Match Register

The External Match Register provides both control and status of the external match pins. In the descriptions below, “n” represents the timer number, 0 or 1, and “m” represent a Match number, 0 through 3.

Match events for Match 0 and Match 1 in each timer can cause a DMA request, see [Section 15.8.2](#).

If the match outputs are configured as PWM output, the function of the external match registers is determined by the PWM rules ([Section 15.8.1 “Rules for single edge controlled PWM outputs” on page 223](#)).

Table 257. Address map EMR register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| CT32B0 | 0x400B 4000 | 0x03C | - | 1 |
| CT32B1 | 0x400B 8000 | 0x03C | - | 1 |
| CT32B2 | 0x4000 4000 | 0x03C | - | 1 |
| CT32B3 | 0x4000 8000 | 0x03C | - | 1 |
| CT32B4 | 0x4000 C000 | 0x03C | - | 1 |

Table 258. Timer external match registers (EMR, address offset 0x03C) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|--|-------------|
| 0 | EM0 | | External Match 0. This bit reflects the state of output MAT0, whether or not this output is connected to a pin. When a match occurs between the TC and MR0, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[5:4]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH. | 0 |
| 1 | EM1 | | External Match 1. This bit reflects the state of output MAT1, whether or not this output is connected to a pin. When a match occurs between the TC and MR1, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[7:6]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH. | 0 |
| 2 | EM2 | | External Match 2. This bit reflects the state of output MAT2, whether or not this output is connected to a pin. When a match occurs between the TC and MR2, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by EMR[9:8]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH. | 0 |
| 3 | EM3 | | External Match 3. This bit reflects the state of output MAT3, whether or not this output is connected to a pin. When a match occurs between the TC and MR3, this bit can either toggle, go LOW, go HIGH, or do nothing, as selected by MR[11:10]. This bit is driven to the MAT pins if the match function is selected via IOCON. 0 = LOW. 1 = HIGH. | 0 |
| 5:4 | EMC0 | | External Match Control 0. Determines the functionality of External Match 0. | 00 |
| | | 0x0 | Do Nothing. | |
| | | 0x1 | Clear. Clear the corresponding External Match bit/output to 0 (MAT0 pin is LOW if pinned out). | |
| | | 0x2 | Set. Set the corresponding External Match bit/output to 1 (MAT0 pin is HIGH if pinned out). | |
| | | 0x3 | Toggle. Toggle the corresponding External Match bit/output. | |
| 7:6 | EMC1 | | External Match Control 1. Determines the functionality of External Match 1. | 00 |
| | | 0x0 | Do Nothing. | |
| | | 0x1 | Clear. Clear the corresponding External Match bit/output to 0 (MAT1 pin is LOW if pinned out). | |
| | | 0x2 | Set. Set the corresponding External Match bit/output to 1 (MAT1 pin is HIGH if pinned out). | |
| | | 0x3 | Toggle. Toggle the corresponding External Match bit/output. | |
| 9:8 | EMC2 | | External Match Control 2. Determines the functionality of External Match 2. | 00 |
| | | 0x0 | Do Nothing. | |
| | | 0x1 | Clear. Clear the corresponding External Match bit/output to 0 (MAT2 pin is LOW if pinned out). | |
| | | 0x2 | Set. Set the corresponding External Match bit/output to 1 (MAT2 pin is HIGH if pinned out). | |
| | | 0x3 | Toggle. Toggle the corresponding External Match bit/output. | |
| 11:10 | EMC3 | | External Match Control 3. Determines the functionality of External Match 3. | 00 |
| | | 0x0 | Do Nothing. | |
| | | 0x1 | Clear. Clear the corresponding External Match bit/output to 0 (MAT3 pin is LOW if pinned out). | |
| | | 0x2 | Set. Set the corresponding External Match bit/output to 1 (MAT3 pin is HIGH if pinned out). | |
| | | 0x3 | Toggle. Toggle the corresponding External Match bit/output. | |
| 31:12 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

15.7.11 Count Control Register

The Count Control Register (CTCR) is used to select between Timer and Counter mode, and in Counter mode to select the pin and edge(s) for counting.

When Counter Mode is chosen as a mode of operation, the CAP input (selected by the CTCR bits 3:2) is sampled on every rising edge of the PCLK clock. After comparing two consecutive samples of this CAP input, one of the following four events is recognized: rising edge, falling edge, either of edges or no changes in the level of the selected CAP input. Only if the identified event occurs and the event corresponds to the one selected by bits 1:0 in the CTCR register, will the Timer Counter register be incremented.

Effective processing of the externally supplied clock to the counter has some limitations. Since two successive rising edges of the PCLK clock are used to identify only one edge on the CAP selected input, the frequency of the CAP input cannot exceed one half of the PCLK clock. Consequently, duration of the HIGH/LOW levels on the same CAP input in this case cannot be shorter than 1/PCLK.

Bits 7:4 of this register are also used to enable and configure the capture-clears-timer feature. This feature allows for a designated edge on a particular CAP input to reset the timer to all zeros. Using this mechanism to clear the timer on the leading edge of an input pulse and performing a capture on the trailing edge, permits direct pulse-width measurement using a single capture input without the need to perform a subtraction operation in software.

Table 259. Address map CTCR register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| CT32B0 | 0x400B 4000 | 0x070 | - | 1 |
| CT32B1 | 0x400B 8000 | 0x070 | - | 1 |
| CT32B2 | 0x4000 4000 | 0x070 | - | 1 |
| CT32B3 | 0x4000 8000 | 0x070 | - | 1 |
| CT32B4 | 0x4000 C000 | 0x070 | - | 1 |

Table 260. Count Control Register (CTCR, address offset 0x070) bit description

| Bit | Symbol | Value | Description | Reset Value |
|-----|--------|-------|---|-------------|
| 1:0 | CTMODE | | Counter/Timer Mode This field selects which rising PCLK edges can increment Timer's Prescale Counter (PC), or clear PC and increment Timer Counter (TC). Timer Mode: the TC is incremented when the Prescale Counter matches the Prescale Register. | 00 |
| | | 0x0 | Timer Mode. Incremented every rising PCLK edge. | |
| | | 0x1 | Counter Mode rising edge. TC is incremented on rising edges on the CAP input selected by bits 3:2. | |
| | | 0x2 | Counter Mode falling edge. TC is incremented on falling edges on the CAP input selected by bits 3:2. | |
| | | 0x3 | Counter Mode dual edge. TC is incremented on both edges on the CAP input selected by bits 3:2. | |

Table 260. Count Control Register (CTCR, address offset 0x070) bit description

| Bit | Symbol | Value | Description | Reset Value |
|------|--------|-------|--|-------------|
| 3:2 | CINSEL | | Count Input Select When bits 1:0 in this register are not 00, these bits select which CAP pin is sampled for clocking. Note: If Counter mode is selected for a particular CAPn input in the CTCR, the 3 bits for that input in the Capture Control Register (CCR) must be programmed as 000. However, capture and/or interrupt can be selected for the other 3 CAPn inputs in the same timer. | 0 |
| | | 0x0 | Channel 0. CAPn.0 for CT32Bn | |
| | | 0x1 | Channel 1. CAPn.1 for CT32Bn | |
| | | 0x2 | Channel 2. CAPn.2 for CT32Bn | |
| | | 0x3 | Channel 3. CAPn.3 for CT32Bn | |
| 4 | ENCC | | Setting this bit to 1 enables clearing of the timer and the prescaler when the capture-edge event specified in bits 7:5 occurs. | 0 |
| 7:5 | SELCC | | Edge select. When bit 4 is 1, these bits select which capture input edge will cause the timer and prescaler to be cleared. These bits have no effect when bit 4 is low. Values 0x2 to 0x3 and 0x6 to 0x7 are reserved. | 0 |
| | | 0x0 | Channel 0 Rising Edge. Rising edge of the signal on capture channel 0 clears the timer (if bit 4 is set). | |
| | | 0x1 | Channel 0 Falling Edge. Falling edge of the signal on capture channel 0 clears the timer (if bit 4 is set). | |
| | | 0x2 | Channel 1 Rising Edge. Rising edge of the signal on capture channel 1 clears the timer (if bit 4 is set). | |
| | | 0x3 | Channel 1 Falling Edge. Falling edge of the signal on capture channel 1 clears the timer (if bit 4 is set). | |
| | | 0x4 | Channel 2 Rising Edge. Rising edge of the signal on capture channel 2 clears the timer (if bit 4 is set). | |
| | | 0x5 | Channel 2 Falling Edge. Falling edge of the signal on capture channel 2 clears the timer (if bit 4 is set). | |
| 31:8 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

15.7.12 PWM Control Register

The PWM Control Register is used to configure the match outputs as PWM outputs. Each match output can be independently set to perform either as PWM output or as match output whose function is controlled by the External Match Register (EMR).

For each timer, a maximum of three single edge controlled PWM outputs can be selected on the MATn.2:0 outputs. One additional match register determines the PWM cycle length. When a match occurs in any of the other match registers, the PWM output is set to HIGH. The timer is reset by the match register that is configured to set the PWM cycle length. When the timer is reset to zero, all currently HIGH match outputs configured as PWM outputs are cleared.

Table 261. Address map PWMC register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| CT32B0 | 0x400B 4000 | 0x074 | - | 1 |
| CT32B1 | 0x400B 8000 | 0x074 | - | 1 |

Table 261. Address map PWMC register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| CT32B2 | 0x4000 4000 | 0x074 | - | 1 |
| CT32B3 | 0x4000 8000 | 0x074 | - | 1 |
| CT32B4 | 0x4000 C000 | 0x074 | - | 1 |

Table 262: PWM Control Register (PWMC, address offset 0x074) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---|-------------|
| 0 | PWMEN0 | | PWM mode enable for channel0. | 0 |
| | | 0 | Match. CT32Bn_MAT0 is controlled by EM0. | |
| | | 1 | PWM. PWM mode is enabled for CT32Bn_MAT0. | |
| 1 | PWMEN1 | | PWM mode enable for channel1. | 0 |
| | | 0 | Match. CT32Bn_MAT01 is controlled by EM1. | |
| | | 1 | PWM. PWM mode is enabled for CT32Bn_MAT1. | |
| 2 | PWMEN2 | | PWM mode enable for channel2. | 0 |
| | | 0 | Match. CT32Bn_MAT2 is controlled by EM2. | |
| | | 1 | PWM. PWM mode is enabled for CT32Bn_MAT2. | |
| 3 | PWMEN3 | | PWM mode enable for channel3. Note: It is recommended to use match channel 3 to set the PWM cycle. | 0 |
| | | 0 | Match. CT32Bn_MAT3 is controlled by EM3. | |
| | | 1 | PWM. PWM mode is enabled for CT132Bn_MAT3. | |
| 31:4 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

15.8 Functional description

Figure 30 shows a timer configured to reset the count and generate an interrupt on match. The prescaler is set to 2 and the match register set to 6. At the end of the timer cycle where the match occurs, the timer count is reset. This gives a full length cycle to the match value. The interrupt indicating that a match occurred is generated in the next clock after the timer reached the match value.

Figure 31 shows a timer configured to stop and generate an interrupt on match. The prescaler is again set to 2 and the match register set to 6. In the next clock after the timer reaches the match value, the timer enable bit in TCR is cleared, and the interrupt indicating that a match occurred is generated.

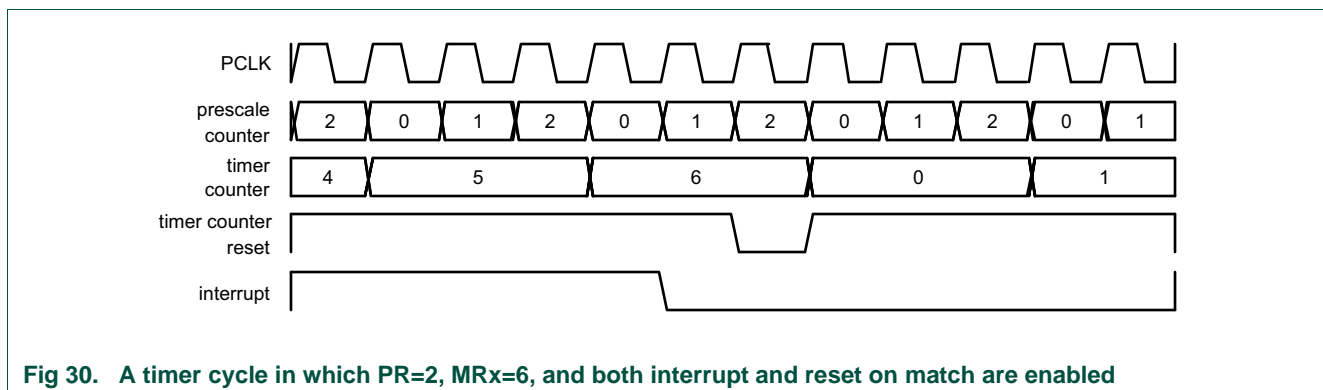


Fig 30. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

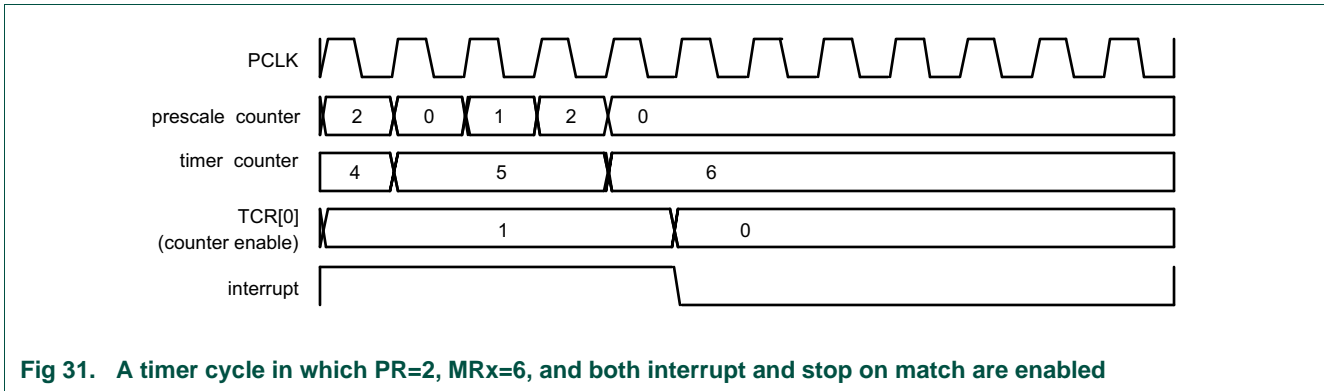


Fig 31. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

15.8.1 Rules for single edge controlled PWM outputs

1. All single edge controlled PWM outputs go LOW at the beginning of each PWM cycle (timer is set to zero) unless their match value is equal to zero.
2. Each PWM output will go HIGH when its match value is reached. If no match occurs (i.e. the match value is greater than the PWM cycle length), the PWM output remains continuously LOW.
3. If a match value larger than the PWM cycle length is written to the match register, and the PWM signal is HIGH already, then the PWM signal will be cleared with the start of the next PWM cycle.
4. If a match register contains the same value as the timer reset value (the PWM cycle length), then the PWM output will be reset to LOW on the next clock tick after the timer reaches the match value. Therefore, the PWM output will always consist of a one clock tick wide positive pulse with a period determined by the PWM cycle length (i.e. the timer reload value).
5. If a match register is set to zero, then the PWM output will go to HIGH the first time the timer goes back to zero and will stay HIGH continuously.

Note: When the match outputs are selected to perform as PWM outputs, the timer reset (MRnR) and timer stop (MRnS) bits in the Match Control Register MCR must be set to zero except for the match register setting the PWM cycle length. For this register, set the MRnR bit to one to enable the timer reset when the timer value matches the value of the corresponding match register.

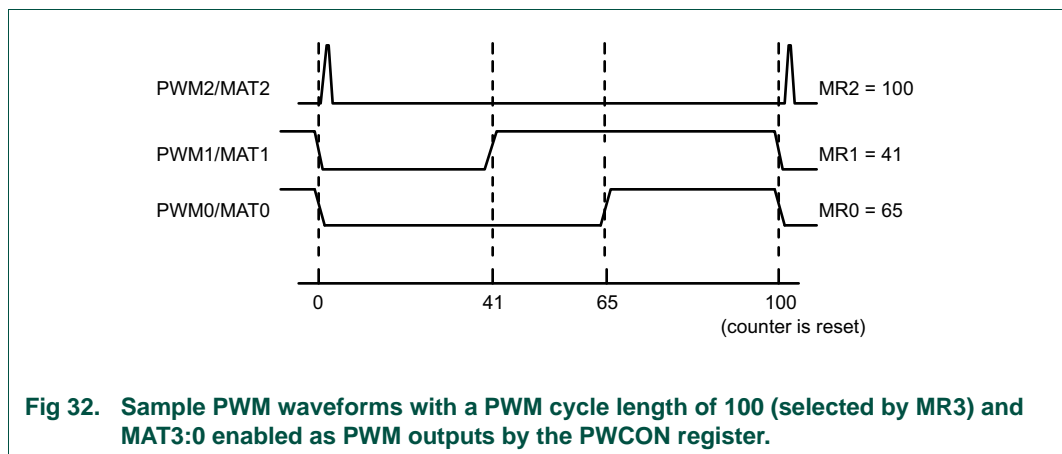


Fig 32. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.

15.8.2 DMA operation

DMA requests are generated by a match of the Timer Counter (TC) register value to either Match Register 0 (MR0) or Match Register 1 (MR1). This is not connected to the operation of the Match outputs controlled by the EMR register. Each match sets a DMA request flag, which is connected to the DMA controller. In order to have an effect, the DMA controller must be configured correctly.

When a timer is initially set up to generate a DMA request, the request may already be asserted before a match condition occurs. An initial DMA request may be avoided by having software write a one to the interrupt flag location, as if clearing a timer interrupt. See [Section 15.7.1](#). A DMA request will be cleared automatically when it is acted upon by the DMA controller.

Note: because timer DMA requests are generated whenever the timer value is equal to the related Match Register value, DMA requests are always generated when the timer is running, unless the Match Register value is higher than the upper count limit of the timer. It is important not to select and enable timer DMA requests in the DMA block unless the timer is correctly configured to generate valid DMA requests.

16.1 How to read this chapter

The watchdog timer is identical on all LPC5410x parts.

16.2 Features

- Internally resets chip if not reloaded during the programmable time-out period.
- Optional windowed operation requires reload to occur between a minimum and maximum time-out period, both programmable.
- Optional warning interrupt can be generated at a programmable time prior to watchdog time-out.
- Programmable 24-bit timer with internal fixed pre-scaler.
- Selectable time period from 1,024 watchdog clocks ($T_{WDCLK} \times 256 \times 4$) to over 67 million watchdog clocks ($T_{WDCLK} \times 2^{24} \times 4$) in increments of 4 watchdog clocks.
- “Safe” watchdog operation. Once enabled, requires a hardware reset or a Watchdog reset to be disabled.
- Incorrect feed sequence causes immediate watchdog event if enabled.
- The watchdog reload value can optionally be protected such that it can only be changed after the “warning interrupt” time is reached.
- Flag to indicate Watchdog reset.
- The Watchdog clock (WDCLK) source is the fixed 500 kHz clock (+/- 40%) provided by the low-power watchdog oscillator.
- The Watchdog timer can be configured to run in Deep-sleep or Power-down mode.
- Debug mode.

16.3 Basic configuration

The WWDT is configured through the following registers:

- Power to the register interface (WWDT PCLK clock): set the WWDT bit in the AHBCLKCTRL0 register, [Table 51](#).
- For waking up from a WWDT interrupt, enable the watchdog interrupt for wake-up in the STARTER0 register ([Table 75](#)).

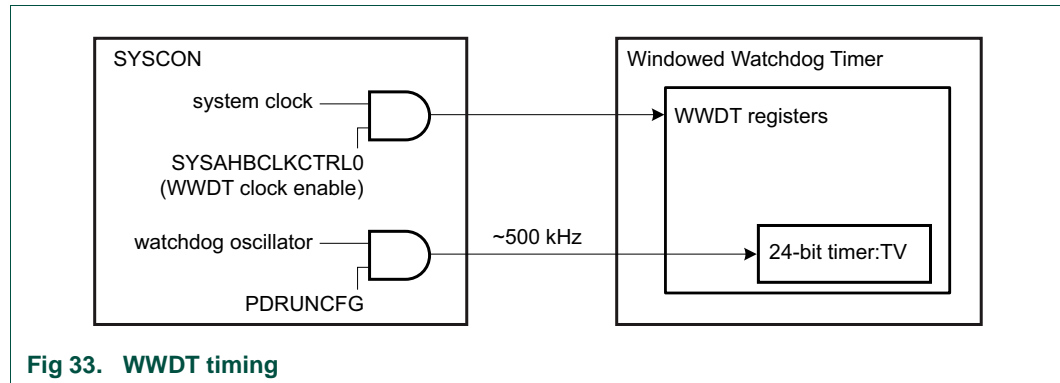


Fig 33. WWDT timing

16.4 Pin description

The WWDT has no external pins.

16.5 General description

The purpose of the Watchdog Timer is to reset or interrupt the microcontroller within a programmable time if it enters an erroneous state. When enabled, a watchdog reset is generated if the user program fails to feed (reload) the Watchdog within a predetermined amount of time.

When a watchdog window is programmed, an early watchdog feed is also treated as a watchdog event. This allows preventing situations where a system failure may still feed the watchdog. For example, application code could be stuck in an interrupt service that contains a watchdog feed. Setting the window such that this would result in an early feed will generate a watchdog event, allowing for system recovery.

The Watchdog consists of a fixed (divide by 4) pre-scaler and a 24-bit counter which decrements when clocked. The minimum value from which the counter decrements is 0xFF. Setting a value lower than 0xFF causes 0xFF to be loaded in the counter. Hence the minimum Watchdog interval is $(T_{WDCLK} \times 256 \times 4)$ and the maximum Watchdog interval is $(T_{WDCLK} \times 2^{24} \times 4)$ in multiples of $(T_{WDCLK} \times 4)$. The Watchdog should be used in the following manner:

- Set the Watchdog timer constant reload value in the TC register.
- Set the Watchdog timer operating mode in the MOD register.
- Set a value for the watchdog window time in the WINDOW register if windowed operation is desired.

- Set a value for the watchdog warning interrupt in the WARNINT register if a warning interrupt is desired.
- Enable the Watchdog by writing 0xAA followed by 0x55 to the FEED register.
- The Watchdog must be fed again before the Watchdog counter reaches zero in order to prevent a watchdog event. If a window value is programmed, the feed must also occur after the watchdog counter passes that value.

When the Watchdog Timer is configured so that a watchdog event will cause a reset and the counter reaches zero, the CPU will be reset, loading the stack pointer and program counter from the vector table as for an external reset. The Watchdog time-out flag (WDTOF) can be examined to determine if the Watchdog has caused the reset condition. The WDTOF flag must be cleared by software.

When the Watchdog Timer is configured to generate a warning interrupt, the interrupt will occur when the counter matches the value defined by the WARNINT register.

16.5.1 Block diagram

The block diagram of the Watchdog is shown below in the [Figure 34](#). The synchronization logic (PCLK - WDCLK) is not shown in the block diagram.

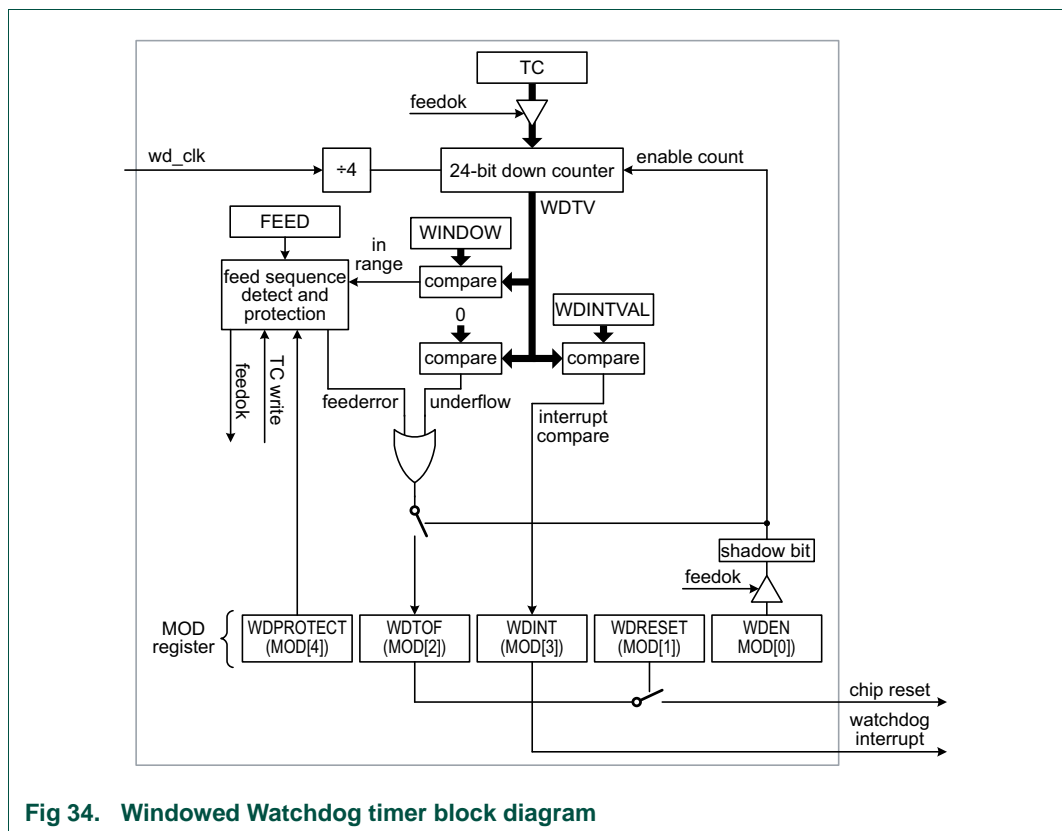


Fig 34. Windowed Watchdog timer block diagram

16.5.2 Clocking and power control

The watchdog timer block uses two clocks: PCLK and WDCLK. PCLK is used for the APB accesses to the watchdog registers and is derived from the system clock (see [Figure 3](#)). The WDCLK is used for the watchdog timer counting and is derived from the watchdog oscillator.

The synchronization logic between the two clock domains works as follows: When the MOD and TC registers are updated by APB operations, the new value will take effect in 3 WDCLK cycles on the logic in the WDCLK clock domain.

When the watchdog timer is counting on WDCLK, the synchronization logic will first lock the value of the counter on WDCLK and then synchronize it with PCLK, so that the CPU can read the TV register.

Remark: Because of the synchronization step, software must add a delay of three WDCLK clock cycles between the feed sequence and the time the WDPROTECT bit is enabled in the MOD register. The length of the delay depends on the selected watchdog clock WDCLK.

16.5.3 Using the WWDT lock features

The WWDT supports several lock features which can be enabled to ensure that the WWDT is running at all times:

- Disabling the WWDT clock source
- Changing the WWDT reload value

16.5.3.1 Disabling the WWDT clock source

If bit 5 in the WWDT MOD register is set, the WWDT clock source is locked and can not be disabled either by software or by hardware when Sleep, Deep-sleep or Power-down modes are entered. Therefore, the user must ensure that the watchdog oscillator for each power mode is enabled **before** setting bit 5 in the MOD register.

16.5.3.2 Changing the WWDT reload value

If bit 4 is set in the WWDT MOD register, the watchdog time-out value (TC) can be changed only after the counter is below the value of WDWARNINT and WDWINDOW.

The reload overwrite lock mechanism can only be disabled by a reset of any type.

16.6 Register description

The Watchdog Timer contains the registers shown in [Table 263](#).

The reset value reflects the data stored in used bits only. It does not include the content of reserved bits.

Table 263. Register overview: Watchdog timer (base address 0x4003 8000)

| Name | Access | Address offset | Description | Reset value | Reference |
|---------|--------|----------------|---|-------------|---------------------------|
| MOD | R/W | 0x000 | Watchdog mode register. This register contains the basic mode and status of the Watchdog Timer. | 0 | Table 264 |
| TC | R/W | 0x004 | Watchdog timer constant register. This 24-bit register determines the time-out value. | 0xFF | Table 266 |
| FEED | WO | 0x008 | Watchdog feed sequence register. Writing 0xAA followed by 0x55 to this register reloads the Watchdog timer with the value contained in the TC register. | NA | Table 267 |
| TV | RO | 0x00C | Watchdog timer value register. This 24-bit register reads out the current value of the Watchdog timer. | 0xFF | Table 268 |
| - | - | 0x010 | Reserved | - | - |
| WARNINT | R/W | 0x014 | Watchdog Warning Interrupt compare value. | 0 | Table 269 |
| WINDOW | R/W | 0x018 | Watchdog Window compare value. | 0xFF FFFF | Table 270 |

16.6.1 Watchdog mode register

The WDMOD register controls the operation of the Watchdog. Note that a watchdog feed must be performed before any changes to the WDMOD register take effect.

Table 264. Watchdog mode register (MOD, 0x4003 8000) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|---------|-------|--|-----------------------------------|
| 0 | WDEN | | Watchdog enable bit. Once this bit is set to one and a watchdog feed is performed, the watchdog timer will run permanently. | 0 |
| | | 0 | Stop. The watchdog timer is stopped. | |
| | | 1 | Run. The watchdog timer is running. | |
| 1 | WDRESET | | Watchdog reset enable bit. Once this bit has been written with a 1 it cannot be re-written with a 0. | 0 |
| | | 0 | Interrupt. A watchdog time-out will not cause a chip reset. | |
| | | 1 | Reset. A watchdog time-out will cause a chip reset. | |
| 2 | WDTOF | | Watchdog time-out flag. Set when the watchdog timer times out, by a feed error, or by events associated with WDPROTECT. Cleared by software. Causes a chip reset if WDRESET = 1. | 0 (following external reset only) |
| 3 | WDINT | | Warning interrupt flag. Set when the timer reaches the value in WDWARNINT. Cleared by software. | 0 |

Table 264. Watchdog mode register (MOD, 0x4003 8000) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|-----------|-------|---|-------------|
| 4 | WDPROTECT | | Watchdog update mode. This bit can be set once by software and is only cleared by a reset. | 0 |
| | | 0 | Flexible. The watchdog time-out value (TC) can be changed at any time. | |
| | | 1 | Threshold. The watchdog time-out value (TC) can be changed only after the counter is below the value of WDWARNINT and WDWINDOW. | |
| 5 | LOCK | | Once this bit is set to one and a watchdog feed is performed, disabling or powering down the watchdog oscillator is prevented by hardware. This bit can be set once by software and is only cleared by any reset. | 0 |
| 31:6 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

Once the **WDEN**, **WDPROTECT**, or **WDRESET** bits are set they can not be cleared by software. Both flags are cleared by an external reset or a Watchdog timer reset.

WDTOF The Watchdog time-out flag is set when the Watchdog times out, when a feed error occurs, or when PROTECT =1 and an attempt is made to write to the TC register. This flag is cleared by software writing a 0 to this bit.

WDINT The Watchdog interrupt flag is set when the Watchdog counter reaches the value specified by WARNINT. This flag is cleared when any reset occurs, and is cleared by software by writing a 0 to this bit.

In all power modes except Deep power-down mode, a Watchdog reset or interrupt can occur when the watchdog is running and has an operating clock source. The watchdog oscillator can be configured to keep running in Sleep, Deep-sleep modes, and Power-down modes.

If a watchdog interrupt occurs in Sleep, Deep-sleep mode, or Power-down mode, and the WWDT interrupt is enabled in the NVIC, the device will wake up. Note that in Deep-sleep and Power-down modes, the WWDT interrupt must be enabled in the STARTER0 register in addition to the NVIC.

See the following registers:

[Table 75 “Start enable register 0 \(STARTER0, address 0x4000 0240\) bit description”](#)

Table 265. Watchdog operating modes selection

| WDEN | WDRESET | Mode of Operation |
|------|------------|---|
| 0 | X (0 or 1) | Debug/Operate without the Watchdog running. |
| 1 | 0 | Watchdog interrupt mode: the watchdog warning interrupt will be generated but watchdog reset will not. When this mode is selected, the watchdog counter reaching the value specified by WDWARNINT will set the WDINT flag and the Watchdog interrupt request will be generated. |
| 1 | 1 | Watchdog reset mode: both the watchdog interrupt and watchdog reset are enabled. When this mode is selected, the watchdog counter reaching the value specified by WDWARNINT will set the WDINT flag and the Watchdog interrupt request will be generated, and the watchdog counter reaching zero will reset the microcontroller. A watchdog feed prior to reaching the value of WDWINDOW will also cause a watchdog reset. |

16.6.2 Watchdog Timer Constant register

The TC register determines the time-out value. Every time a feed sequence occurs the value in the TC is loaded into the Watchdog timer. The TC resets to 0x00 00FF. Writing a value below 0xFF will cause 0x00 00FF to be loaded into the TC. Thus the minimum time-out interval is $T_{WDCLK} \times 256 \times 4$.

If the WDPROTECT bit in WDMOD = 1, an attempt to change the value of TC before the watchdog counter is below the values of WDWARNINT and WDWINDOW will cause a watchdog reset and set the WDTOF flag.

Table 266. Watchdog Timer Constant register (TC, 0x4003 8004) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 23:0 | COUNT | Watchdog time-out value. | 0x00 00FF |
| 31:24 | - | Reserved. Read value is undefined, only zero should be written. | NA |

16.6.3 Watchdog Feed register

Writing 0xAA followed by 0x55 to this register will reload the Watchdog timer with the TC register value. This operation will also start the Watchdog if it is enabled via the WDMOD register. Setting the WDEN bit in the WDMOD register is not sufficient to enable the Watchdog. A valid feed sequence must be completed after setting WDEN before the Watchdog is capable of generating a reset. Until then, the Watchdog will ignore feed errors.

After writing 0xAA to WDFEED, access to any Watchdog register other than writing 0x55 to WDFEED causes an immediate reset/interrupt when the Watchdog is enabled, and sets the WDTOF flag. The reset will be generated during the second PCLK following an incorrect access to a Watchdog register during a feed sequence.

It is good practice to disable interrupts around a feed sequence, if the application is such that an interrupt might result in rescheduling processor control away from the current task in the middle of the feed, and then lead to some other access to the WDT before control is returned to the interrupted task.

Table 267. Watchdog Feed register (FEED, 0x4003 8008) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 7:0 | FEED | Feed value should be 0xAA followed by 0x55. | NA |
| 31:8 | - | Reserved. Read value is undefined, only zero should be written. | NA |

16.6.4 Watchdog Timer Value register

The TV register is used to read the current value of Watchdog timer counter.

When reading the value of the 24-bit counter, the lock and synchronization procedure takes up to 6 WDCLK cycles plus 6 PCLK cycles, so the value of TV is older than the actual value of the timer when it's being read by the CPU.

Table 268. Watchdog Timer Value register (TV, 0x4003 800C) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 23:0 | COUNT | Counter timer value. | 0x00 00FF |
| 31:24 | - | Reserved. Read value is undefined, only zero should be written. | NA |

16.6.5 Watchdog Timer Warning Interrupt register

The WDWARNINT register determines the watchdog timer counter value that will generate a watchdog interrupt. When the watchdog timer counter matches the value defined by WARNINT, an interrupt will be generated after the subsequent WDCLK.

A match of the watchdog timer counter to WARNINT occurs when the bottom 10 bits of the counter have the same value as the 10 bits of WARNINT, and the remaining upper bits of the counter are all 0. This gives a maximum time of 1,023 watchdog timer counts (4,096 watchdog clocks) for the interrupt to occur prior to a watchdog event. If WARNINT is 0, the interrupt will occur at the same time as the watchdog event.

Table 269. Watchdog Timer Warning Interrupt register (WARNINT, 0x4003 8014) bit description

| Bit | Symbol | Description | Reset value |
|-------|---------|---|-------------|
| 9:0 | WARNINT | Watchdog warning interrupt compare value. | 0 |
| 31:10 | - | Reserved, only zero should be written. | NA |

16.6.6 Watchdog Timer Window register

The WINDOW register determines the highest TV value allowed when a watchdog feed is performed. If a feed sequence occurs when TV is greater than the value in WINDOW, a watchdog event will occur.

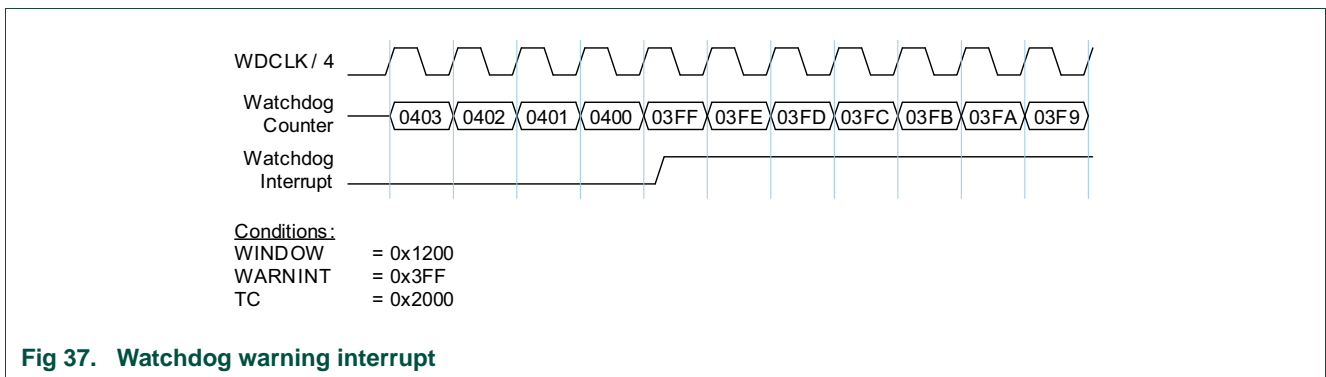
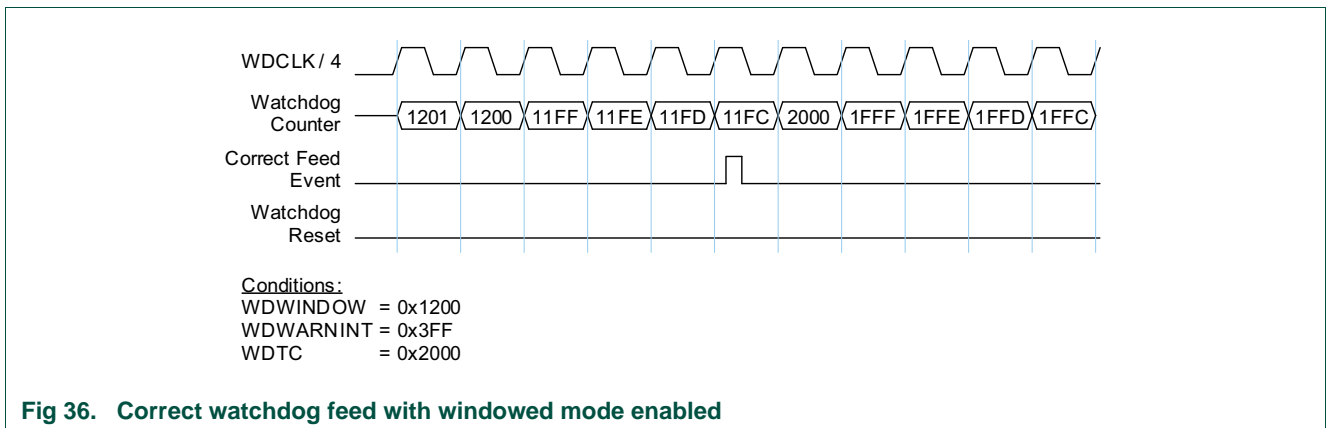
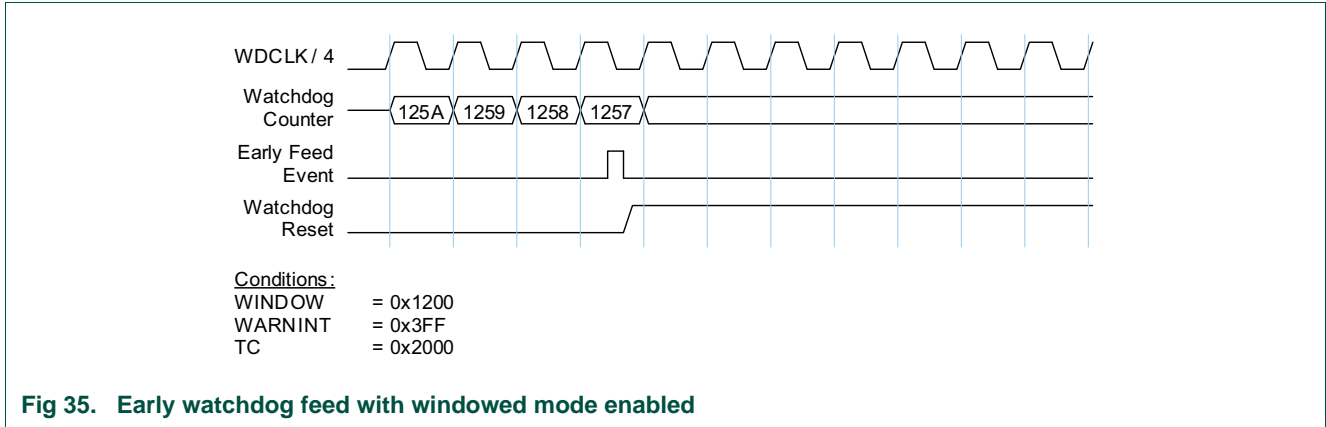
WINDOW resets to the maximum possible TV value, so windowing is not in effect.

Table 270. Watchdog Timer Window register (WINDOW, 0x4003 8018) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 23:0 | WINDOW | Watchdog window value. | 0xFF FFFF |
| 31:24 | - | Reserved, only zero should be written. | NA |

16.7 Functional description

The following figures illustrate several aspects of Watchdog Timer operation.



17.1 How to read this chapter

The RTC is identical on all LPC5410x parts.

17.2 Features

- The RTC and its independent oscillator operate directly from the device power pins, not using the on-chip regulator. The RTC oscillator has the following clock outputs:
 - 32 kHz clock, selectable for system clock and CLKOUT pin.
 - 1 Hz clock for RTC timing.
 - 1 kHz clock for high-resolution RTC timing.
- 32-bit, 1 Hz RTC counter and associated match register for alarm generation.
- Separate 16-bit high-resolution/wake-up timer clocked at 1 kHz for 1 ms resolution with a more that one minute maximum time-out period.
- RTC alarm and high-resolution/wake-up timer time-out each generate independent interrupt requests. Either time-out can wake up the part from any of the low power modes, including Deep power-down.

17.3 Basic configuration

Configure the RTC as follows:

- Use the AHBCLKCTRL0 register ([Table 51](#)) to enable the clock to the RTC register interface and peripheral clock.
- For RTC software reset use the RTC CTRL register. See [Table 273](#). The RTC is reset only by initial power-up of the device or when an RTC software reset is applied, it is not initialized by other system resets.
- The RTC provides an interrupt to NVIC slot #29 for the RTC_WAKE and RTC_ALARM functions.
- To enable the RTC interrupts for waking up from Deep-sleep and Power-down modes, enable the interrupts in the STARTER0 register ([Table 75](#)) and the NVIC.
- To enable the RTC interrupts for waking up from Deep power-down, enable the appropriate RTC clock and wake-up in the RTC CTRL register ([Table 273](#)).
- If enabled, the RTC and its oscillator continue running in all reduced power modes as long as power is supplied to the device. So, the 32 kHz output is always available to be enabled for syscon clock generation (see [Table 65](#)). Once enabled, the 32 kHz clock can be selected for the system clock or be observed through the CLKOUT pin. The 1 Hz output is enabled in the RTC CTRL register (RTC_EN bit). Once the 1 Hz output is enabled, the 1 kHz output for the high-resolution wake-up timer can be enabled in the RTC CTRL register (RTC1KHZ_EN bit).
- If the 32 kHz output of the RTC is used by another part of the system, enable it via the EN bit in the RTCOSCCTRL register. See [Table 65](#).

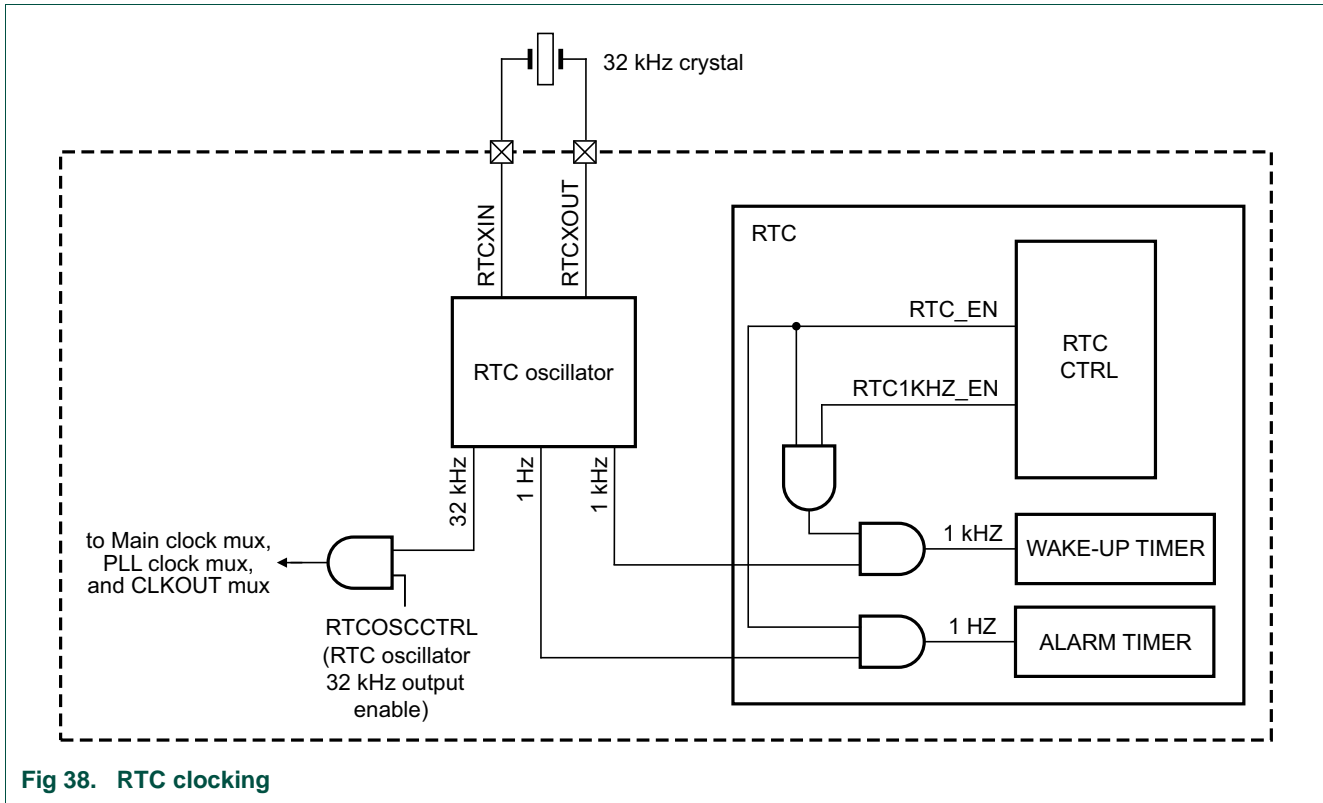


Fig 38. RTC clocking

17.3.1 RTC timers

The RTC contains two timers:

1. The main RTC timer. This 32-bit timer uses a 1 Hz clock and is intended to run continuously as a real-time clock. When the timer value reaches a match value, an interrupt is raised. The alarm interrupt can also wake up the part from any low power mode if enabled.
2. The high-resolution/wake-up timer. This 16-bit timer uses a 1 kHz clock and operates as a one-shot down timer. Once the timer is loaded, it starts counting down to 0 at which point an interrupt is raised. The interrupt can wake up the part from any low power mode if enabled. This timer is intended to be used for timed wake-up from Deep-sleep, power-down, or Deep power-down modes. The high-resolution wake-up timer can be disabled to conserve power if not used.

17.4 General description

17.4.1 Real-time clock

The real-time clock is a 32-bit up-counter which can be cleared or initialized by software. Once enabled, it counts continuously at a 1 Hz clock rate as long as the device is powered up and the RTC remains enabled.

The main purpose of the RTC is to count seconds and generate an alarm interrupt to the processor whenever the counter value equals the value programmed into the associated 32-bit match register.

If the part is in one of the reduced-power modes (deep-sleep, power-down, deep power-down) an RTC alarm interrupt can also wake up the part to exit the power mode and begin normal operation.

17.4.2 High-resolution/wake-up timer

The time interval required for many applications, including waking the part up from a low-power mode, will often demand a greater degree of resolution than the one-second minimum interval afforded by the main RTC counter. For these applications, a higher frequency secondary timer has been provided.

This secondary timer is an independent, stand-alone wake-up or general-purpose timer for timing intervals of up to 64 seconds with approximately one millisecond of resolution.

The High-Resolution/Wake-up Timer is a 16-bit down counter which is clocked at a 1 kHz rate when it is enabled. Writing any non-zero value to this timer will automatically enable the counter and launch a countdown sequence. When the counter is being used as a wake-up timer, this write can occur just prior to entering a reduced power mode.

When a starting count value is loaded, the High-Resolution/Wake-up Timer will turn on, count from the pre-loaded value down to zero, generate an interrupt and/or a wake-up command, and then turn itself off until re-launched by a subsequent software write.

17.4.3 RTC power

The RTC module and the oscillator that drives it, run directly from device power pins. As a result, the RTC will continue operating in deep power-down mode when power is internally turned off to the rest of the device.

17.5 Pin description

[Table 271](#) gives a summary of pins related to the RTC.

Table 271. RTC pin description

| Pin | Type | Description |
|---------|--------|------------------------|
| RTCXIN | Input | RTC oscillator input. |
| RTCXOUT | Output | RTC oscillator output. |

17.6 Register description

Reset Values pertain to initial power-up of the device or when an RTC software reset is applied (except where noted). This block is not initialized by any other system reset.

Table 272. Register overview: RTC (base address 0x4003 C000)

| Name | Access | Offset | Description | Reset value | Reference |
|-------|--------|--------|--|-------------|---------------------------|
| CTRL | R/W | 0x000 | RTC control register | 0xF | Table 273 |
| MATCH | R/W | 0x004 | RTC match register | 0xFFFF | Table 274 |
| COUNT | R/W | 0x008 | RTC counter register | 0 | Table 275 |
| WAKE | R/W | 0x00C | RTC high-resolution/wake-up timer control register | 0 | Table 276 |

17.6.1 RTC CTRL register

This register controls which clock the RTC uses (1 kHz or 1 Hz) and enables the two RTC interrupts to wake up the part from Deep power-down. To wake up the part from Deep-sleep or Power-down modes, enable the RTC interrupts in the system control block STARTLOGIC1 register.

Table 273. RTC control register (CTRL, address 0x4003 C000) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|----------|-------|---|-------------|
| 0 | SWRESET | | Software reset control | 1 |
| | | 0 | Not in reset. The RTC is not held in reset. This bit must be cleared prior to configuring or initiating any operation of the RTC. | |
| | | 1 | In reset. The RTC is held in reset. All register bits within the RTC will be forced to their reset value except the OFD bit. This bit must be cleared before writing to any register in the RTC - including writes to set any of the other bits within this register. Do not attempt to write to any bits of this register at the same time that the reset bit is being cleared. | |
| 1 | OFD | | Oscillator fail detect status. | 1 |
| | | 0 | Run. The RTC oscillator is running properly. Writing a 0 has no effect. | |
| | | 1 | Fail. RTC oscillator fail detected. Clear this flag after the following power-up. Writing a 1 clears this bit. | |
| 2 | ALARM1HZ | | RTC 1 Hz timer alarm flag status. | 1 |
| | | 0 | No match. No match has occurred on the 1 Hz RTC timer. Writing a 0 has no effect. | |
| | | 1 | Match. A match condition has occurred on the 1 Hz RTC timer. This flag generates an RTC alarm interrupt request RTC_ALARM which can also wake up the part from any low power mode. Writing a 1 clears this bit. | |
| 3 | WAKE1KHZ | | RTC 1 kHz timer wake-up flag status. | 1 |
| | | 0 | Run. The RTC 1 kHz timer is running. Writing a 0 has no effect. | |
| | | 1 | Time-out. The 1 kHz high-resolution/wake-up timer has timed out. This flag generates an RTC wake-up interrupt request RTC-WAKE which can also wake up the part from any low power mode. Writing a 1 clears this bit. | |

Table 273. RTC control register (CTRL, address 0x4003 C000) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|-------------|-------|--|-------------|
| 4 | ALARMDPD_EN | | RTC 1 Hz timer alarm enable for Deep power-down. | 0 |
| | | 0 | Disable. A match on the 1 Hz RTC timer will not bring the part out of Deep power-down mode. | |
| | | 1 | Enable. A match on the 1 Hz RTC timer bring the part out of Deep power-down mode. | |
| 5 | WAKEDPD_EN | | RTC 1 kHz timer wake-up enable for Deep power-down. | 0 |
| | | 0 | Disable. A match on the 1 kHz RTC timer will not bring the part out of Deep power-down mode. | |
| | | 1 | Enable. A match on the 1 kHz RTC timer bring the part out of Deep power-down mode. | |
| 6 | RTC1KHZ_EN | | RTC 1 kHz clock enable. This bit can be set to 0 to conserve power if the 1 kHz timer is not used. This bit has no effect when the RTC is disabled (bit 7 of this register is 0). | 0 |
| | | 0 | Disable. A match on the 1 kHz RTC timer will not bring the part out of Deep power-down mode. | |
| | | 1 | Enable. The 1 kHz RTC timer is enabled. | |
| 7 | RTC_EN | | RTC enable. | 0 |
| | | 0 | Disable. The RTC 1 Hz and 1 kHz clocks are shut down and the RTC operation is disabled. This bit should be 0 when writing to load a value in the RTC counter register. | |
| | | 1 | Enable. The 1 Hz RTC clock is running and RTC operation is enabled. This bit must be set to initiate operation of the RTC. The first clock to the RTC counter occurs 1 s after this bit is set. To also enable the high-resolution, 1 kHz clock, set bit 6 in this register. | |
| 31:8 | - | | Reserved. Read value is undefined, only zero should be written. | 0 |

17.6.2 RTC match register

Table 274. RTC match register (MATCH, address 0x4003 C004) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 31:0 | MATVAL | Contains the match value against which the 1 Hz RTC timer will be compared to generate set the alarm flag RTC_ALARM and generate an alarm interrupt/wake-up if enabled. | 0xFFFF |

17.6.3 RTC counter register

Table 275. RTC counter register (COUNT, address 0x4003 C008) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|--|-------------|
| 31:0 | VAL | A read reflects the current value of the main, 1 Hz RTC timer. A write loads a new initial value into the timer. The RTC counter will count up continuously at a 1 Hz rate once the RTC Software Reset is removed (by clearing bit 0 of the CTRL register). Remark: Only write to this register when the RTC_EN bit in the RTC CTRL Register is 0. The counter increments one second after the RTC_EN bit is set. | 0 |

17.6.4 RTC high-resolution/wake-up register

Table 276. RTC high-resolution/wake-up register (WAKE, address 0x4003 C00C) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 15:0 | VAL | A read reflects the current value of the high-resolution/wake-up timer. A write pre-loads a start count value into the wake-up timer and initializes a count-down sequence. Do not write to this register while counting is in progress. | 0 |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | |

18.1 How to read this chapter

The MRT is available on all LPC5410x parts.

18.2 Features

- 24-bit interrupt timer
- Four channels independently counting down from individually set values
- Repeat interrupt, one-shot interrupt, and one-shot bus stall modes

18.3 Basic configuration

Configure the MRT using the following registers:

- In the AHBCLKCTRL1 register ([Table 52](#)), set the MRT bit to enable the clock to the register interface.
- Clear the MRT reset using the PRESETCTRL1 register ([Table 36](#)).
- The global MRT interrupt is connected to an interrupt slot in the NVIC (see [Table 2](#)).

18.4 Pin description

The MRT is not associated with any device pins.

18.5 General description

The Multi-Rate Timer (MRT) provides a repetitive interrupt timer with four channels. Each channel can be programmed with an independent time interval.

Each channel operates independently from the other channels in one of the following modes:

- Repeat interrupt mode. See [Section 18.5.1](#).
- One-shot interrupt mode. See [Section 18.5.2](#).
- One-shot stall mode. See [Section 18.5.3](#).

The modes for each timer are set in the timer's control register. See [Table 280](#).

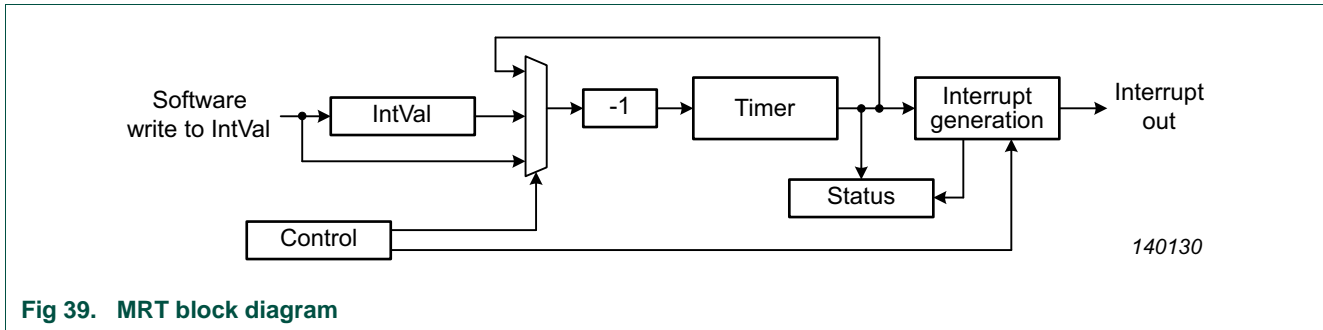


Fig 39. MRT block diagram

18.5.1 Repeat interrupt mode

The repeat interrupt mode generates repeated interrupts after a selected time interval. This mode can be used for software-based PWM or PPM applications.

When the timer n is in idle state, writing a non-zero value $IVALUE$ to the $INTVALn$ register immediately loads the time interval value $IVALUE - 1$, and the timer begins to count down from this value. When the timer reaches zero, an interrupt is generated, the value in the $INTVALn$ register $IVALUE - 1$ is reloaded automatically, and the timer starts to count down again.

While the timer is running in repeat interrupt mode, the following actions can be performed:

- Change the interval value on the next timer cycle by writing a new value (>0) to the $INTVALn$ register and setting the LOAD bit to 0. An interrupt is generated when the timer reaches zero. On the next cycle, the timer counts down from the new value.
- Change the interval value on-the-fly immediately by writing a new value (>0) to the $INTVALn$ register and setting the LOAD bit to 1. The timer immediately starts to count down from the new timer interval value. An interrupt is generated when the timer reaches 0.
- Stop the timer at the end of time interval by writing a 0 to the $INTVALn$ register and setting the LOAD bit to 0. An interrupt is generated when the timer reaches zero.
- Stop the timer immediately by writing a 0 to the $INTVALn$ register and setting the LOAD bit to 1. No interrupt is generated when the $INTVALn$ register is written.

18.5.2 One-shot interrupt mode

The one-shot interrupt generates one interrupt after a one-time count. With this mode, a single interrupt can be generated at any point. This mode can be used to introduce a specific delay in a software task.

When the timer is in the idle state, writing a non-zero value $IVALUE$ to the $INTVALn$ register immediately loads the time interval value $IVALUE - 1$, and the timer starts to count down. When the timer reaches 0, an interrupt is generated and the timer stops and enters the idle state.

While the timer is running in the one-shot interrupt mode, the following actions can be performed:

- Update the INTVALn register with a new time interval value (>0) and set the LOAD bit to 1. The timer immediately reloads the new time interval, and starts counting down from the new value. No interrupt is generated when the TIME_INTVALn register is updated.
- Write a 0 to the INTVALn register and set the LOAD bit to 1. The timer immediately stops counting and moves to the idle state. No interrupt is generated when the INTVALn register is updated.

18.5.3 One-shot stall mode

One-shot stall mode is similar to one-shot interrupt mode, except that it is intended for very short delays, for instance when the delay needed is less than the time it takes to get to an interrupt service routine. This mode is designed for very low software overhead, requiring only a single write to the INTVAL register (if the channel is already configured for one-shot stall mode). The MRT times the requested delay while stalling the bus write operation, concluding the write when the delay is complete. No interrupt or status polling is needed.

Bus stall mode can be used when a short delay is need between two software controlled events, or when a delay is expected before software can continue. Since in this mode there are no bus transactions while the MRT is counting down, the PCU consumes a minimum amount of power during that time.

Note that bus stall mode provides a minimum amount of time between the execution of the instruction that performs the write to INTVAL and the time that software continues. Other system events, such as interrupts or other bus masters accessing the APB bus where the MRT resides, can cause the delay to be longer.

18.6 Register description

The reset values shown in [Table 277](#) are POR reset values.

Table 277. Register overview: MRT (base address 0x4007 4000)

| Name | Access | Address offset | Description | Reset value | Reference |
|------------------------------|--------|----------------|---|-------------|---------------------------|
| MRT Timer 0 registers | | | | | |
| INTVAL0 | R/W | 0x0 | MRT0 Time interval value register. This value is loaded into the TIMER0 register. | 0 | Table 278 |
| TIMER0 | RO | 0x4 | MRT0 Timer register. This register reads the value of the down-counter. | 0xFF FFFF | Table 279 |
| CTRL0 | R/W | 0x8 | MRT0 Control register. This register controls the MRT0 modes. | 0 | Table 280 |
| STAT0 | R/W | 0xC | MRT0 Status register. | 0 | Table 281 |
| MRT Timer 1 registers | | | | | |
| INTVAL1 | R/W | 0x10 | MRT1 Time interval value register. This value is loaded into the TIMER1 register. | 0 | Table 278 |
| TIMER1 | R/W | 0x14 | MRT1 Timer register. This register reads the value of the down-counter. | 0xFF FFFF | Table 279 |
| CTRL1 | R/W | 0x18 | MRT1 Control register. This register controls the MRT1 modes. | 0 | Table 280 |
| STAT1 | R/W | 0x1C | MRT1 Status register. | 0 | Table 281 |
| MRT Timer 2 registers | | | | | |
| INTVAL2 | R/W | 0x20 | MRT2 Time interval value register. This value is loaded into the TIMER2 register. | 0 | Table 278 |
| TIMER2 | R/W | 0x24 | MRT2 Timer register. This register reads the value of the down-counter. | 0xFF FFFF | Table 279 |
| CTRL2 | R/W | 0x28 | MRT2 Control register. This register controls the MRT2 modes. | 0 | Table 280 |
| STAT2 | R/W | 0x2C | MRT2 Status register. | 0 | Table 281 |
| MRT Timer 3 registers | | | | | |
| INTVAL3 | R/W | 0x30 | MRT3 Time interval value register. This value is loaded into the TIMER3 register. | 0 | Table 278 |
| TIMER3 | R/W | 0x34 | MRT3 Timer register. This register reads the value of the down-counter. | 0xFF FFFF | Table 279 |
| CTRL3 | R/W | 0x38 | MRT3 Control register. This register controls the MRT modes. | 0 | Table 280 |
| STAT3 | R/W | 0x3C | MRT3 Status register. | 0 | Table 281 |
| Global MRT registers | | | | | |
| MODCFG | R/W | 0xF0 | Module Configuration register. This register provides information about this particular MRT instance, and allows choosing an overall mode for the idle channel feature. | 0 | Table 282 |
| IDLE_CH | RO | 0xF4 | Idle channel register. This register returns the number of the first idle channel. | 0 | Table 283 |
| IRQ_FLAG | R/W | 0xF8 | Global interrupt flag register | 0 | Table 284 |

18.6.1 Time interval register (INTVAL)

This register contains the MRT load value and controls how the timer is reloaded. The load value is IVALUE -1.

Table 278. Time interval register (INTVAL[0:3], address 0x4007 4000 (INTVAL0) to 0x4007 4030 (INTVAL3)) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|--|-------------|
| 23:0 | IVALUE | | Time interval load value. This value is loaded into the TIMERN register and the MRT channel n starts counting down from IVALUE -1. If the timer is idle, writing a non-zero value to this bit field starts the timer immediately. If the timer is running, writing a zero to this bit field does the following: <ul style="list-style-type: none"> • If LOAD = 1, the timer stops immediately. • If LOAD = 0, the timer stops at the end of the time interval. | 0 |
| 30:24 | - | | Reserved. Read value is undefined, only zero should be written. | - |
| 31 | LOAD | | Determines how the timer interval value IVALUE -1 is loaded into the TIMERN register. This bit is write-only. Reading this bit always returns 0. | 0 |
| | | 0 | No force load. The load from the INTVALn register to the TIMERN register is processed at the end of the time interval if the repeat mode is selected. | |
| | | 1 | Force load. The INTVALn interval value IVALUE -1 is immediately loaded into the TIMERN register while TIMERN is running. | |

18.6.2 Timer register (TIMER)

The timer register holds the current timer value. This register is read-only.

Table 279. Timer register (TIMER[0:3], address 0x4007 4004 (TIMER0) to 0x4007 4034 (TIMER3)) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 23:0 | VALUE | Holds the current timer value of the down-counter. The initial value of the TIMERN register is loaded as IVALUE - 1 from the INTVALn register either at the end of the time interval or immediately in the following cases: INTVALn register is updated in the idle state. INTVALn register is updated with LOAD = 1. When the timer is in idle state, reading this bit fields returns -1 (0x00FF FFFF). | 0x00FF FFFF |
| 31:24 | - | Reserved. Read value is undefined, only zero should be written. | 0 |

18.6.3 Control register (CTRL)

The control register configures the mode for each MRT and enables the interrupt.

Table 280. Control register (CTRL[0:3], address 0x4007 4008 (CTRL0) to 0x4007 4038 (CTRL3)) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|---|-------------|
| 0 | INTEN | | Enable the TIMERN interrupt. | 0 |
| | | 0 | Disabled. TIMERN interrupt is disabled. | |
| | | 1 | Enabled. TIMERN interrupt is enabled. | |
| 2:1 | MODE | | Selects timer mode. | 0 |
| | | 0x0 | Repeat interrupt mode. | |
| | | 0x1 | One-shot interrupt mode. | |
| | | 0x2 | One-shot stall mode. | |
| | | 0x3 | Reserved. | |
| 31:3 | - | | Reserved. | 0 |

18.6.4 Status register (STAT)

This register indicates the status of each MRT.

Table 281. Status register (STAT[0:3], address 0x4007 400C (STAT0) to 0x4007 403C (STAT3)) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|---------|-------|---|-------------|
| 0 | INTFLAG | | Monitors the interrupt flag. | 0 |
| | | 0 | No pending interrupt. Writing a zero is equivalent to no operation. | |
| | | 1 | Pending interrupt. The interrupt is pending because TIMERN has reached the end of the time interval. If the INTEN bit in the CONTROLn is also set to 1, the interrupt for timer channel n and the global interrupt are raised. Writing a 1 to this bit clears the interrupt request. | |
| 1 | RUN | | Indicates the state of TIMERN. This bit is read-only. | 0 |
| | | 0 | Idle state. TIMERN is stopped. | |
| | | 1 | Running. TIMERN is running. | |
| 2 | INUSE | | Channel In Use flag. Operating details depend on the MULTITASK bit in the MODCFG register, and affects the use of IDLE_CH. See Section 18.6.6 "Idle channel register (IDLE_CH)" for details of the two operating modes. | 0 |
| | | 0 | This channel is not in use. | |
| | | 1 | This channel is in use. | |
| 31:2 | - | | Reserved. | 0 |

18.6.5 Module Configuration register (MODCFG)

The MODCFG register provides the configuration (number of channels and timer width) for this MRT. See [Section 18.6.6 "Idle channel register \(IDLE_CH\)"](#) for details.

Table 282. Module Configuration register (MODCFG, address 0x4007 40F0) bit description

| Bit | Symbol | Value | Description | Reset Value |
|------|-----------|-------|--|-------------|
| 3:0 | NOC | | Identifies the number of channels in this MRT. | 4 |
| 8:4 | NOB | | Identifies the number of timer bits in this MRT. | 24 |
| 30:9 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 31 | MULTITASK | | Selects the operating mode for the INUSE flags and the IDLE_CH register. | 0 |
| | | 0 | Hardware status mode. In this mode, the INUSE(n) flags for all channels are reset. | |
| | | 1 | Multi-task mode. | |

18.6.6 Idle channel register (IDLE_CH)

The idle channel register can be used to assist software in finding available channels in the MRT. This allows more flexibility by not giving hard assignments to software that makes use of the MRT, without the need to search for an available channel. Generally, IDLE_CH returns the lowest available channel number.

IDLE_CH can be used in two ways, controlled by the value of the MULTITASK bit in the MODCFG register. MULTITASK affects both the function of IDLE_CH, and the function of the INUSE bit for each MRT channel as follows:

- MULTITASK = 0: hardware status mode. The INUSE flags for all MRT channels are reset. IDLECH returns the lowest idle channel number. A channel is considered idle if its RUN flag = 0, and there is no interrupt pending for that channel.
- MULTITASK = 1: multi-task mode. In this mode, the INUSE flags allow more control over when MRT channels are released for further use. When IDLE_CH is read, returning a channel number of an idle channel, the INUSE flag for that channel is set by hardware. That channel will not be considered idle until its RUN flag = 0, there is no interrupt pending, and its INUSE flag = 0. This allows reserving an MRT channel with a single register read, and no need to start the channel before it is no longer considered idle by IDLE_CH. It also allows software to identify a specific MRT channel that it can use, then use it more than once without releasing it, removing the need to ask for an available channel for every use.

Table 283. Idle channel register (IDLE_CH, address 0x4007 40F4) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 3:0 | - | Reserved. | 0 |
| 7:4 | CHAN | Idle channel. Reading the CHAN bits, returns the lowest idle timer channel. The number is positioned such that it can be used as an offset from the MRT base address in order to access the registers for the allocated channel. If all timer channels are running, CHAN = 0xF. See text above for more details. | 0 |
| 31:8 | - | Reserved. | 0 |

18.6.7 Global interrupt flag register (IRQ_FLAG)

The global interrupt register combines the interrupt flags from the individual timer channels in one register. Setting and clearing each flag behaves in the same way as setting and clearing the INTFLAG bit in each of the STATUSn registers.

Table 284. Global interrupt flag register (IRQ_FLAG, address 0x4007 40F8) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|--|-------------|
| 0 | GFLAG0 | | Monitors the interrupt flag of TIMER0. | 0 |
| | | 0 | No pending interrupt. Writing a zero is equivalent to no operation. | |
| | | 1 | Pending interrupt. The interrupt is pending because TIMER0 has reached the end of the time interval. If the INTEN bit in the CONTROL0 register is also set to 1, the interrupt for timer channel 0 and the global interrupt are raised. Writing a 1 to this bit clears the interrupt request. | |
| 1 | GFLAG1 | | Monitors the interrupt flag of TIMER1. See description of channel 0. | 0 |
| 2 | GFLAG2 | | Monitors the interrupt flag of TIMER2. See description of channel 0. | 0 |
| 3 | GFLAG3 | | Monitors the interrupt flag of TIMER3. See description of channel 0. | 0 |
| 31:4 | - | | Reserved. Read value is undefined, only zero should be written. | 0 |

19.1 How to read this chapter

19.2 Basic configuration

The RI timer is configured through the following registers:

- Power to the register interface: set the RIT bit in the AHBCLKCTRL1 register, [Table 52](#).

19.3 Features

- 48-bit counter running from the main clock. Counter can be free-running or be reset by a generated interrupt.
- 48-bit compare value.
- 48-bit compare mask. An interrupt is generated when the counter value equals the compare value, after masking. This allows for combinations not possible with a simple compare.

19.4 General description

The Repetitive Interrupt Timer (RIT) provides a versatile means of generating interrupts at specified time intervals, without using a standard timer. It is intended for repeating interrupts that aren't related to Operating System interrupts. The RIT could also be used as an alternative to the System Tick Timer if there are different system requirements.

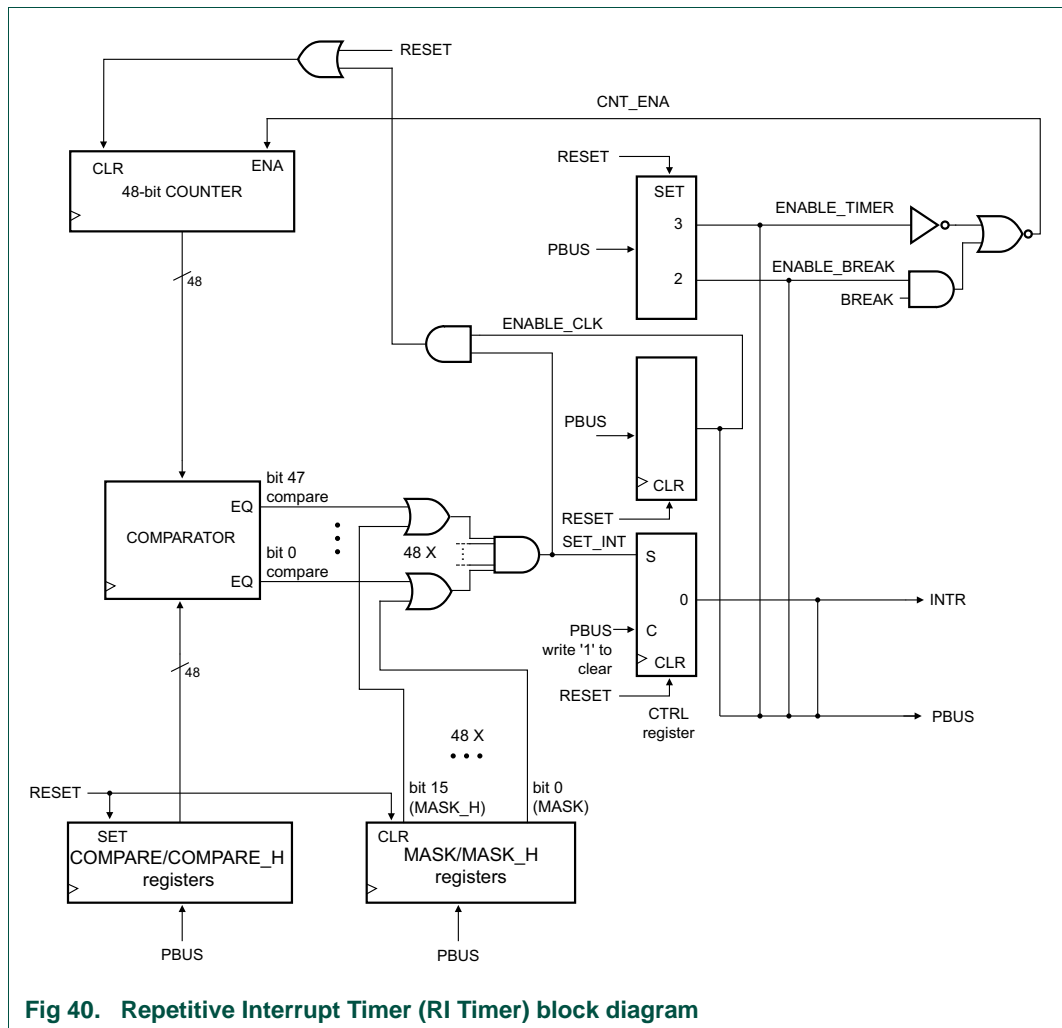


Fig 40. Repetitive Interrupt Timer (RI Timer) block diagram

19.5 Register description

Table 285. Register overview: Repetitive Interrupt Timer (RIT) (base address 0x4007 0000)

| Name | Access | Offset | Description | Reset value ^[1] | Reference |
|-----------|--------|--------|---|----------------------------|---------------------------|
| COMPVAL | R/W | 0x000 | Compare value LSB register. Holds the 32 LSBs of the compare value. | 0xFFFF FFFF | Table 286 |
| MASK | R/W | 0x004 | Mask LSB register. This register holds the 32 LSBs of the mask value. A 1 written to any bit will force the compare to be true for the corresponding bit of the counter and compare register. | 0 | Table 287 |
| CTRL | R/W | 0x008 | Control register. | 0xC | Table 288 |
| COUNTER | R/W | 0x00C | Counter LSB register. 32 LSBs of the counter. | 0 | Table 289 |
| COMPVAL_H | R/W | 0x010 | Compare value MSB register. Holds the 16 MSBs of the compare value. | 0x0000 FFFF | Table 286 |
| MASK_H | R/W | 0x014 | Mask MSB register. This register holds the 16 MSBs of the mask value. A '1' written to any bit will force a compare on the corresponding bit of the counter and compare register. | 0 | Table 287 |
| COUNTER_H | R/W | 0x01C | Counter MSB register. 16 MSBs of the counter. | 0 | Table 289 |

[1] Reset Value reflects the data stored in used bits only. It does not include content of reserved bits.

19.5.1 RI Compare Value LSB register

Table 286. RI Compare Value LSB register (COMPVAL, address 0x4007 0000) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|--|-------------|
| 31:0 | RICOMP | Compare register. Holds the 32 LSBs of the value which is compared to the counter. | 0xFFFF FFFF |

19.5.2 RI Mask LSB register

Table 287. RI Mask LSB register (MASK, address 0x4007 0004) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 31:0 | RIMASK | Mask register. This register holds the 32 LSBs of the mask value. A one written to any bit overrides the result of the comparison for the corresponding bit of the counter and compare register (causes the comparison of the register bits to be always true). | 0 |

19.5.3 RI Control register

Table 288. RI Control register (CTRL, address 0x4007 0008) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|--------|-------|---|-------------|
| 0 | RITINT | | Interrupt flag | 0 |
| | | 1 | This bit is set to 1 by hardware whenever the counter value equals the masked compare value specified by the contents of RICOMPVAL and RIMASK registers. Writing a 1 to this bit will clear it to 0. Writing a 0 has no effect. | |
| | | 0 | The counter value does not equal the masked compare value. | |

Table 288. RI Control register (CTRL, address 0x4007 0008) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|----------|-------|---|-------------|
| 1 | RITENCLR | | Timer enable clear | 0 |
| | | 1 | The timer will be cleared to 0 whenever the counter value equals the masked compare value specified by the contents of COMPVAL/COMPVAL_H and MASK/MASK_H registers. This will occur on the same clock that sets the interrupt flag. | |
| | | 0 | The timer will not be cleared to 0. | |
| 2 | RITENBR | | Timer enable for debug | 1 |
| | | 1 | The timer is halted when the processor is halted for debugging. | |
| | | 0 | Debug has no effect on the timer operation. | |
| 3 | RITEN | | Timer enable. | 1 |
| | | 1 | Timer enabled. Remark: This can be overruled by a debug halt if enabled in bit 2. | |
| | | 0 | Timer disabled. | |
| 31:4 | - | - | Reserved. Read value is undefined, only zero should be written. | NA |

19.5.4 RI Counter LSB register

Table 289. RI Counter register (COUNTER, address 0x4007 000C) bit description

| Bit | Symbol | Description | Reset value |
|------|-----------|---|-------------|
| 31:0 | RICOUNTER | 32 LSBs of the up counter. Counts continuously unless RITEN bit in CTRL register is cleared or debug mode is entered (if enabled by the RITNEBR bit in RICTRL). Can be loaded to any value in software. | 0 |

19.5.5 RI Compare Value MSB register

Table 290. RI Compare Value MSB register (COMPVAL_H, address 0x4007 0010) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 15:0 | RICOMP | Compare value MSB register. Holds the 16 MSBs of the value which is compared to the counter. | 0x0000 FFFF |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | - |

19.5.6 RI Mask MSB register

Table 291. RI Mask MSB register (MASK_H, address 0x4007 0014) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 15:0 | RIMASK | Mask register. This register holds the 16 MSBs of the mask value. A one written to any bit overrides the result of the comparison for the corresponding bit of the counter and compare register (causes the comparison of the register bits to be always true). | 0 |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | - |

19.5.7 RI Counter MSB register

Table 292. RI Counter MSB register (COUNTER_H, address 0x4007 001C) bit description

| Bit | Symbol | Description | Reset value |
|-------|-----------|---|-------------|
| 15:0 | RICOUNTER | 16 LSBs of the up counter. Counts continuously unless RITEN bit in RICTRL register is cleared or debug mode is entered (if enabled by the RITNEBR bit in RICTRL). Can be loaded to any value in software. | 0 |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | - |

19.6 RI timer operation

Following reset, the counter begins counting up from 0. (The RIT bit must be set in the AHBCLKCTRL1 register to enable the clock to the RIT.) Whenever the counter value equals the 48-bit value programmed into the COMPVAL and COMPVAL_H registers, the interrupt flag will be set. Any bit or combination of bits can be removed from this comparison (i.e. forced to compare) by writing a 1 to the corresponding bit(s) in the MASK and MASK_H registers. If the RITENCLR bit is low (default state), a valid comparison ONLY causes the interrupt flag to be set. It has no effect on the count sequence. Counting continues as usual. When the counter reaches 0xFFFF FFFF FFFF it rolls-over to 0 on the next clock and continues counting. If the RITENCLR bit is set to 1 a valid comparison will also cause the counter to be reset to zero. Counting will resume from there on the next clock edge.

Counting can be halted in software by writing a '0' to the RITEN bit. Counting will also be halted when the processor is halted for debugging provided the RITENBR bit is set. Both the RITEN and RITENBR bits are set on reset.

The interrupt flag can be cleared in software by writing a 1 to the RITINT bit.

Software must stop the counter before reloading it with a new value.

The counter (COUNTER/COUNTER_H), COMPVAL/COMPVAL_H registers, MASK/MASK_H registers, and the CTRL register can all be read by software at any time.

20.1 How to read this chapter

The system tick timer (SysTick timer) is present on all LPC5410x devices in both the ARM Cortex-M4 and ARM Cortex-M0+ cores (when the Cortex-M0+ is present on LPC54102 devices).

20.2 Basic configuration

The system tick timer is configured using the following registers:

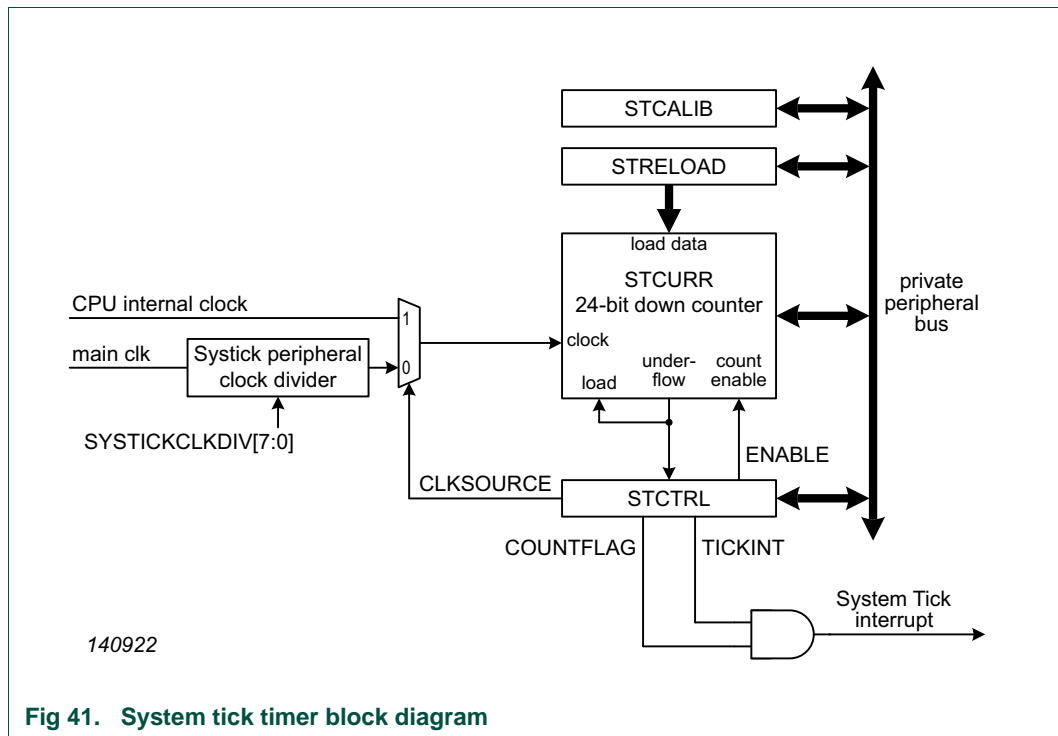
1. Pins: The system tick timer uses no external pins.
2. Power: The system tick timer is enabled through the SysTick control register). The system tick timer clock is fixed to half the frequency of the system clock.
3. Enable the clock source for the SysTick timer in the SYST_CSR register.

20.3 Features

- Simple 24-bit timer.
- Uses dedicated exception vector.
- Clocked internally by the system clock or the SYSTICKCLK.

20.4 General description

The block diagram of the SysTick timer is shown below in the [Figure 41](#).



The SysTick timer is an integral part of both the Cortex-M4 and Cortex-M0+ (if present). The SysTick timer is intended to generate a fixed 10 millisecond interrupt for use by an operating system or other system management software.

Since the SysTick timer is a part of the CPU, it facilitates porting of software by providing a standard timer that is available on ARM Cortex-based devices. The SysTick timer can be used for:

- An RTOS tick timer which fires at a programmable rate (for example 100 Hz) and invokes a SysTick routine.
- A high-speed alarm timer using the core clock.
- A simple counter. Software can use this to measure time to completion and time used.
- An internal clock source control based on missing/meeting durations. The COUNTFLAG bit-field in the control and status register can be used to determine if an action completed within a set duration, as part of a dynamic clock management control loop.

Refer to the appropriate ARM Cortex User Guide for details.

20.5 Register description

The systick timer registers are located on the private peripheral bus of each CPU (see [Figure 3](#)).

Table 293. Register overview: SysTick timer (base address 0xE000 E000)

| Name | Access | Address offset | Description | Reset value ^[1] | Reference |
|------------|--------|----------------|--|----------------------------|---------------------------|
| SYST_CSR | R/W | 0x010 | System Timer Control and status register | 0 | Table 294 |
| SYST_RVR | R/W | 0x014 | System Timer Reload value register | 0 | Table 295 |
| SYST_CVR | R/W | 0x018 | System Timer Current value register | 0 | Table 296 |
| SYST_CALIB | RO | 0x01C | System Timer Calibration value register | 0 | Table 297 |

[1] Reset Value reflects the data stored in used bits only. It does not include content of reserved bits.

20.5.1 System Timer Control and status register

The SYST_CSR register contains control information for the SysTick timer and provides a status flag. This register is part of the CPU.

This register determines the clock source for the system tick timer.

Table 294. SysTick Timer Control and status register (SYST_CSR - 0xE000 E010) bit description

| Bit | Symbol | Description | Reset value |
|-------|-----------|---|-------------|
| 0 | ENABLE | System Tick counter enable. When 1, the counter is enabled. When 0, the counter is disabled. | 0 |
| 1 | TICKINT | System Tick interrupt enable. When 1, the System Tick interrupt is enabled. When 0, the System Tick interrupt is disabled. When enabled, the interrupt is generated when the System Tick counter counts down to 0. | 0 |
| 2 | CLKSOURCE | System Tick clock source selection. When 1, the system clock is selected. When 0, the output clock from the system tick clock divider (SYSTICKCLKDIV, see Figure 41 and Table 57) is selected as the reference clock. Remark: When the system tick clock divider is selected as the clock source, the CPU clock must be at least 2.5 times faster than the divider output. | 0 |
| 15:3 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 16 | COUNTFLAG | Returns 1 if the SysTick timer counted to 0 since the last read of this register. | 0 |
| 31:17 | - | Reserved. Read value is undefined, only zero should be written. | NA |

20.5.2 System Timer Reload value register

The SYST_RVR register is set to the value that will be loaded into the SysTick timer whenever it counts down to zero. This register is loaded by software as part of timer initialization. The SYST_CALIB register may be read and used as the value for SYST_RVR register if the CPU is running at the frequency intended for use with the SYST_CALIB value.

Table 295. System Timer Reload value register (SYST_RVR - 0xE000 E014) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 23:0 | RELOAD | This is the value that is loaded into the System Tick counter when it counts down to 0. | 0 |
| 31:24 | - | Reserved. Read value is undefined, only zero should be written. | NA |

20.5.3 System Timer Current value register

The SYST_CVR register returns the current count from the System Tick counter when it is read by software.

Table 296. System Timer Current value register (SYST_CVR - 0xE000 E018) bit description

| Bit | Symbol | Description | Reset value |
|-------|---------|---|-------------|
| 23:0 | CURRENT | Reading this register returns the current value of the System Tick counter. Writing any value clears the System Tick counter and the COUNTFLAG bit in STCTRL. | 0 |
| 31:24 | - | Reserved. Read value is undefined, only zero should be written. | NA |

20.5.4 System Timer Calibration value register

The value of the SYST_CALIB register is read-only and is provided by the value of the SYSTCKCAL register in the system configuration block (see [Table 31](#)).

Table 297. System Timer Calibration value register (SYST_CALIB - 0xE000 E01C) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|--|-------------|
| 23:0 | TENMS | | Reload value from the SYSTCKCAL register in the SYSCON block. This field is loaded from the SYSTCKCAL register in Syscon. | 0 |
| 29:24 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 30 | SKEW | | Indicates whether the TENMS value will generate a precise 10 millisecond time, or an approximation. This bit is loaded from the SYSTCKCAL register in Syscon. When 0, the value of TENMS is considered to be precise. When 1, the value of TENMS is not considered to be precise. | 0 |
| 31 | NOREF | | Indicates whether an external reference clock is available. This bit is loaded from the SYSTCKCAL register in Syscon. When 0, a separate reference clock is available. When 1, a separate reference clock is not available. | 0 |

20.6 Functional description

The SysTick timer is a 24-bit timer that counts down to zero and generates an interrupt. The intent is to provide a fixed 10 millisecond time interval between interrupts. The SysTick timer is clocked from the CPU clock (the system clock, see [Figure 3](#)) or from the reference clock, which is fixed to half the frequency of the CPU clock. In order to generate recurring interrupts at a specific interval, the SYST_RVR register must be initialized with the correct value for the desired interval.

20.7 Example timer calculations

To use the system tick timer, do the following:

1. Program the LOAD register with the reload value RELOAD to obtain the desired time interval.
2. Clear the VAL register by writing to it. This ensures that the timer will count from the LOAD value rather than an arbitrary value when the timer is enabled.

The following examples illustrate selecting SysTick timer reload values for different system configurations. All of the examples calculate an interrupt interval of 10 milliseconds, as the SysTick timer is intended to be used, and there are no rounding errors.

System clock = 72 MHz

Program the CTRL register with the value 0x7 which selects the system clock as the clock source and enables the SysTick timer and the SysTick timer interrupt.

$$\text{RELOAD} = (\text{system clock frequency} \times 10 \text{ ms}) - 1 = (72 \text{ MHz} \times 10 \text{ ms}) - 1 = 720000 - 1 = 719999 = 0x000A \text{ FC7F}$$

System tick timer clock = 24 MHz

Program the CTRL register with the value 0x3 which selects the clock from the system tick clock divider (use DIV = 3) as the clock source and enables the SysTick timer and the SysTick timer interrupt.

$$\text{RELOAD} = (\text{system tick timer clock frequency} \times 10 \text{ ms}) - 1 = (24 \text{ MHz} \times 10 \text{ ms}) - 1 = 240000 - 1 = 239999 = 0x0003 \text{ A97F}$$

System clock = 12 MHz

Program the CTRL register with the value 0x7 which selects the system clock as the clock source and enables the SysTick timer and the SysTick timer interrupt.

In this case the system clock is derived from the IRC clock.

$$\text{RELOAD} = (\text{system clock frequency} \times 10 \text{ ms}) - 1 = (12 \text{ MHz} \times 10 \text{ ms}) - 1 = 120000 - 1 = 119999 = 0x0001 \text{ D4BF}$$

21.1 How to read this chapter

The Micro-Tick Timer is available on all LPC5410x devices.

21.2 Features

- Ultra simple timer.
- Write once to start.
- Interrupt or software polling.

21.3 Basic configuration

Configure the Micro-Tick Timer as follows:

- Set the UTICK bit in the AHBCLKCTRL1 register ([Table 52](#)) to enable the clock to the Micro-Tick Timer register interface.
- The Micro-Tick Timer provides an interrupt to the NVIC, connected to slot #9.
- To enable Micro-Tick Timer interrupts for waking up from Deep-sleep and Power-down modes, enable the interrupts in the STARTER0 register ([Table 75](#)) and the NVIC.
- The Micro-Tick Timer has no external pins.
- Enable the Watchdog oscillator that provides the Micro-Tick Timer clock in the syscon block via the Power API. See [Chapter 31](#).

21.4 General description

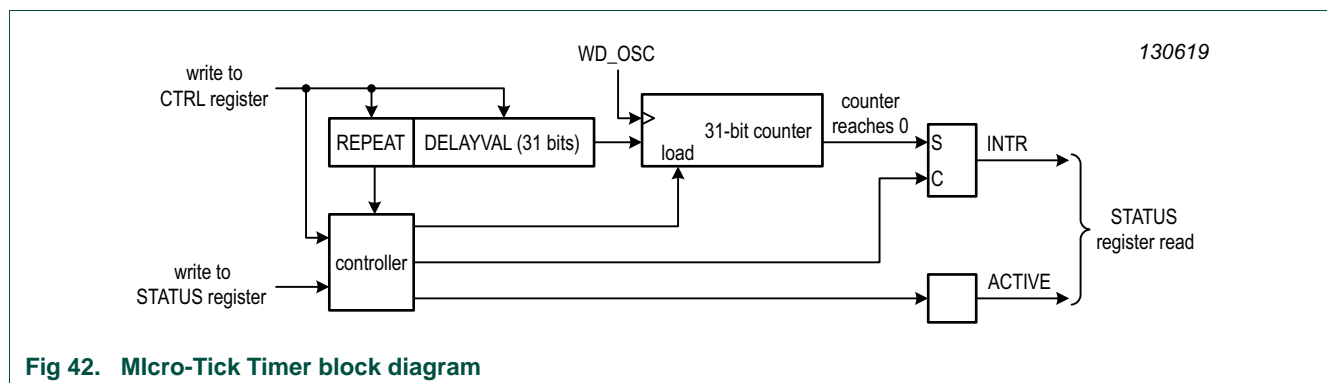


Fig 42. Micro-Tick Timer block diagram

21.5 Register description

The Micro-Tick Timer contains the registers shown in [Table 298](#). Note that the Micro-Tick Timer operates from a different (typically slower) clock than the CPU and bus systems. This means there may be a synchronization delay when accessing Micro-Tick Timer registers.

Table 298. Register overview: Micro-Tick Timer (base address 0x4002 0000)

| Name | Access | Address Offset | Description | Reset value | Reference |
|------|--------|----------------|-------------------|-------------|---------------------------|
| CTRL | R/W | 0x00 | Control register. | 0 | Table 299 |
| STAT | R/W | 0x04 | Status register. | 0 | Table 300 |

21.5.1 CTRL register

This register controls the Micro-tick timer. Any write to the CTRL register resets the counter, meaning a new interval will be measured if one was in progress.

Table 299. Control register (CTRL, address offset 0x00) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|--|-------------|
| 30:0 | DELAYVAL | Tick interval value. The delay will be equal to DELAYVAL + 1 periods of the timer clock. The minimum usable value is 1, for a delay of 2 timer clocks. A value of 0 stops the timer. | 0 |
| 31 | REPEAT | Repeat delay. 0 = One-time delay. 1 = Delay repeats continuously. | 0 |

21.5.2 Status register

This register provides status for the Micro-tick timer.

Table 300. Status register (STAT, address offset 0x04) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 0 | INTR | Interrupt flag. 0 = No interrupt is pending. 1 = An interrupt is pending. A write of any value to this register clears this flag. | 0 |
| 1 | ACTIVE | Active flag. 0 = The Micro-Tick Timer is stopped. 1 = The Micro-Tick Timer is currently active. | 0 |
| 31:2 | - | Reserved | - |

22.1 How to read this chapter

Read this chapter for a description of the USART peripheral and the software interface.

22.2 Features

- 7, 8, or 9 data bits and 1 or 2 stop bits.
- Synchronous mode with master or slave operation. Includes data phase selection and continuous clock option.
- Multiprocessor/multidrop (9-bit) mode with software address compare.
- RS-485 transceiver output enable.
- Parity generation and checking: odd, even, or none.
- Software selectable oversampling from 5 to 16 clocks in asynchronous mode.
- One transmit and one receive data buffer.
- FIFO support from the System FIFO, see [Chapter 25](#) for details.
- RTS/CTS for hardware signaling for automatic flow control. Software flow control can be performed using Delta CTS detect, Transmit Disable control, and any GPIO as an RTS output.
- Received data and status can optionally be read from a single register.
- Break generation and detection.
- Receive data is 2 of 3 sample "voting". Status flag set when one sample differs.
- Built-in Baud Rate Generator with auto-baud function.
- A fractional rate divider is shared among all USARTs.
- Interrupts available for Receiver Ready, Transmitter Ready, Receiver Idle, change in receiver break detect, Framing error, Parity error, Overrun, Underrun, Delta CTS detect, and receiver sample noise detected.
- Loopback mode for testing of data and flow control.
- USART transmit and receive functions can operated with the system DMA controller.
- Special operating mode allows operation at up to 9600 baud using the 32 kHz RTC oscillator as the USART clock. This mode can be used while the device is in Power-down mode and can wake-up the device when a character is received.
- USART transmit and receive functions can be operated with the system DMA controller.

22.3 Basic configuration

If using the USARTs with FIFO support, configure the FIFOs, see [Chapter 25](#).

Configure USARTs for receiving and transmitting data:

- In the ASYNCAPBCLKCTRL register, set bit 1 to 4 ([Table 92](#)) to enable the clock to the register interface.
- Clear the USART0/1/2/3 peripheral resets using the ASYNCPRESETCTRL register ([Table 89](#)).
- Enable or disable the USART0/1/2/3 interrupts in slots #17 to 20 in the NVIC.
- Configure the USART0/1/2/3 pin functions via IOCON, see [Chapter 8](#).
- Configure the USART clock and baud rate. See [Section 22.3.1](#).
- Send and receive lines are connected to DMA request lines. See [Table 173](#).

Configure the USART0/1/2/3 to wake up the part from low power modes:

- Configure the USART to receive and transmit data in synchronous slave mode. See [Section 22.3.2](#).

22.3.1 Configure the USART clock and baud rate

The USARTs share a common base clock, which can be adjusted by a fractional divider (also see [Section 22.7.1.4 “32 kHz mode”](#)). The peripheral clock and the fractional divider for the baud rate calculation are set up in the SYSCON block as follows (see [Figure 43](#)):

1. The Asynchronous APB clock must be configured via the ASYNCCLKDIV register if it has not already been set up. See [Section 4.5.54](#) through [Section 4.5.59](#).
2. If a fractional value is needed to obtain a particular baud rate, program the fractional rate divider (FRG, controlled by Syscon register FRGCTRL). The fractional divider value is the fraction of MULT/DIV. The MULT and DIV values are programmed in the FRGCTRL register. The DIV value must be programmed with the fixed value of 256.

$$\text{USART clock} = \text{ASYNCCLKDIV} / (1 + (\text{MULT} / \text{DIV}))$$

The following rules apply for MULT and DIV:

- Always set DIV to 256 by programming the FRGCTRL register with the value of 0xFF.
- Set the MULT to any value between 0 and 255.

See [Table 98](#) for more information on the FRG.

3. In asynchronous mode: For each USART, configure the baud rate divider BRGVAL in that USART's BRG register. The baud rate divider divides the common USART peripheral clock to create the clock needed to produce the desired baud rate.

$$\text{baud rate} = \text{U_PCLK} / \text{oversample rate} \times (\text{BRGVAL} + 1)$$

$$\text{baud rate} = \text{U_PCLK} / (\text{OSRVAL} + 1) \times (\text{BRGVAL} + 1)$$

$$\text{BRGVAL} = (\text{U_PCLK} / \text{OSRVAL} + 1) \times \text{baud rate} - 1$$

(assumes U_PCLK ≥ oversample rate x baud rate)

See [Section 22.6.9 “USART Baud Rate Generator register”](#).

- In synchronous master mode: The serial clock is $U_n_SCLK = U_PCLK / (BRGVAL+1)$.

The USART can also be clocked by the 32 kHz RTC oscillator. Set the MODE32K bit to enable this 32 kHz mode. See also [Section 22.7.1.4 “32 kHz mode”](#).

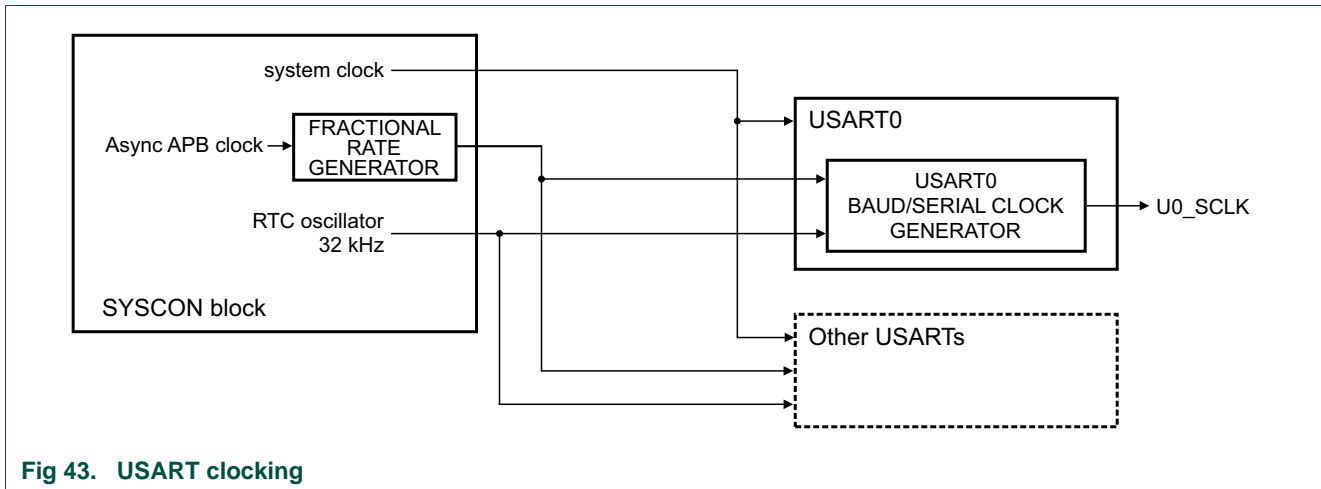


Fig 43. USART clocking

For details on the clock configuration see:

[Section 22.7.1 “Clocking and baud rates”](#)

22.3.2 Configure the USART for wake-up

A USART can wake up the system from sleep mode in asynchronous or synchronous mode on any enabled USART interrupt.

In Deep-sleep or power-down mode, there are two options for configuring USART for wake-up:

- If the USART is configured for synchronous slave mode, the USART block can create an interrupt on a received signal even when the USART block receives no on-chip clocks - that is in Deep-sleep or Power-down mode.

As long as the USART receives a clock signal from the master, it can receive up to one byte in the RXDAT register while in Deep-sleep or Power-down mode. Any interrupt raised as part of the receive data process can then wake up the part.

- If the 32 kHz mode is enabled, the USART can run in asynchronous mode using the 32 kHz RTC oscillator and create interrupts.

22.3.2.1 Wake-up from Sleep mode

- Configure the USART in either asynchronous mode or synchronous mode. See [Table 304](#).
- Enable the USART interrupt in the NVIC.
- Any USART interrupt wakes up the part from sleep mode. Enable the USART interrupt in the INTENSET register ([Table 307](#)).

22.3.2.2 Wake-up from Deep-sleep or Power-down mode

- Configure the USART in synchronous slave mode. See [Table 304](#). The SCLK function must be connected to a pin and also connect the pin to the master. Alternatively, the 32 kHz mode can be enabled and the USART operated in asynchronous mode with the 32 kHz RTC oscillator.
- Enable the USART interrupt in the STARTER1 register. See [Table 75 “Start enable register 0 \(STARTER0, address 0x4000 0240\) bit description”](#).
- Enable the USART interrupt in the NVIC.
- The USART wakes up the part from Deep-sleep or Power-down mode on all events that cause an interrupt and are enabled in the INTENSET register. Typical wake-up events are:
 - A start bit has been received.
 - The RXDAT buffer has received a byte.
 - In synchronous mode, data is ready to be transmitted in the TXDAT buffer and a serial clock from the master has been received.
 - A change in the state of the CTS pin if the CTS function is connected.
 - **Remark:** By enabling or disabling the interrupt in the INTENSET register ([Table 307](#)), you can customize when the wake-up occurs in the USART receive/transmit protocol.

22.4 Pin description

The USART receive, transmit, and control signals are movable functions and are assigned to external pins through via IOCON.

See the IOCON description ([Chapter 8](#)) to assign the USART functions to pins on the device package. Recommended IOCON settings are shown in [Table 302](#).

Table 301. USART pin description

| Function | Direction | Description |
|-----------------------------|-----------|---|
| U0_TXD | O | Transmitter output for USART0. Serial transmit data. |
| U0_RXD | I | Receiver input for USART0. Serial receive data. |
| $\overline{\text{U0_RTS}}$ | O | Request To Send output for USART0. This signal supports inter-processor communication through the use of hardware flow control. This signal can also be configured to act as an output enable for an external RS-485 transceiver. RTS is active when the USART RTS signal is configured to appear on a device pin. |
| $\overline{\text{U0_CTS}}$ | I | Clear To Send input for USART0. Active low signal indicates that the external device that is in communication with the USART is ready to accept data. This feature is active when enabled by the CTSEn bit in CFG register and when configured to appear on a device pin. When deasserted (high) by the external device, the USART will complete transmitting any character already in progress, then stop until CTS is again asserted (low). |
| U0_SCLK | I/O | Serial clock input/output for USART0 in synchronous mode. Clock input or output in synchronous mode. |
| U1_TXD | O | Transmitter output for USART1. Serial transmit data. |
| U1_RXD | I | Receiver input for USART1. |
| $\overline{\text{U1_RTS}}$ | O | Request To Send output for USART1. |
| $\overline{\text{U1_CTS}}$ | I | Clear To Send input for USART1. |
| U1_SCLK | I/O | Serial clock input/output for USART1 in synchronous mode. |
| U2_TXD | O | Transmitter output for USART2. Serial transmit data. |
| U2_RXD | I | Receiver input for USART2. |
| $\overline{\text{U2_RTS}}$ | O | Request To Send output for USART2. |
| $\overline{\text{U2_CTS}}$ | I | Clear To Send input for USART2. |
| U2_SCLK | I/O | Serial clock input/output for USART2 in synchronous mode. |
| U3_TXD | O | Transmitter output for USART3. Serial transmit data. |
| U3_RXD | I | Receiver input for USART3. |
| U3_SCLK | I/O | Serial clock input/output for USART3 in synchronous mode. |

Table 302: Suggested USART pin settings

| IOCON bit(s) | Type D pin | Type A pin | Type I pin |
|--------------|---|---------------------|----------------------|
| 10 | OD: Set to 0 unless open-drain output is desired. | Same as type D. | I2CFILTER: Set to 1. |
| 9 | SLEW: Generally set to 0. | Not used, set to 0. | I2CDRIVE: Set to 0. |
| 8 | FILTEROFF: Generally set to 1. | Same as type D. | Same as type D. |
| 7 | DIGIMODE: Set to 1. | Same as type D. | Same as type D. |
| 6 | INVERT: Set to 0. | Same as type D. | Same as type D. |
| 5 | Not used, set to 0. | Not used, set to 0. | I2CSLEW: Set to 1. |

Table 302: Suggested USART pin settings

| IOCON bit(s) | Type D pin | Type A pin | Type I pin |
|-----------------|---|--|---|
| 4:3 | MODE: Set 00 (to pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input). | Same as type D. | Not used, set to 0. |
| 2:0 | FUNC: Must select the correct function for this peripheral. | Same as type D. | Same as type D. |
| General comment | A good choice for USART input or output. | A reasonable choice for USART input or output. | Not recommended for USART functions that can be outputs in the chosen mode. |

22.5 General description

The USART receiver block monitors the serial input line, Un_RXD, for valid input. The receiver shift register assembles characters as they are received, after which they are passed to the receiver buffer register to await access by the CPU or DMA controller. In IrDA mode, an incoming pulse is converted to a standard USART signal. When RTS is configured as an RS-485 output enable, it is asserted at the beginning of an transmitted character, and deasserted either at the end of the character, or after a one character delay (selected by software).

The USART transmitter block accepts data written by the CPU or DMA controller and buffers the data in the transmit holding register. When the transmitter is available, the transmit shift register takes that data, formats it, and serializes it to the serial output, Un_TXD. In IrDA mode, the outgoing USART signal is converted to IrDA pulses.

The Baud Rate Generator block divides the incoming clock to create an oversample clock (typically 16x) in the standard asynchronous operating mode. The BRG clock input source is the shared Fractional Rate Generator that runs from the APB clock (PCLK). The 32 kHz operating mode generates a specially timed internal clock based on the RTC oscillator frequency.

In synchronous slave mode, data is transmitted and received using the serial clock directly. In synchronous master mode, data is transmitted and received using the baud rate clock without division.

Status information from the transmitter and receiver is saved and provided via the STAT register. Many of the status flags are able to generate interrupts, as selected by software. The INTSTAT register provides a view of all interrupts that are both enabled and pending.

Remark: The fractional value and the USART peripheral clock are shared between all USARTs.

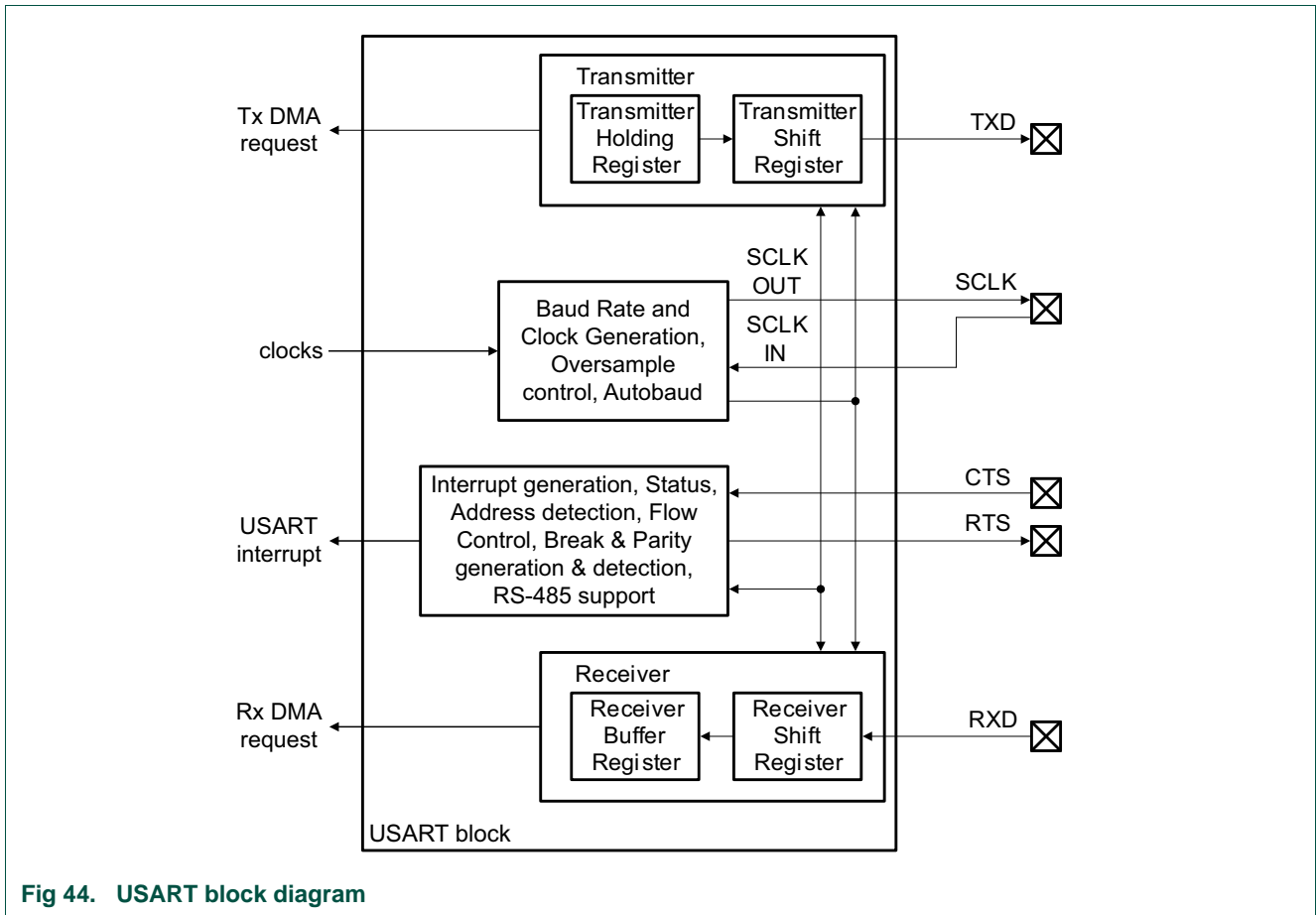


Fig 44. USART block diagram

22.6 Register description

The reset value reflects the data stored in used bits only. It does not include the content of reserved bits.

Table 303: Register overview: USART (base address 0x4008 4000 (USART0), 0x4008 8000 (USART1), 0x4008 C000 (USART2), 0x0x4009 0000 (USART3))

| Name | Access | Offset | Description | Reset value | Reference |
|-----------|--------|--------|---|-------------|---------------------------|
| CFG | R/W | 0x00 | USART Configuration register. Basic USART configuration settings that typically are not changed during operation. | 0 | Table 304 |
| CTL | R/W | 0x04 | USART Control register. USART control settings that are more likely to change during operation. | 0 | Table 305 |
| STAT | R/W | 0x08 | USART Status register. The complete status value can be read here. Writing ones clears some bits in the register. Some bits can be cleared by writing a 1 to them. | 0x0E | Table 306 |
| INTENSET | R/W | 0x0C | Interrupt Enable read and Set register. Contains an individual interrupt enable bit for each potential USART interrupt. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set. | 0 | Table 307 |
| INTENCLR | WO | 0x10 | Interrupt Enable Clear register. Allows clearing any combination of bits in the INTENSET register. Writing a 1 to any implemented bit position causes the corresponding bit to be cleared. | - | Table 308 |
| RXDAT | RO | 0x14 | Receiver Data register. Contains the last character received. | - | Table 309 |
| RXDATSTAT | RO | 0x18 | Receiver Data with Status register. Combines the last character received with the current USART receive status. Allows DMA or software to recover incoming data and status together. | - | Table 310 |
| TXDAT | R/W | 0x1C | Transmit Data register. Data to be transmitted is written here. | 0 | Table 311 |
| BRG | R/W | 0x20 | Baud Rate Generator register. 16-bit integer baud rate divisor value. | 0 | Table 312 |
| INTSTAT | RO | 0x24 | Interrupt status register. Reflects interrupts that are currently enabled. | 0x05 | Table 313 |
| OSR | R/W | 0x28 | Oversample selection register for asynchronous communication. | 0xF | Table 314 |
| ADDR | R/W | 0x2C | Address register for automatic address matching. | 0 | Table 315 |

22.6.1 USART Configuration register

The CFG register contains communication and mode settings for aspects of the USART that would normally be configured once in an application.

Remark: Only the CFG register can be written when the ENABLE bit = 0. CFG can be set up by software with ENABLE = 1, then the rest of the USART can be configured.

Remark: If software needs to change configuration values, the following sequence should be used: 1) Make sure the USART is not currently sending or receiving data. 2) Disable the USART by writing a 0 to the Enable bit (0 may be written to the entire register). 3) Write the new configuration value, with the ENABLE bit set to 1.

Table 304. USART Configuration register (CFG, offset 0x00) bit description

| Bit | Symbol | Value | Description | Reset Value |
|-----|-----------|-------|---|-------------|
| 0 | ENABLE | | USART Enable. | 0 |
| | | 0 | Disabled. The USART is disabled and the internal state machine and counters are reset. While Enable = 0, all USART interrupts and DMA transfers are disabled. When Enable is set again, CFG and most other control bits remain unchanged. For instance, when re-enabled, the USART will immediately generate a TxRdy interrupt (if enabled in the INTENSET register) or a DMA transfer request because the transmitter has been reset and is therefore available. | |
| | | 1 | Enabled. The USART is enabled for operation. | |
| 1 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 3:2 | DATALEN | | Selects the data size for the USART. | 00 |
| | | 0x0 | 7 bit Data length. | |
| | | 0x1 | 8 bit Data length. | |
| | | 0x2 | 9 bit data length. The 9th bit is commonly used for addressing in multidrop mode. See the ADDRDET bit in the CTL register. | |
| | | 0x3 | Reserved. | |
| 5:4 | PARITYSEL | | Selects what type of parity is used by the USART. | 00 |
| | | 0x0 | No parity. | |
| | | 0x1 | Reserved. | |
| | | 0x2 | Even parity. Adds a bit to each character such that the number of 1s in a transmitted character is even, and the number of 1s in a received character is expected to be even. | |
| | | 0x3 | Odd parity. Adds a bit to each character such that the number of 1s in a transmitted character is odd, and the number of 1s in a received character is expected to be odd. | |
| 6 | STOPLEN | | Number of stop bits appended to transmitted data. Only a single stop bit is required for received data. | 0 |
| | | 0 | 1 stop bit. | |
| | | 1 | 2 stop bits. This setting should only be used for asynchronous communication. | |
| 7 | MODE32K | | Selects standard or 32 kHz clocking mode. | 0 |
| | | 0 | Disabled. USART uses standard clocking. | |
| | | 1 | Enabled. USART uses the 32 kHz clock from the RTC oscillator as the clock source to the BRG, and uses a special bit clocking scheme. | |

Table 304. USART Configuration register (CFG, offset 0x00) bit description ...continued

| Bit | Symbol | Value | Description | Reset Value |
|-----|----------|-------|---|-------------|
| 8 | LINMODE | | LIN break mode enable. | 0 |
| | | 0 | Disabled. Break detect and generate is configured for normal operation. | |
| | | 1 | Enabled. Break detect and generate is configured for LIN bus operation. | |
| 9 | CTSEN | | CTS Enable. Determines whether CTS is used for flow control. CTS can be from the input pin, or from the USART's own RTS if loopback mode is enabled. | 0 |
| | | 0 | No flow control. The transmitter does not receive any automatic flow control signal. | |
| | | 1 | Flow control enabled. The transmitter uses the CTS input (or RTS output in loopback mode) for flow control purposes. | |
| 10 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 11 | SYNCEN | | Selects synchronous or asynchronous operation. | 0 |
| | | 0 | Asynchronous mode. | |
| | | 1 | Synchronous mode. | |
| 12 | CLKPOL | | Selects the clock polarity and sampling edge of received data in synchronous mode. | 0 |
| | | 0 | Falling edge. Un_RXD is sampled on the falling edge of SCLK. | |
| | | 1 | Rising edge. Un_RXD is sampled on the rising edge of SCLK. | |
| 13 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 14 | SYNCMST | | Synchronous mode Master select. | 0 |
| | | 0 | Slave. When synchronous mode is enabled, the USART is a slave. | |
| | | 1 | Master. When synchronous mode is enabled, the USART is a master. | |
| 15 | LOOP | | Selects data loopback mode. | 0 |
| | | 0 | Normal operation. | |
| | | 1 | Loopback mode. This provides a mechanism to perform diagnostic loopback testing for USART data. Serial data from the transmitter (Un_TXD) is connected internally to serial input of the receive (Un_RXD). Un_TXD and Un_RTS activity will also appear on external pins if these functions are configured to appear on device pins. The receiver RTS signal is also looped back to CTS and performs flow control if enabled by CTSEN. | |
| 16 | IOMODE | | I/O output mode. | 0 |
| | | 0 | Standard. USART output and input operate in standard fashion. | |
| | | 1 | IrDA. USART output and input operate in IrDA mode. | |
| 17 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 18 | OETA | | Output Enable Turnaround time enable for RS-485 operation. | 0 |
| | | 0 | Disabled. If selected by OESEL, the Output Enable signal deasserted at the end of the last stop bit of a transmission. | |
| | | 1 | Enabled. If selected by OESEL, the Output Enable signal remains asserted for one character time after the end of the last stop bit of a transmission. OE will also remain asserted if another transmit begins before it is deasserted. | |
| 19 | AUTOADDR | | Automatic Address matching enable. | 0 |
| | | 0 | Disabled. When addressing is enabled by ADDRDET, address matching is done by software. This provides the possibility of versatile addressing (e.g. respond to more than one address). | |
| | | 1 | Enabled. When addressing is enabled by ADDRDET, address matching is done by hardware, using the value in the ADDR register as the address to match. | |

Table 304. USART Configuration register (CFG, offset 0x00) bit description ...continued

| Bit | Symbol | Value | Description | Reset Value |
|-------|--------|-------|--|-------------|
| 20 | OESEL | | Output Enable Select. | 0 |
| | | 0 | Standard. The RTS signal is used as the standard flow control function. | |
| | | 1 | RS-485. The RTS signal configured to provide an output enable signal to control an RS-485 transceiver. | |
| 21 | OEPOL | | Output Enable Polarity. | 0 |
| | | 0 | Low. If selected by OESEL, the output enable is active low. | |
| | | 1 | High. If selected by OESEL, the output enable is active high. | |
| 22 | RXPOL | | Receive data polarity. | 0 |
| | | 0 | Standard. The RX signal is used as it arrives from the pin. This means that the RX rest value is 1, start bit is 0, data is not inverted, and the stop bit is 1. | |
| | | 1 | Inverted. The RX signal is inverted before being used by the USART. This means that the RX rest value is 0, start bit is 1, data is inverted, and the stop bit is 0. | |
| 23 | TXPOL | | Transmit data polarity. | 0 |
| | | 0 | Standard. The TX signal is sent out without change. This means that the TX rest value is 1, start bit is 0, data is not inverted, and the stop bit is 1. | |
| | | 1 | Inverted. The TX signal is inverted by the USART before being sent out. This means that the TX rest value is 0, start bit is 1, data is inverted, and the stop bit is 0. | |
| 31:24 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

22.6.2 USART Control register

The CTL register controls aspects of USART operation that are more likely to change during operation.

Table 305. USART Control register (CTL, offset 0x04) bit description

| Bit | Symbol | Value | Description | Reset Value |
|-------|-----------|-------|---|-------------|
| 0 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 1 | TXBRKEN | 0 | Normal operation. | 0 |
| | | 1 | Continuous break. Continuous break is sent immediately when this bit is set, and remains until this bit is cleared. A break may be sent without danger of corrupting any currently transmitting character if the transmitter is first disabled (TXDIS in CTL is set) and then waiting for the transmitter to be disabled (TXDISINT in STAT = 1) before writing 1 to TXBRKEN. | |
| 2 | ADDRDET | 0 | Disabled. The USART presents all incoming data. | 0 |
| | | 1 | Enabled. The USART receiver ignores incoming data that does not have the most significant bit of the data (typically the 9th bit) = 1. When the data MSB bit = 1, the receiver treats the incoming data normally, generating a received data interrupt. Software can then check the data to see if this is an address that should be handled. If it is, the ADDRDET bit is cleared by software and further incoming data is handled normally. | |
| 5:3 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 6 | TXDIS | 0 | Not disabled. USART transmitter is not disabled. | 0 |
| | | 1 | Disabled. USART transmitter is disabled after any character currently being transmitted is complete. This feature can be used to facilitate software flow control. | |
| 7 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 8 | CC | 0 | Clock on character. In synchronous mode, SCLK cycles only when characters are being sent on Un_TXD or to complete a character that is being received. | 0 |
| | | 1 | Continuous clock. SCLK runs continuously in synchronous mode, allowing characters to be received on Un_RxD independently from transmission on Un_TXD). | |
| 9 | CLRCCONRX | 0 | No effect. No effect on the CC bit. | 0 |
| | | 1 | Auto-clear. The CC bit is automatically cleared when a complete character has been received. This bit is cleared at the same time. | |
| 15:10 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 16 | AUTOBAUD | 0 | Disabled. USART is in normal operating mode. | 0 |
| | | 1 | Enabled. USART is in autobaud mode. This bit should only be set when the USART receiver is idle. The first start bit of RX is measured and used to update the BRG register to match the received data rate. AUTOBAUD is cleared once this process is complete, or if there is an AERR. | |
| 31:17 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

22.6.3 USART Status register

The STAT register primarily provides a complete set of USART status flags for software to read. Flags other than read-only flags may be cleared by writing ones to corresponding bits of STAT. Interrupt status flags that are read-only and cannot be cleared by software, can be masked using the INTENCLR register (see [Table 308](#)).

The error flags for received noise, parity error, framing error, and overrun are set immediately upon detection and remain set until cleared by software action in STAT.

Table 306. USART Status register (STAT, offset 0x08) bit description

| Bit | Symbol | Description | Reset value | Access [1] |
|-----|------------|---|-------------|----------------------------|
| 0 | RXRDY | Receiver Ready flag. When 1, indicates that data is available to be read from the receiver buffer. Cleared after a read of the RXDAT or RXDATSTAT registers. | 0 | RO |
| 1 | RXIDLE | Receiver Idle. When 0, indicates that the receiver is currently in the process of receiving data. When 1, indicates that the receiver is not currently in the process of receiving data. | 1 | RO |
| 2 | TXRDY | Transmitter Ready flag. When 1, this bit indicates that data may be written to the transmit buffer. Previous data may still be in the process of being transmitted. Cleared when data is written to TXDAT. Set when the data is moved from the transmit buffer to the transmit shift register. | 1 | RO |
| 3 | TXIDLE | Transmitter Idle. When 0, indicates that the transmitter is currently in the process of sending data. When 1, indicate that the transmitter is not currently in the process of sending data. | 1 | RO |
| 4 | CTS | This bit reflects the current state of the CTS signal, regardless of the setting of the CTSEN bit in the CFG register. This will be the value of the CTS input pin unless loopback mode is enabled. | NA | RO |
| 5 | DELTACTS | This bit is set when a change in the state is detected for the CTS flag above. This bit is cleared by software. | 0 | W1 |
| 6 | TXDISSTAT | Transmitter Disabled Status flag. When 1, this bit indicates that the USART transmitter is fully idle after being disabled via the TXDIS bit in the CFG register (TXDIS = 1). | 0 | RO |
| 7 | - | Reserved. Read value is undefined, only zero should be written. | NA | NA |
| 8 | OVERRUNINT | Overrun Error interrupt flag. This flag is set when a new character is received while the receiver buffer is still in use. If this occurs, the newly received character in the shift register is lost. | 0 | W1 |
| 9 | - | Reserved. Read value is undefined, only zero should be written. | NA | NA |
| 10 | RXBRK | Received Break. This bit reflects the current state of the receiver break detection logic. It is set when the Un_RXD pin remains low for 16 bit times. Note that FRAMERRINT will also be set when this condition occurs because the stop bit(s) for the character would be missing. RXBRK is cleared when the Un_RXD pin goes high. | 0 | RO |
| 11 | DELTARXBRK | This bit is set when a change in the state of receiver break detection occurs. Cleared by software. | 0 | W1 |
| 12 | START | This bit is set when a start is detected on the receiver input. Its purpose is primarily to allow wake-up from Deep-sleep or Power-down mode immediately when a start is detected. Cleared by software. | 0 | W1 |
| 13 | FRAMERRINT | Framing Error interrupt flag. This flag is set when a character is received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source. | 0 | W1 |

Table 306. USART Status register (STAT, offset 0x08) bit description

| Bit | Symbol | Description | Reset value | Access [1] |
|-------|--------------|--|-------------|----------------------------|
| 14 | PARITYERRINT | Parity Error interrupt flag. This flag is set when a parity error is detected in a received character. | 0 | W1 |
| 15 | RXNOISEINT | Received Noise interrupt flag. Three samples of received data are taken in order to determine the value of each received data bit, except in synchronous mode. This acts as a noise filter if one sample disagrees. This flag is set when a received data bit contains one disagreeing sample. This could indicate line noise, a baud rate or character format mismatch, or loss of synchronization during data reception. | 0 | W1 |
| 16 | ABERR | Auto baud Error. An auto baud error can occur if the BRG counts to its limit before the end of the start bit that is being measured, essentially an auto baud time-out. | 0 | W1 |
| 31:17 | - | Reserved. Read value is undefined, only zero should be written. | NA | NA |

[1] RO = Read-only, W1 = write 1 to clear.

22.6.4 USART Interrupt Enable read and set register

The INTENSET register is used to enable various USART interrupt sources. Enable bits in INTENSET are mapped in locations that correspond to the flags in the STAT register. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The INTENCLR register is used to clear bits in this register.

Table 307. USART Interrupt Enable read and set register (INTENSET, offset 0x0C) bit description

| Bit | Symbol | Description | Reset Value |
|------|--------------|---|-------------|
| 0 | RXRDYEN | When 1, enables an interrupt when there is a received character available to be read from the RXDAT register. | 0 |
| 1 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 2 | TXRDYEN | When 1, enables an interrupt when the TXDAT register is available to take another character to transmit. | 0 |
| 3 | TXIDLEEN | When 1, enables an interrupt when the transmitter becomes idle (TXIDLE = 1). | 0 |
| 4 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 5 | DELTACTSEN | When 1, enables an interrupt when there is a change in the state of the CTS input. | 0 |
| 6 | TXDISEN | When 1, enables an interrupt when the transmitter is fully disabled as indicated by the TXDISINT flag in STAT. See description of the TXDISINT bit for details. | 0 |
| 7 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 8 | OVERRUNEN | When 1, enables an interrupt when an overrun error occurred. | 0 |
| 10:9 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 11 | DELTARXBRKEN | When 1, enables an interrupt when a change of state has occurred in the detection of a received break condition (break condition asserted or deasserted). | 0 |
| 12 | STARTEN | When 1, enables an interrupt when a received start bit has been detected. | 0 |
| 13 | FRAMERREN | When 1, enables an interrupt when a framing error has been detected. | 0 |
| 14 | PARITYERREN | When 1, enables an interrupt when a parity error has been detected. | 0 |

Table 307. USART Interrupt Enable read and set register (INTENSET, offset 0x0C) bit description ...continued

| Bit | Symbol | Description | Reset Value |
|-------|-----------|---|-------------|
| 15 | RXNOISEEN | When 1, enables an interrupt when noise is detected. See description of the RXNOISEINT bit in Table 306 . | 0 |
| 16 | ABERREN | When 1, enables an interrupt when an auto baud error occurs. | 0 |
| 31:17 | - | Reserved. Read value is undefined, only zero should be written. | NA |

22.6.5 USART Interrupt Enable Clear register

The INTENCLR register is used to clear bits in the INTENSET register.

Table 308. USART Interrupt Enable clear register (INTENCLR, offset 0x10) bit description

| Bit | Symbol | Description | Reset Value |
|-------|---------------|--|-------------|
| 0 | RXRDYCLR | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 1 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 2 | TXRDYCLR | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 3 | TXIDLECLR | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 4 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 5 | DELTACTSCLR | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 6 | TXDISCLR | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 7 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 8 | OVERRUNCLR | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 10:9 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 11 | DELTARXBRKCLR | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 12 | STARTCLR | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 13 | FRAMERRCLR | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 14 | PARITYERRCLR | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 15 | RXNOISECLR | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 16 | ABERRCLR | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 31:17 | - | Reserved. Read value is undefined, only zero should be written. | NA |

22.6.6 USART Receiver Data register

The RXDAT register contains the last character received before any overrun.

Remark: Reading this register changes the status flags in the RXDATSTAT register.

Remark: If the USART is used with FIFO support, do not use this register to receive data, see [Chapter 25](#).

Table 309. USART Receiver Data register (RXDAT, offset 0x14) bit description

| Bit | Symbol | Description | Reset Value |
|------|--------|--|-------------|
| 8:0 | DATA | The USART Receiver Data register contains the next received character. The number of bits that are relevant depends on the USART configuration settings. | 0 |
| 31:9 | - | Reserved, the value read from a reserved bit is not defined. | NA |

22.6.7 USART Receiver Data with Status register

The RXDATSTAT register contains the next complete character to be read and its relevant status flags. This allows getting all information related to a received character with one 16-bit read, which may be especially useful when the DMA is used with the USART receiver.

Remark: Reading this register changes the status flags.

Remark: If the USART is used with FIFO support, do not use this register to receive data, see [Chapter 25](#).

Table 310. USART Receiver Data with Status register (RXDATSTAT, offset 0x18) bit description

| Bit | Symbol | Description | Reset Value |
|-------|-----------|--|-------------|
| 8:0 | RXDATA | The USART Receiver Data register contains the next received character. The number of bits that are relevant depends on the USART configuration settings. | 0 |
| 12:9 | - | Reserved, the value read from a reserved bit is not defined. | NA |
| 13 | FRAMERR | Framing Error status flag. This bit is valid when there is a character to be read in the RXDAT register and reflects the status of that character. This bit will set when the character in RXDAT was received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source. | 0 |
| 14 | PARITYERR | Parity Error status flag. This bit is valid when there is a character to be read in the RXDAT register and reflects the status of that character. This bit will be set when a parity error is detected in a received character. | 0 |
| 15 | RXNOISE | Received Noise flag. See description of the RxNoiseInt bit in Table 306 . | 0 |
| 31:16 | - | Reserved, the value read from a reserved bit is not defined. | NA |

22.6.8 USART Transmitter Data Register

The TXDAT register is written in order to send data via the USART transmitter. That data will be transferred to the transmit shift register when it is available, and another character may then be written to TXDAT.

Remark: If the USART is used with FIFO support, do not use this register to transmit data, see [Chapter 25](#).

Table 311. USART Transmitter Data Register (TXDAT, offset 0x1C) bit description

| Bit | Symbol | Description | Reset Value |
|------|--------|--|-------------|
| 8:0 | TXDATA | Writing to the USART Transmit Data Register causes the data to be transmitted as soon as the transmit shift register is available and any conditions for transmitting data are met: CTS low (if CTSEN bit = 1), TXDIS bit = 0. | 0 |
| 31:9 | - | Reserved. Only zero should be written. | NA |

22.6.9 USART Baud Rate Generator register

The Baud Rate Generator is a simple 16-bit integer divider controlled by the BRG register. The BRG register contains the value used to divide the base clock in order to produce the clock used for USART internal operations.

A 16-bit value allows producing standard baud rates from 300 baud and lower at the highest frequency of the device, up to 921,600 baud from a base clock as low as 14.7456 MHz.

Typically, the baud rate clock is 16 times the actual baud rate. This overclocking allows for centering the data sampling time within a bit cell, and for noise reduction and detection by taking three samples of incoming data.

Note that in 32 kHz mode, the baud rate generator is still used and must be set to 0 if 9600 baud is required.

For more information on USART clocking, see [Section 22.7.1](#) and [Section 22.3.1](#).

Remark: In order to change a baud rate after a USART is running, the following sequence should be used:

1. Make sure the USART is not currently sending or receiving data.
2. Disable the USART by writing a 0 to the Enable bit (0 may be written to the entire register).
3. Write the new BRGVAL.
4. Write to the CFG register to set the Enable bit to 1.

Table 312. USART Baud Rate Generator register (BRG, offset 0x20) bit description

| Bit | Symbol | Description | Reset Value |
|-------|--------|--|-------------|
| 15:0 | BRGVAL | This value is used to divide the USART input clock to determine the baud rate, based on the input clock from the FRG. 0 = The FRG clock is used directly by the USART function. 1 = The FRG clock is divided by 2 before use by the USART function. 2 = The FRG clock is divided by 3 before use by the USART function. ... 0xFFFF = The FRG clock is divided by 65,536 before use by the USART function. | 0 |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | NA |

22.6.10 USART Interrupt Status register

The read-only INTSTAT register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See [Table 306](#) for detailed descriptions of the interrupt flags.

Table 313. USART Interrupt Status register (INTSTAT, offset 0x24) bit description

| Bit | Symbol | Description | Reset Value |
|-------|--------------|--|-------------|
| 0 | RXRDY | Receiver Ready flag. | 0 |
| 1 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 2 | TXRDY | Transmitter Ready flag. | 1 |
| 3 | TXIDLE | Transmitter Idle status. | 0 |
| 4 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 5 | DELTACTS | This bit is set when a change in the state of the CTS input is detected. | 0 |
| 6 | TXDISINT | Transmitter Disabled Interrupt flag. | 0 |
| 7 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 8 | OVERRUNINT | Overrun Error interrupt flag. | 0 |
| 10:9 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 11 | DELTARXBRK | This bit is set when a change in the state of receiver break detection occurs. | 0 |
| 12 | START | This bit is set when a start is detected on the receiver input. | 0 |
| 13 | FRAMERRINT | Framing Error interrupt flag. | 0 |
| 14 | PARITYERRINT | Parity Error interrupt flag. | 0 |
| 15 | RXNOISEINT | Received Noise interrupt flag. | 0 |
| 16 | ABERRINT | Auto baud Error Interrupt flag. | 0 |
| 31:17 | - | Reserved. Read value is undefined, only zero should be written. | NA |

22.6.11 Oversample selection register

The OSR register allows selection of oversampling in asynchronous modes. The oversample value is the number of BRG clocks used to receive one data bit. The default is industry standard 16x oversampling.

Changing the oversampling can sometimes allow better matching of baud rates in cases where the peripheral clock rate is not a multiple of 16 times the expected maximum baud rate. For all modes where the OSR setting is used, the USART receiver takes three consecutive samples of input data in the approximate middle of the bit time. Smaller values of OSR can make the sampling position within a data bit less accurate and may potentially cause more noise errors or incorrect data.

Table 314. Oversample selection register (OSR, offset 0x28) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|--|-------------|
| 3:0 | OSRVAL | Oversample Selection Value. 0 to 3 = not supported 0x4 = 5 peripheral clocks are used to transmit and receive each data bit. 0x5 = 6 peripheral clocks are used to transmit and receive each data bit. ... 0xF = 16 peripheral clocks are used to transmit and receive each data bit. | 0xF |
| 31:4 | - | Reserved, the value read from a reserved bit is not defined. | NA |

22.6.12 Address register

The ADDR register holds the address for hardware address matching in address detect mode with automatic address matching enabled.

Table 315. Address register (ADDR, offset 0x2C) bit description

| Bit | Symbol | Description | Reset value |
|------|---------|--|-------------|
| 7:0 | ADDRESS | 8-bit address used with automatic address matching. Used when address detection is enabled (ADDRDET in CTL = 1) and automatic address matching is enabled (AUTOADDR in CFG = 1). | 0 |
| 31:8 | - | Reserved, the value read from a reserved bit is not defined. | NA |

22.7 Functional description

22.7.1 Clocking and baud rates

In order to use the USART, clocking details must be defined such as setting up the BRG, and typically also setting up the FRG. See [Figure 43](#).

22.7.1.1 Fractional Rate Generator (FRG)

The Fractional Rate Generator can be used to obtain more precise baud rates when the peripheral clock is not a good multiple of standard (or otherwise desirable) baud rates.

The FRG is typically set up to produce an integer multiple of the highest required baud rate, or a very close approximation. The BRG is then used to obtain the actual baud rate needed.

The FRG register controls the USART Fractional Rate Generator, which provides the base clock for the USART. The Fractional Rate Generator creates a lower rate output clock by suppressing selected input clocks. When not needed, the value of 0 can be set for the FRG, which will then not divide the input clock.

The FRG output clock is defined as the input clock divided by $1 + (\text{MULT} / 256)$, where MULT is in the range of 1 to 255. This allows producing an output clock that ranges from the input clock divided by $1 + 1/256$ to $1 + 255/256$ (just more than 1 to just less than 2). Any further division can be done specific to each USART block by the integer BRG divider contained in each USART.

The base clock produced by the FRG cannot be perfectly symmetrical, so the FRG distributes the output clocks as evenly as is practical. Since the USART normally uses 16x overclocking, the jitter in the fractional rate clock in these cases tends to disappear in the ultimate USART output.

For setting up the fractional divider use the following registers:

[Table 98 “USART fractional baud rate generator register \(FRGCTRL, address 0x4008 0030\) bit description”](#)

For details see [Section 22.3.1 “Configure the USART clock and baud rate”](#).

22.7.1.2 Baud Rate Generator (BRG)

The Baud Rate Generator (see [Section 22.6.9](#)) is used to divide the base clock to produce a rate 16 times the desired baud rate. Typically, standard baud rates can be generated by integer divides of higher baud rates.

22.7.1.3 Baud rate calculations

Base clock rates are 16x for asynchronous mode and 1x for synchronous mode.

22.7.1.4 32 kHz mode

In order to use a 32 kHz clock to operate a USART at any reasonable speed, a number of adaptations need to be made. First, 16x overclocking has to be abandoned. Otherwise, the maximum data rate would be very low. For the same reason, multiple samples of each

data bit must be reduced to one. Finally, special clocking has to be used for individual bit times because 32 kHz is not particularly close to an even multiple of any standard baud rate.

When 32 kHz mode is enabled, clocking comes from the RTC oscillator. The FRG is bypassed, and the BRG can be used to divide down the default 9600 baud to lower rates. Other adaptations required to make the USART work for rates up to 9600 baud are done internally. Rate error will be less than one half percent in this mode, provided the RTC oscillator is operating at the intended frequency of 32.768 kHz.

22.7.2 DMA

A DMA request is provided for each USART direction, and can be used in lieu of interrupts for transferring data by configuring the DMA controller appropriately. The DMA controller provides an acknowledgement signal that clears the related request when it completes handling a that request. The transmitter DMA request is asserted when the transmitter can accept more data. The receiver DMA request is asserted when received data is available to be read.

When DMA is used to perform USART data transfers, other mechanisms can be used to generate interrupts when needed. For instance, completion of the configured DMA transfer can generate an interrupt from the DMA controller. Also, interrupts for special conditions, such as a received break, can still generate useful interrupts.

22.7.3 Synchronous mode

In synchronous mode, a master generates a clock as defined by the clock selection and BRG, which is used to transmit and receive data. As a slave, the external clock is used to transmit and receive data. There is no overclocking in either case.

22.7.4 Flow control

The USART supports both hardware and software flow control.

22.7.4.1 Hardware flow control

The USART supports hardware flow control using RTS and/or CTS signalling. If RTS is configured to appear on a device pin so that it can be sent to an external device, it indicates to an external device the ability of the receiver to receive more data. It can also be used internally to throttle the transmitter from the receiver, which can be especially useful if loopback mode is enabled.

If connected to a pin, and if enabled to do so, the CTS input can allow an external device to throttle the USART transmitter. Both internal and external CTS can be used separately or together.

[Figure 45](#) shows an overview of RTS and CTS within the USART.

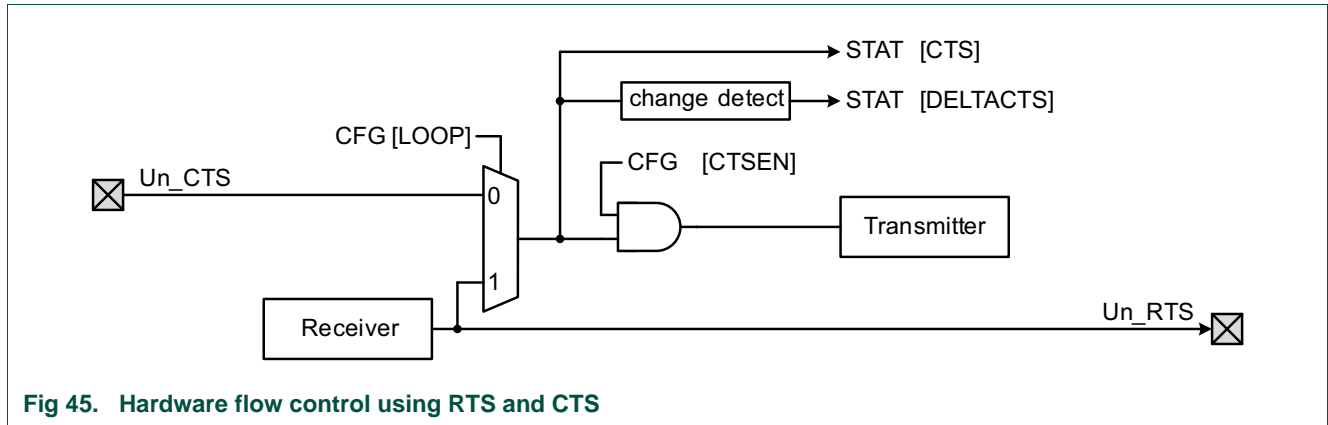


Fig 45. Hardware flow control using RTS and CTS

22.7.4.2 Software flow control

Software flow control could include XON / XOFF flow control, or other mechanisms. these are supported by the ability to check the current state of the CTS input, and/or have an interrupt when CTS changes state (via the CTS and DELTACTS bits, respectively, in the STAT register), and by the ability of software to gracefully turn off the transmitter (via the TXDIS bit in the CTL register).

22.7.5 Autobaud function

The autobaud functions attempts to measure the start bit time of the next received character. For this to work, the measured character must have a 1 in the least significant bit position, so that the start bit is bounded by a falling and rising edge. The measurement is made using the current clocking settings, including the oversampling configuration. The result is that a value is stored in the BRG register that is as close as possible to the correct setting for the sampled character and the current clocking settings. The sampled character is provided in the RXDAT and RXDATSTAT registers, allowing software to double check for the expected character.

Autobaud includes a time-out that is flagged by ABERR if no character is received at the expected time. It is recommended that autobaud only be enabled when the USART receiver is idle. Once enabled, either RXRDY or ABERR will be asserted at some point, at which time software should turn off autobaud.

Autobaud has no meaning, and should not be enabled, if the USART is in synchronous mode.

22.7.6 RS-485 support

RS-485 support requires some form of address recognition and data direction control.

This USART has provisions for hardware address recognition (see the AUTOADDR bit in the CFG register in [Section 22.6.1](#) and the ADDR register in [Section 22.6.12](#)), as well as software address recognition (see the ADDRDET bit in the CTL register in [Section 22.6.2](#)).

Automatic data direction control with the RTS pin can be set up using the OESEL, OEPOL, and OETA bits in the CFG register ([Section 22.6.1](#)). Data direction control can also be implemented in software using a GPIO pin.

22.7.7 Oversampling

Typical industry standard USARTs use a 16x oversample clock to transmit and receive asynchronous data. This is the number of BRG clocks used for one data bit. The Oversample Select Register (OSR) allows this USART to use a 16x down to a 5x oversample clock. There is no oversampling in synchronous modes.

Reducing the oversampling can sometimes help in getting better baud rate matching when the baud rate is very high, or the peripheral clock is very low. For example, the closest actual rate near 115,200 baud with a 12 MHz peripheral clock and 16x oversampling is 107,143 baud, giving a rate error of 7%. Changing the oversampling to 15x gets the actual rate to 114,286 baud, a rate error of 0.8%. Reducing the oversampling to 13x gets the actual rate to 115,385 baud, a rate error of only 0.16%.

There is a cost for altering the oversampling. In asynchronous modes, the USART takes three samples of incoming data on consecutive oversample clocks, as close to the center of a bit time as can be done. When the oversample rate is reduced, the three samples spread out and occupy a larger proportion of a bit time. For example, with 5x oversampling, there is one oversample clock, then three data samples taken, then one more oversample clock before the end of the bit time. Since the oversample clock is running asynchronously from the input data, skew of the input data relative to the expected timing has little room for error. At 16x oversampling, there are several oversample clocks before actual data sampling is done, making the sampling more robust. Generally speaking, it is recommended to use the highest oversampling where the rate error is acceptable in the system.

22.7.8 Break generation and detection

A line break may be sent at any time, regardless of other USART activity. Received break is also detected at any time, including during reception of a character. Received break is signaled when the RX input remains low for 16 bit times. Both the beginning and end of a received break are noted by the DELTARXBRK status flag, which can be used as an interrupt. See [Section 22.7.9](#) for details of LIN mode break.

In order to avoid corrupting any character currently being transmitted, it is recommended that the USART transmitter be disabled by setting the TXDIS bit in the CTL register, then waiting for the TXDISSTAT flag to be set prior to sending a break. Then a 1 may be written to the TXBRKEN bit in the CTL register. This sends a break until TXBRKEN is cleared, allowing any length break to be sent.

22.7.9 LIN bus

The only difference between standard operation and LIN mode is that LIN mode alters the way that break generation and detection is performed (see [Section 22.7.8](#) for details of the standard break). When a break is requested by setting the TXBRKEN bit in the CTL register, then sending a dummy character, a 13 bit time break is sent. A received break is flagged when the RX input remains low for 11 bit times. As for non-LIN mode, a received character is also flagged, and accompanied by a framing error status.

As a LIN slave, the autobaud feature can be used to synchronize to a LIN sync byte, and will return the value of the sync byte as confirmation of success.

Wake-up for LIN can potentially be handled in a number of ways, depending on the system, and what clocks may be running in a slave device. For instance, as long as the USART is receiving internal clocks allowing it to function, it can be set to wake up the CPU for any interrupt, including a received start bit. If there are no clocks running, the GPIO function of the USART RX pin can be programmed to wake up the device.

22.7.10 IrDA communication

IrDA communication can be accomplished with the USART by enabling the IrDA mode using the IOMODE bit in the CFG register ([Section 22.6.1](#)).

23.1 How to read this chapter

SPI0 and SPI1 are available on all parts.

23.2 Features

- Data transmits of 1 to 16 bits supported directly. Larger frames supported by software.
- Master and slave operation.
- Data can be transmitted to a slave without the need to read incoming data. This can be useful while setting up an SPI memory.
- Control information can optionally be written along with data. This allows very versatile operation, including frames of arbitrary length.
- Up to four Slave Select input/outputs with selectable polarity and flexible usage.
- Supports DMA transfers: SPI transmit and receive functions can be operated with the system DMA controller.
- FIFO support from the System FIFO, see [Chapter 25](#) for details.

Remark: Texas Instruments SSI and National Microwire modes are not supported.

23.3 Basic configuration

If using the SPIs with FIFO support, configure the FIFOs, see [Chapter 25](#).

Configure SPI0/1 using the following registers:

- In the ASYNCAPBCLKCTRL register, set bit 9 / 10 ([Table 93](#)) to enable the clock to the register interface.
- Clear the SPI0/1 peripheral resets using the ASYNCPRESETCTRL register ([Table 89](#)).
- Enable/disable the SPI0/1 interrupts in interrupt slots #24 / 25 in the NVIC.
- Configure the SPI0/1 pin functions through IOCON. See [Section 23.4](#).
- The peripheral clock for both SPIs is the asynchronous APB clock (see [Figure 3](#) “Clock generation”).

23.3.1 Configure the SPI for wake-up

In sleep mode, any signal that triggers an SPI interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the SPI clock SPI_PCLK remains active in sleep mode, the SPI can wake up the part independently of whether the SPI block is configured in master or slave mode.

In Deep-sleep or Power-down mode, the SPI clock is turned off as are all peripheral clocks. However, if the SPI is configured in slave mode and an external master on the provides the clock signal, the SPI can create an interrupt asynchronously. This interrupt, if enabled in the NVIC and in the SPI's INTENSET register, can then wake up the core.

23.3.1.1 Wake-up from Sleep mode

- Configure the SPI in either master or slave mode. See [Table 319](#).
- Enable the SPI interrupt in the NVIC.
- Any SPI interrupt wakes up the part from sleep mode. Enable the SPI interrupt in the INTENSET register ([Table 322](#)).

23.3.1.2 Wake-up from Deep-sleep or Power-down mode

- Configure the SPI in slave mode. See [Table 319](#). The SCK function must be connected to a pin and the pin connected to the master.
- Enable the SPI interrupt in the STARTER0 register. See [Table 75 “Start enable register 0 \(STARTER0, address 0x4000 0240\) bit description”](#).
- Enable the SPI interrupt in the NVIC.
- Enable the interrupt in the INTENSET register which configures the interrupt as wake-up event ([Table 322](#)). Examples are the following wake-up events:
 - A change in the state of the SSEL pins.
 - Data available to be received.
 - Receiver overrun.

23.4 Pin description

The SPI signals are movable functions and are assigned to external pins via IOCON. See [Chapter 8](#). Recommended IOCON settings are shown in [Table 317](#).

Table 316: SPI Pin Description

| Function | I/O | Description |
|------------|-----|---|
| SPI0_SCK | I/O | Serial Clock. SCK is a clock signal used to synchronize the transfer of data. It is driven by the master and received by the slave. When the SPI interface is used, the clock is programmable to be active-high or active-low. SCK only switches during a data transfer. It is driven whenever the Master bit in CFG equals 1, regardless of the state of the Enable bit. |
| SPI0_MOSI | I/O | Master Out Slave In. The MOSI signal transfers serial data from the master to the slave. When the SPI is a master, it outputs serial data on this signal. When the SPI is a slave, it clocks in serial data from this signal. MOSI is driven whenever the Master bit in SPInCfg equals 1, regardless of the state of the Enable bit. |
| SPI0_MISO | I/O | Master In Slave Out. The MISO signal transfers serial data from the slave to the master. When the SPI is a master, serial data is input from this signal. When the SPI is a slave, serial data is output to this signal. MISO is driven when the SPI block is enabled, the Master bit in CFG equals 0, and when the slave is selected by one or more SSEL signals. |
| SPI0_SSEL0 | I/O | Slave Select 0. When the SPI interface is a master, it will drive the SSEL signals to an active state before the start of serial data and then release them to an inactive state after the serial data has been sent. By default, this signal is active low but can be selected to operate as active high. When the SPI is a slave, any SSEL in an active state indicates that this slave is being addressed. The SSEL pin is driven whenever the Master bit in the CFG register equals 1, regardless of the state of the Enable bit. |
| SPI0_SSEL1 | I/O | Slave Select 1. |
| SPI0_SSEL2 | I/O | Slave Select 2. |
| SPI0_SSEL3 | I/O | Slave Select 3. |
| SPI1_SCK | I/O | Serial Clock. |
| SPI1_MOSI | I/O | Master Out Slave In. |
| SPI1_MISO | I/O | Master In Slave Out. |
| SPI1_SSEL0 | I/O | Slave Select 0. |
| SPI1_SSEL1 | I/O | Slave Select 1. |
| SPI1_SSEL2 | I/O | Slave Select 2. |
| SPI1_SSEL3 | I/O | Slave Select 3. |

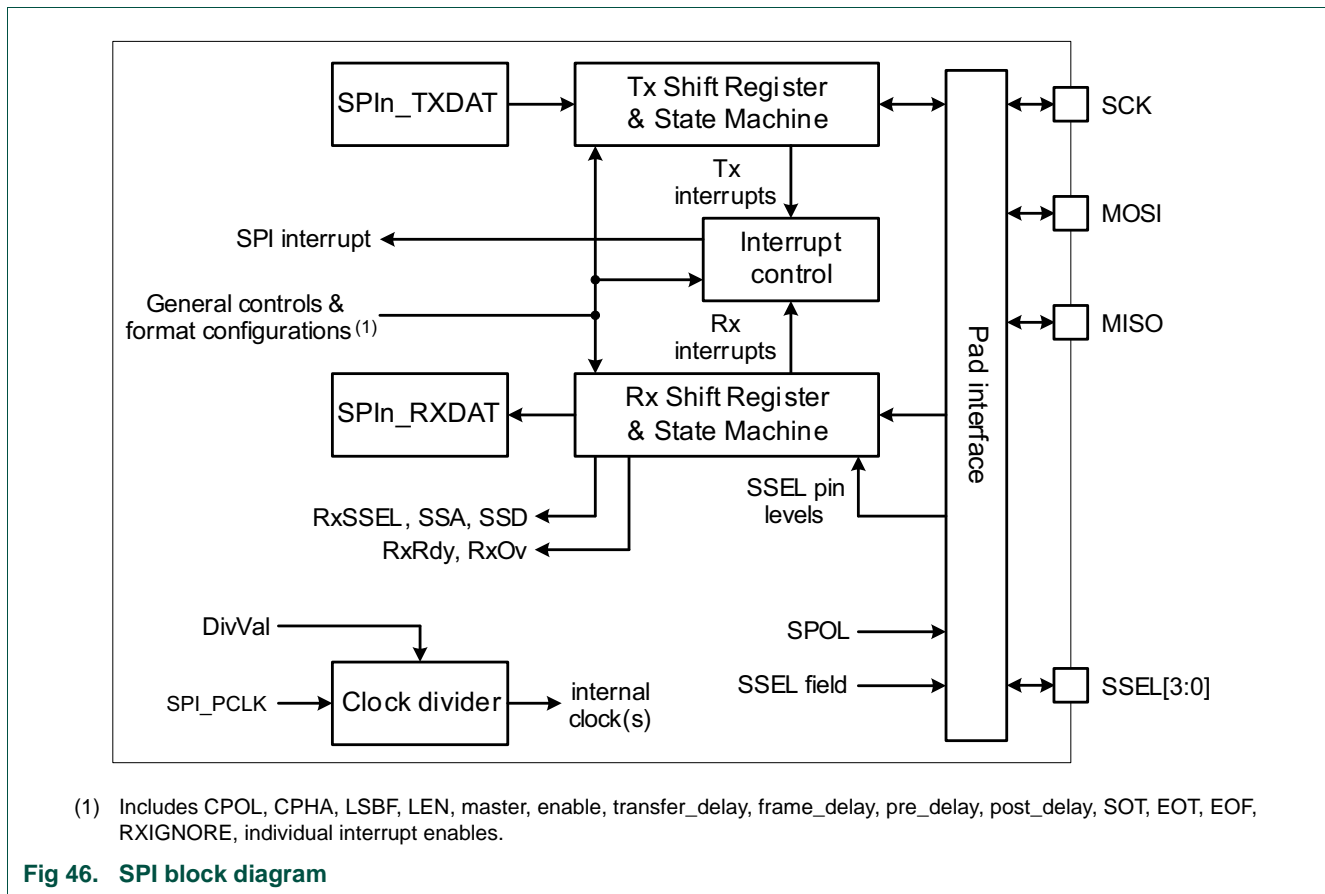
Table 317: Suggested SPI pin settings

| IOCON bit(s) | Type D pin | Type A pin | Type I pin |
|--------------|---|---------------------|----------------------|
| 10 | OD: Set to 0 unless open-drain output is desired. | Same as type D. | I2CFILTER: Set to 1. |
| 9 | SLEW: Generally set to 0. Setting to 1 at higher SPI rates may improve performance. | Not used, set to 0. | I2CDRIVE: Set to 0 |
| 8 | FILTEROFF: Generally set to 1. | Same as type D. | Same as type D. |
| 7 | DIGIMODE: Set to 1. | DIGIMODE: Set to 1. | DIGIMODE: Set to 1. |
| 6 | INVERT: Set to 0. | Same as type D. | Same as type D. |
| 5 | Not used, set to 0. | Same as type D. | I2CSLEW: Set to 1. |

Table 317: Suggested SPI pin settings

| IOCON bit(s) | Type D pin | Type A pin | Type I pin |
|-----------------|---|--|---|
| 4:3 | MODE: Set to 00 (pull-down/pull-up resistor not enabled). Could be another setting if the input might sometimes be floating (causing leakage within the pin input). | Same as type D. | Not used, set to 0. |
| 2:0 | FUNC: Must select the correct function for this peripheral. | Same as type D. | Same as type D. |
| General comment | A good choice for SPI input or output. | A reasonable choice for SPI input or output. | Not recommended for SPI functions that can be outputs in the chosen mode. |

23.5 General description



23.6 Register description

The Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

Table 318. Register overview: SPI (base address 0x400A 4000 (SPI0) and 0x400A 8000 (SPI1))

| Name | Access | Offset | Description | Reset value | Reference |
|----------|--------|--------|--|-------------|---------------------------|
| CFG | R/W | 0x00 | SPI Configuration register | 0 | Table 319 |
| DLY | R/W | 0x04 | SPI Delay register | 0 | Table 320 |
| STAT | R/W | 0x08 | SPI Status. Some status flags can be cleared by writing a 1 to that bit position | 0x0102 | Table 321 |
| INTENSET | R/W | 0x0C | SPI Interrupt Enable read and Set. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set. | 0 | Table 322 |
| INTENCLR | WO | 0x10 | SPI Interrupt Enable Clear. Writing a 1 to any implemented bit position causes the corresponding bit in INTENSET to be cleared. | NA | Table 323 |
| RXDAT | RO | 0x14 | SPI Receive Data | NA | Table 324 |
| TXDATCTL | R/W | 0x18 | SPI Transmit Data with Control | 0 | Table 325 |
| TXDAT | R/W | 0x1C | SPI Transmit Data | 0 | Table 326 |
| TXCTL | R/W | 0x20 | SPI Transmit Control | 0 | Table 327 |
| DIV | R/W | 0x24 | SPI clock Divider | 0 | Table 328 |
| INTSTAT | RO | 0x28 | SPI Interrupt Status | 0 | Table 329 |

23.6.1 SPI Configuration register

The CFG register contains information for the general configuration of the SPI. Typically, this information is not changed during operation. See the description of the master idle status (MSTIDLE in [Table 321](#)) for more information.

Remark: If changes need to be made to settings in the CFG register after the interface has been in use, the interface should first be disabled by clearing the ENABLE bit once the interface is fully idle. See the description of the master idle status (MSTIDLE in [Table 321](#)) for more information. Then the new configuration may be written to CFG. Finally, set the ENABLE bit.

Table 319. SPI Configuration register (CFG, offset 0x00) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|--------|-------|--|-------------|
| 0 | ENABLE | | SPI enable. | 0 |
| | | 0 | Disabled. The SPI is disabled and the internal state machine and counters are reset. | |
| | | 1 | Enabled. The SPI is enabled for operation. | |
| 1 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 2 | MASTER | | Master mode select. | 0 |
| | | 0 | Slave mode. The SPI will operate in slave mode. SCK, MOSI, and the SSEL signals are inputs, MISO is an output. | |
| | | 1 | Master mode. The SPI will operate in master mode. SCK, MOSI, and the SSEL signals are outputs, MISO is an input. | |

Table 319. SPI Configuration register (CFG, offset 0x00) bit description ...continued

| Bit | Symbol | Value | Description | Reset value |
|-------|--------|-------|---|-------------|
| 3 | LSBF | | LSB First mode enable. | 0 |
| | | 0 | Standard. Data is transmitted and received in standard MSB first order. | |
| | | 1 | Reverse. Data is transmitted and received in reverse order (LSB first). | |
| 4 | CPHA | | Clock Phase select. | 0 |
| | | 0 | Change. The SPI captures serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is changed on the following edge. | |
| | | 1 | Capture. The SPI changes serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is captured on the following edge. | |
| 5 | CPOL | | Clock Polarity select. | 0 |
| | | 0 | Low. The rest state of the clock (between transfers) is low. | |
| | | 1 | High. The rest state of the clock (between transfers) is high. | |
| 6 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 7 | LOOP | | Loopback mode enable. Loopback mode applies only to Master mode, and connects transmit and receive data connected together to allow simple software testing. | 0 |
| | | 0 | Disabled. | |
| | | 1 | Enabled. | |
| 8 | SPOL0 | | SSEL0 Polarity select. | 0 |
| | | 0 | Low. The SSEL0 pin is active low. | |
| | | 1 | High. The SSEL0 pin is active high. | |
| 9 | SPOL1 | | SSEL1 Polarity select. | 0 |
| | | 0 | Low. The SSEL1 pin is active low. | |
| | | 1 | High. The SSEL1 pin is active high. | |
| 10 | SPOL2 | | SSEL2 Polarity select. | 0 |
| | | 0 | Low. The SSEL2 pin is active low. | |
| | | 1 | High. The SSEL2 pin is active high. | |
| 11 | SPOL3 | | SSEL3 Polarity select. | 0 |
| | | 0 | Low. The SSEL3 pin is active low. | |
| | | 1 | High. The SSEL3 pin is active high. | |
| 31:12 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

23.6.2 SPI Delay register

The DLY register controls several programmable delays related to SPI signalling. These delays apply only to master mode, and are all stated in SPI clocks.

Timing details are shown in:

[Section 23.7.2.1 “Pre_delay and Post_delay”](#)

[Section 23.7.2.2 “Frame_delay”](#)

[Section 23.7.2.3 “Transfer_delay”](#)

Table 320. SPI Delay register (DLY, offset 0x04) bit description

| Bit | Symbol | Description | Reset value |
|-------|----------------|--|-------------|
| 3:0 | PRE_DELAY | Controls the amount of time between SSEL assertion and the beginning of a data transfer. There is always one SPI clock time between SSEL assertion and the first clock edge. This is not considered part of the pre-delay. 0x0 = No additional time is inserted. 0x1 = 1 SPI clock time is inserted. 0x2 = 2 SPI clock times are inserted. ... 0xF = 15 SPI clock times are inserted. | 0 |
| 7:4 | POST_DELAY | Controls the amount of time between the end of a data transfer and SSEL deassertion. 0x0 = No additional time is inserted. 0x1 = 1 SPI clock time is inserted. 0x2 = 2 SPI clock times are inserted. ... 0xF = 15 SPI clock times are inserted. | 0 |
| 11:8 | FRAME_DELAY | If the EOF flag is set, controls the minimum amount of time between the current frame and the next frame (or SSEL deassertion if EOT). 0x0 = No additional time is inserted. 0x1 = 1 SPI clock time is inserted. 0x2 = 2 SPI clock times are inserted. ... 0xF = 15 SPI clock times are inserted. | 0 |
| 15:12 | TRANSFER_DELAY | Controls the minimum amount of time that the SSEL is deasserted between transfers. 0x0 = The minimum time that SSEL is deasserted is 1 SPI clock time. (Zero added time.) 0x1 = The minimum time that SSEL is deasserted is 2 SPI clock times. 0x2 = The minimum time that SSEL is deasserted is 3 SPI clock times. ... 0xF = The minimum time that SSEL is deasserted is 16 SPI clock times. | 0 |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | NA |

23.6.3 SPI Status register

The STAT register provides SPI status flags for software to read, and a control bit for forcing an end of transfer. Flags other than read-only flags may be cleared by writing ones to corresponding bits of STAT.

STAT contains 2 error flags (in slave mode only): RXOV and TXUR. These are receiver overrun and transmit underrun, respectively. If either of these errors occur during operation, the SPI should be disabled, then re-enabled in order to make sure all internal states are cleared before attempting to resume operation.

In this register, the following notation is used: RO = Read-only, W1 = write 1 to clear.

Table 321. SPI Status register (STAT, offset 0x08) bit description

| Bit | Symbol | Description | Reset value | Access [1] |
|------|--------------|---|-------------|----------------------------|
| 0 | RXRDY | Receiver Ready flag. When 1, indicates that data is available to be read from the receiver buffer. Cleared after a read of the RXDAT register. | 0 | RO |
| 1 | TXRDY | Transmitter Ready flag. When 1, this bit indicates that data may be written to the transmit buffer. Previous data may still be in the process of being transmitted. Cleared when data is written to TXDAT or TXDATCTL until the data is moved to the transmit shift register. | 1 | RO |
| 2 | RXOV | Receiver Overrun interrupt flag. This flag applies only to slave mode (Master = 0). This flag is set when the beginning of a received character is detected while the receiver buffer is still in use. If this occurs, the receiver buffer contents are preserved, and the incoming data is lost. Data received by the SPI should be considered undefined if RxOv is set. | 0 | W1 |
| 3 | TXUR | Transmitter Underrun interrupt flag. This flag applies only to slave mode (Master = 0). In this case, the transmitter must begin sending new data on the next input clock if the transmitter is idle. If that data is not available in the transmitter holding register at that point, there is no data to transmit and the TXUR flag is set. Data transmitted by the SPI should be considered undefined if TXUR is set. | 0 | W1 |
| 4 | SSA | Slave Select Assert. This flag is set whenever any slave select transitions from deasserted to asserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become busy, and allows waking up the device from reduced power modes when a slave mode access begins. This flag is cleared by software. | 0 | W1 |
| 5 | SSD | Slave Select Deassert. This flag is set whenever any asserted slave selects transition to deasserted, in both master and slave modes. This allows determining when the SPI transmit/receive functions become idle. This flag is cleared by software. | 0 | W1 |
| 6 | STALLED | Stalled status flag. This indicates whether the SPI is currently in a stall condition. | 0 | RO |
| 7 | END TRANSFER | End Transfer control bit. Software can set this bit to force an end to the current transfer when the transmitter finishes any activity already in progress, as if the EOT flag had been set prior to the last transmission. This capability is included to support cases where it is not known when transmit data is written that it will be the end of a transfer. The bit is cleared when the transmitter becomes idle as the transfer comes to an end. Forcing an end of transfer in this manner causes any specified FRAME_DELAY and TRANSFER_DELAY to be inserted. | 0 | RO/W1 |
| 8 | MSTIDLE | Master idle status flag. This bit is 1 whenever the SPI master function is fully idle. This means that the transmit holding register is empty and the transmitter is not in the process of sending data. | 1 | RO |
| 31:9 | - | Reserved. Read value is undefined, only zero should be written. | NA | NA |

[1] RO = Read-only, W1 = write 1 to clear.

23.6.4 SPI Interrupt Enable read and Set register

The INTENSET register is used to enable various SPI interrupt sources. Enable bits in INTENSET are mapped in locations that correspond to the flags in the STAT register. The complete set of interrupt enables may be read from this register. Writing ones to implemented bits in this register causes those bits to be set. The INTENCLR register is used to clear bits in this register. See [Table 321](#) for details of the interrupts.

Table 322. SPI Interrupt Enable read and Set register (INTENSET, offset 0x0C) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|---------|-------|--|-------------|
| 0 | RXRDYEN | | RX ready interrupt enable. Determines whether an interrupt occurs when receiver data is available. | 0 |
| | | 0 | Disabled. No interrupt will be generated when receiver data is available. | |
| | | 1 | Enabled. An interrupt will be generated when receiver data is available in the RXDAT register. | |
| 1 | TXRDYEN | | TX ready interrupt enable. Determines whether an interrupt occurs when the transmitter holding register is available. | 0 |
| | | 0 | Disabled. No interrupt will be generated when the transmitter holding register is available. | |
| | | 1 | Enabled. An interrupt will be generated when data may be written to TXDAT. | |
| 2 | RXOVEN | | RX overrun interrupt enable. Determines whether an interrupt occurs when a receiver overrun occurs. This happens in slave mode when there is a need for the receiver to move newly received data to the RXDAT register when it is already in use. The interface prevents receiver overrun in Master mode by not allowing a new transmission to begin when a receiver overrun would otherwise occur. | 0 |
| | | 0 | Disabled. No interrupt will be generated when a receiver overrun occurs. | |
| | | 1 | Enabled. An interrupt will be generated if a receiver overrun occurs. | |
| 3 | TXUREN | | TX underrun interrupt enable. Determines whether an interrupt occurs when a transmitter underrun occurs. This happens in slave mode when there is a need to transmit data when none is available. | 0 |
| | | 0 | Disabled. No interrupt will be generated when the transmitter underruns. | |
| | | 1 | Enabled. An interrupt will be generated if the transmitter underruns. | |
| 4 | SSAEN | | Slave select assert interrupt enable. Determines whether an interrupt occurs when the Slave Select is asserted. | 0 |
| | | 0 | Disabled. No interrupt will be generated when any Slave Select transitions from deasserted to asserted. | |
| | | 1 | Enabled. An interrupt will be generated when any Slave Select transitions from deasserted to asserted. | |
| 5 | SSDEN | | Slave select deassert interrupt enable. Determines whether an interrupt occurs when the Slave Select is deasserted. | 0 |
| | | 0 | Disabled. No interrupt will be generated when all asserted Slave Selects transition to deasserted. | |
| | | 1 | Enabled. An interrupt will be generated when all asserted Slave Selects transition to deasserted. | |
| 7:6 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

Table 322. SPI Interrupt Enable read and Set register (INTENSET, offset 0x0C) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|-----------|-------|--|-------------|
| 8 | MSTIDLEEN | | Master idle interrupt enable. | 0 |
| | | 0 | No interrupt will be generated when the SPI master function is idle. | |
| | | 1 | An interrupt will be generated when the SPI master function is fully idle. | |
| 31:9 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

23.6.5 SPI Interrupt Enable Clear register

The INTENCLR register is used to clear interrupt enable bits in the INTENSET register.

Table 323. SPI Interrupt Enable clear register (INTENCLR, offset 0x10) bit description

| Bit | Symbol | Description | Reset value |
|------|---------|--|-------------|
| 0 | RXRDYEN | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 1 | TXRDYEN | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 2 | RXOVEN | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 3 | TXUREN | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 4 | SSAEN | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 5 | SSDEN | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 7:6 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 8 | MSTIDLE | Writing 1 clears the corresponding bit in the INTENSET register. | 0 |
| 31:9 | - | Reserved. Read value is undefined, only zero should be written. | NA |

23.6.6 SPI Receiver Data register

The read-only RXDAT register provides the means to read the most recently received data. The value of SSEL can be read along with the data.

For details on the slave select process, see [Section 23.7.4](#).

Remark: If the SPI is used with FIFO support, do not use this register to receive data, see [Chapter 25](#).

Table 324. SPI Receiver Data register (RXDAT, offset 0x14) bit description

| Bit | Symbol | Description | Reset value |
|-------|-----------|---|-------------|
| 15:0 | RXDAT | Receiver Data. This contains the next piece of received data. The number of bits that are used depends on the LEN setting in TXCTL / TXDATCTL. | undefined |
| 16 | RXSSEL0_N | Slave Select for receive. This field allows the state of the SSEL0 pin to be saved along with received data. The value will reflect the SSEL0 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG. | undefined |
| 17 | RXSSEL1_N | Slave Select for receive. This field allows the state of the SSEL1 pin to be saved along with received data. The value will reflect the SSEL1 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG. | undefined |
| 18 | RXSSEL2_N | Slave Select for receive. This field allows the state of the SSEL2 pin to be saved along with received data. The value will reflect the SSEL2 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG. | undefined |
| 19 | RXSSEL3_N | Slave Select for receive. This field allows the state of the SSEL3 pin to be saved along with received data. The value will reflect the SSEL3 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG. | undefined |
| 20 | SOT | Start of Transfer flag. This flag will be 1 if this is the first data after the SSELs went from deasserted to asserted (i.e., any previous transfer has ended). This information can be used to identify the first piece of data in cases where the transfer length is greater than 16 bit. | |
| 31:21 | - | Reserved, the value read from a reserved bit is not defined. | NA |

23.6.7 SPI Transmitter Data and Control register

The TXDATCTL register provides a location where both transmit data and control information can be written simultaneously. This allows detailed control of the SPI without a separate write of control information for each piece of data, which can be especially useful when the SPI is used with DMA (see [Section 23.7.5](#)).

Remark: The SPI has no receiver control registers. Hence software needs to set the data length in the transmitter control or transmitter data and control register first in order to handle reception with correct data length. The programmed data length becomes active only when data is actually transmitted. Therefore, this must be done before any data can be received.

When control information remains static during transmit, the TXDAT register should be used (see [Section 23.6.8](#)) instead of the TXDATCTL register. Control information can then be written separately via the TXCTL register (see [Section 23.6.9](#)). The upper part of TXDATCTL (bits 27 to 16) are the same bits contained in the TXCTL register. The two registers simply provide two ways to access them.

For details on the slave select process, see [Section 23.7.4](#).

For details on using multiple consecutive data transmits for transfer lengths larger than 16 bit, see [Section 23.7.6 “Data lengths greater than 16 bits”](#).

Remark: If the SPI is used with FIFO support, do not use this register to transmit data, see [Chapter 25](#).

Table 325. SPI Transmitter Data and Control register (TXDATCTL, offset 0x18) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|-----------|---------------------|---|-------------|
| 15:0 | TXDAT | | Transmit Data. This field provides from 1 to 16 bits of data to be transmitted. | 0 |
| 16 | TXSSEL0_N | | Transmit Slave Select. This field asserts SSEL0 in master mode. The output on the pin is active LOW by default. | 0 |
| | | | Remark: The active state of the SSEL0 pin is configured by bits in the CFG register. | |
| | | 0 | SSEL0 asserted. | |
| | 1 | SSEL0 not asserted. | | |
| 17 | TXSSEL1_N | | Transmit Slave Select. This field asserts SSEL1 in master mode. The output on the pin is active LOW by default. | 0 |
| | | | Remark: The active state of the SSEL1 pin is configured by bits in the CFG register. | |
| | | 0 | SSEL1 asserted. | |
| | 1 | SSEL1 not asserted. | | |
| 18 | TXSSEL2_N | | Transmit Slave Select. This field asserts SSEL2 in master mode. The output on the pin is active LOW by default. | 0 |
| | | | Remark: The active state of the SSEL2 pin is configured by bits in the CFG register. | |
| | | 0 | SSEL2 asserted. | |
| | 1 | SSEL2 not asserted. | | |

Table 325. SPI Transmitter Data and Control register (TXDATCTL, offset 0x18) bit description ...continued

| Bit | Symbol | Value | Description | Reset value |
|-------|-----------|-------|---|-------------|
| 19 | TXSSEL3_N | | Transmit Slave Select. This field asserts SSEL3 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL3 pin is configured by bits in the CFG register. | 0 |
| | | 0 | SSEL3 asserted. | |
| | | 1 | SSEL3 not asserted. | |
| 20 | EOT | | End of Transfer. The asserted SSEL will be deasserted at the end of a transfer, and remain so for at least the time specified by the Transfer_delay value in the DLY register. | 0 |
| | | 0 | SSEL not deasserted. This piece of data is not treated as the end of a transfer. SSEL will not be deasserted at the end of this data. | |
| | | 1 | SSEL deasserted. This piece of data is treated as the end of a transfer. SSEL will be deasserted at the end of this piece of data. | |
| 21 | EOF | | End of Frame. Between frames, a delay may be inserted, as defined by the FRAME_DELAY value in the DLY register. The end of a frame may not be particularly meaningful if the FRAME_DELAY value = 0. This control can be used as part of the support for frame lengths greater than 16 bits. | 0 |
| | | 0 | Data not EOF. This piece of data transmitted is not treated as the end of a frame. | |
| | | 1 | Data EOF. This piece of data is treated as the end of a frame, causing the FRAME_DELAY time to be inserted before subsequent data is transmitted. | |
| 22 | RXIGNORE | | Receive Ignore. This allows data to be transmitted using the SPI without the need to read unneeded data from the receiver. Setting this bit simplifies the transmit process and can be used with the DMA. | 0 |
| | | 0 | Read received data. Received data must be read in order to allow transmission to progress. In slave mode, an overrun error will occur if received data is not read before new data is received. | |
| | | 1 | Ignore received data. Received data is ignored, allowing transmission without reading unneeded received data. No receiver flags are generated. | |
| 23 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 27:24 | LEN | | Data Length. Specifies the data length from 1 to 16 bits. Note that transfer lengths greater than 16 bits are supported by implementing multiple sequential transmits. 0x0 = Data transfer is 1 bit in length. Note: when LEN = 0, the underrun status is not meaningful. 0x1 = Data transfer is 2 bits in length. 0x2 = Data transfer is 3 bits in length. ... 0xF = Data transfer is 16 bits in length. | 0x0 |
| 31:28 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

23.6.8 SPI Transmitter Data Register

The TXDAT register is written in order to send data via the SPI transmitter when control information is not changing during the transfer (see [Section 23.6.7](#)). That data will be sent to the transmit shift register when it is available, and another character may then be written to TXDAT.

Remark: If the SPI is used with FIFO support, do not use this register to transmit data, see [Chapter 25](#).

Table 326. SPI Transmitter Data Register (TXDAT, offset 0x1C) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 15:0 | DATA | Transmit Data. This field provides from 4 to 16 bits of data to be transmitted. | 0 |
| 31:16 | - | Reserved. Only zero should be written. | NA |

23.6.9 SPI Transmitter Control register

The TXCTL register provides a way to separately access control information for the SPI. These bits are another view of the same-named bits in the TXDATCTL register (see [Section 23.6.7](#)). Changing bits in TXCTL has no effect unless data is later written to the TXDAT register. Data written to TXDATCTL overwrites the TXCTL register.

When control information needs to be changed during transmission, the TXDATCTL register should be used (see [Section 23.6.7](#)) instead of TXDAT. Control information can then be written along with data.

Table 327. SPI Transmitter Control register (TXCTL, offset 0x20) bit description

| Bit | Symbol | Description | Reset value |
|-------|-----------|---|-------------|
| 15:0 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 16 | TXSSEL0_N | Transmit Slave Select 0. | 0x0 |
| 17 | TXSSEL1_N | Transmit Slave Select 1. | 0x0 |
| 18 | TXSSEL2_N | Transmit Slave Select 2. | 0x0 |
| 19 | TXSSEL3_n | Transmit Slave Select 3. | 0x0 |
| 20 | EOT | End of Transfer. | 0 |
| 21 | EOF | End of Frame. | 0 |
| 22 | RXIGNORE | Receive Ignore. | 0 |
| 23 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 27:24 | LEN | Data transfer Length. | 0x0 |
| 31:28 | - | Reserved. Read value is undefined, only zero should be written. | NA |

23.6.10 SPI Divider register

The DIV register determines the clock used by the SPI in master mode.

For details on clocking, see [Section 23.7.3 “Clocking and data rates”](#).

Table 328. SPI Divider register (DIV, offset 0x24) bit description

| Bit | Symbol | Description | Reset Value |
|-------|--------|---|-------------|
| 15:0 | DIVVAL | Rate divider value. Specifies how the PCLK for the SPI is divided to produce the SPI clock rate in master mode. DIVVAL is -1 encoded such that the value 0 results in PCLK/1, the value 1 results in PCLK/2, up to the maximum possible divide value of 0xFFFF, which results in PCLK/65536. | 0 |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | NA |

23.6.11 SPI Interrupt Status register

The read-only INTSTAT register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See [Table 321](#) for detailed descriptions of the interrupt flags.

Table 329. SPI Interrupt Status register (INTSTAT, offset 0x28) bit description

| Bit | Symbol | Description | Reset value |
|------|---------|---|-------------|
| 0 | RXRDY | Receiver Ready flag. | 0 |
| 1 | TXRDY | Transmitter Ready flag. | 1 |
| 2 | RXOV | Receiver Overrun interrupt flag. | 0 |
| 3 | TXUR | Transmitter Underrun interrupt flag. | 0 |
| 4 | SSA | Slave Select Assert. | 0 |
| 5 | SSD | Slave Select Deassert. | 0 |
| 7:6 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 8 | MSTIDLE | Master Idle status flag. | 0 |
| 31:9 | - | Reserved. Read value is undefined, only zero should be written. | NA |

23.7 Functional description

23.7.1 Operating modes: clock and phase selection

SPI interfaces typically allow configuration of clock phase and polarity. These are sometimes referred to as numbered SPI modes, as described in [Table 330](#) and shown in [Figure 47](#). CPOL and CPHA are configured by bits in the CFG register ([Section 23.6.1](#)).

Table 330: SPI mode summary

| CPOL | CPHA | SPI Mode | Description | SCK rest state | SCK data change edge | SCK data sample edge |
|------|------|----------|--|----------------|----------------------|----------------------|
| 0 | 0 | 0 | The SPI captures serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is changed on the following edge. | low | falling | rising |
| 0 | 1 | 1 | The SPI changes serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is captured on the following edge. | low | rising | falling |
| 1 | 0 | 2 | Same as mode 0 with SCK inverted. | high | rising | falling |
| 1 | 1 | 3 | Same as mode 1 with SCK inverted. | high | falling | rising |

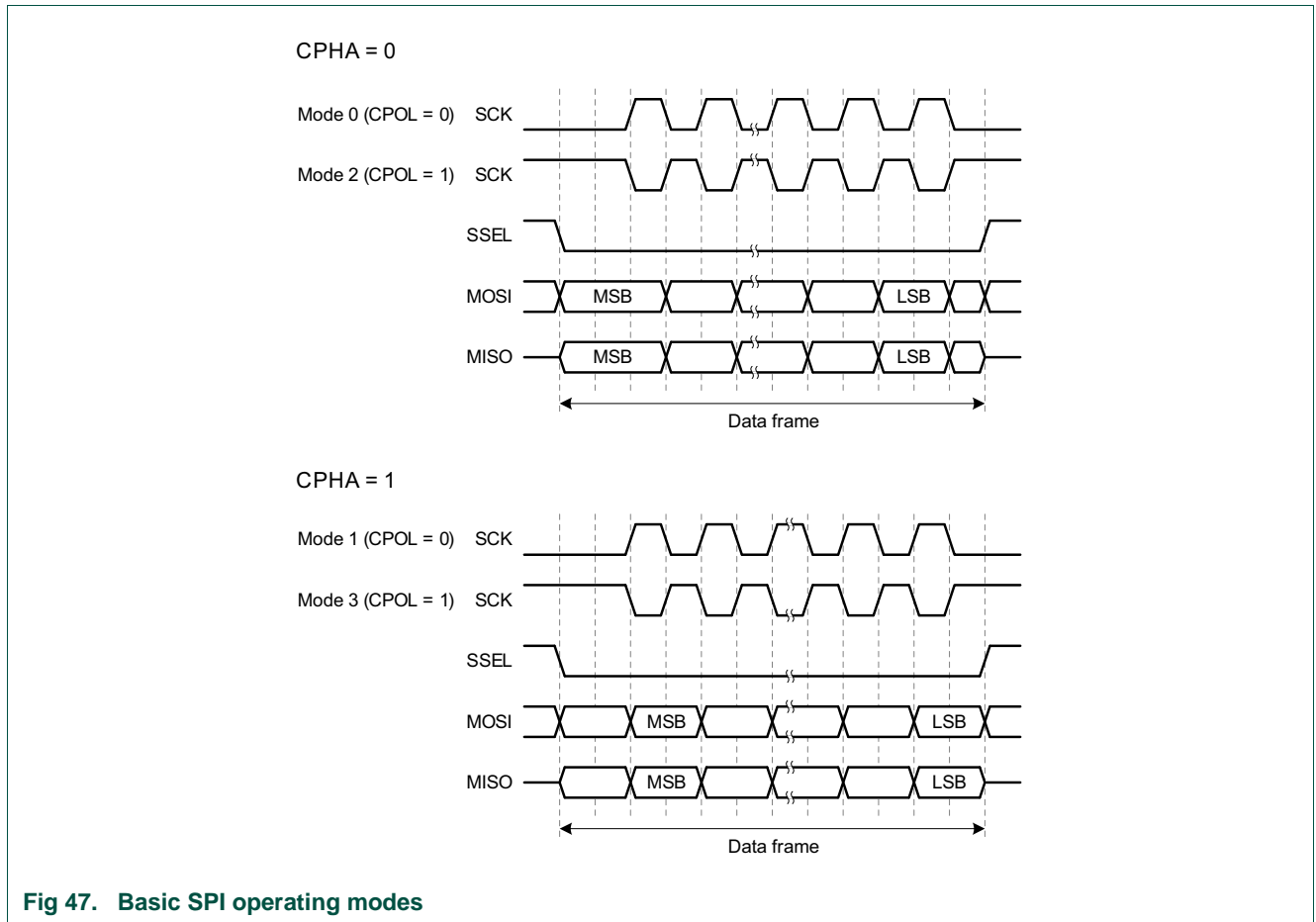


Fig 47. Basic SPI operating modes

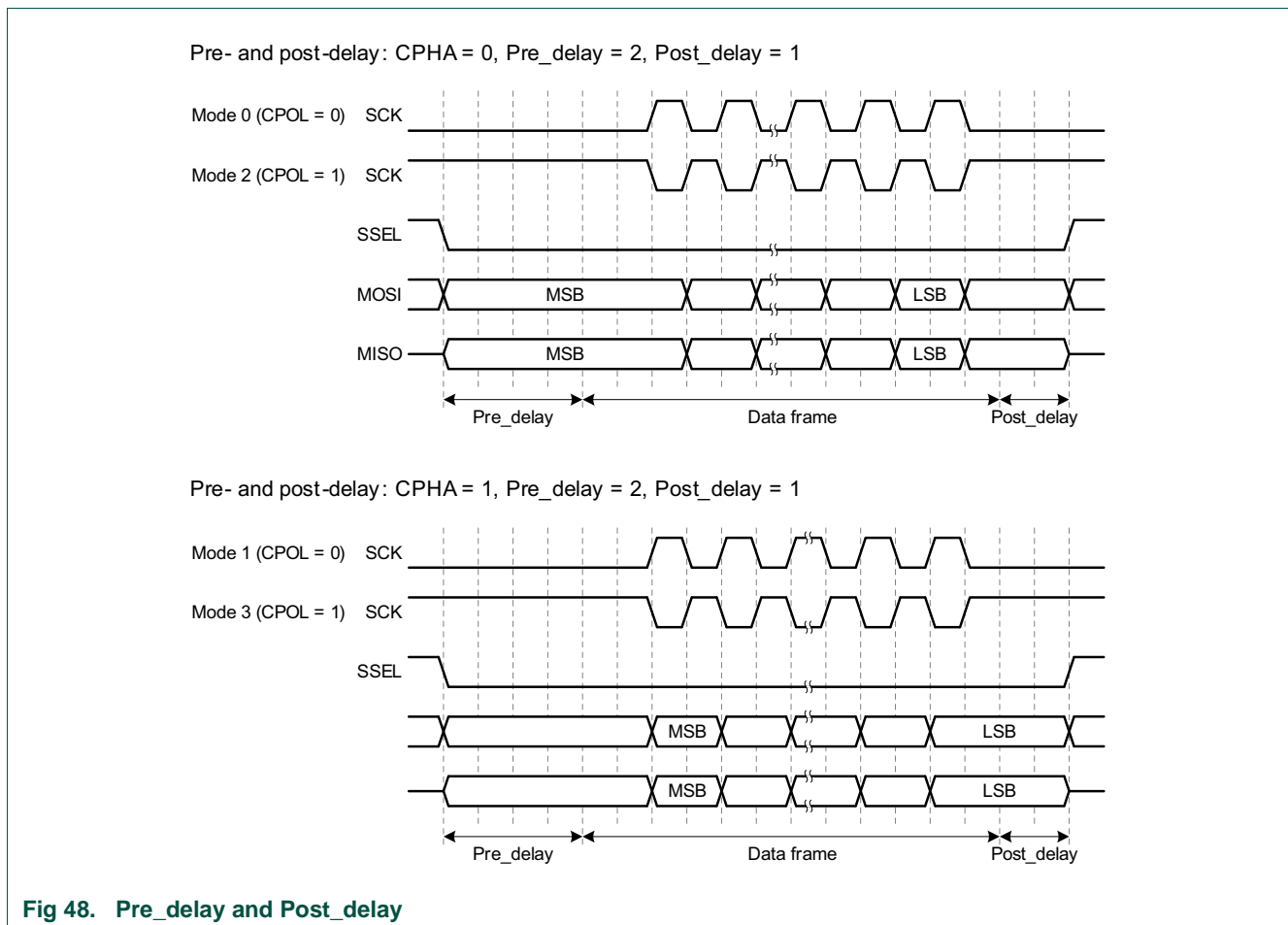
23.7.2 Frame delays

Several delays can be specified for SPI frames. These include:

- Pre_delay: delay after SSEL is asserted before data clocking begins
- Post_delay: delay at the end of a data frame before SSEL is deasserted
- Frame_delay: delay between data frames when SSEL is not deasserted
- Transfer_delay: minimum duration of SSEL in the deasserted state between transfers

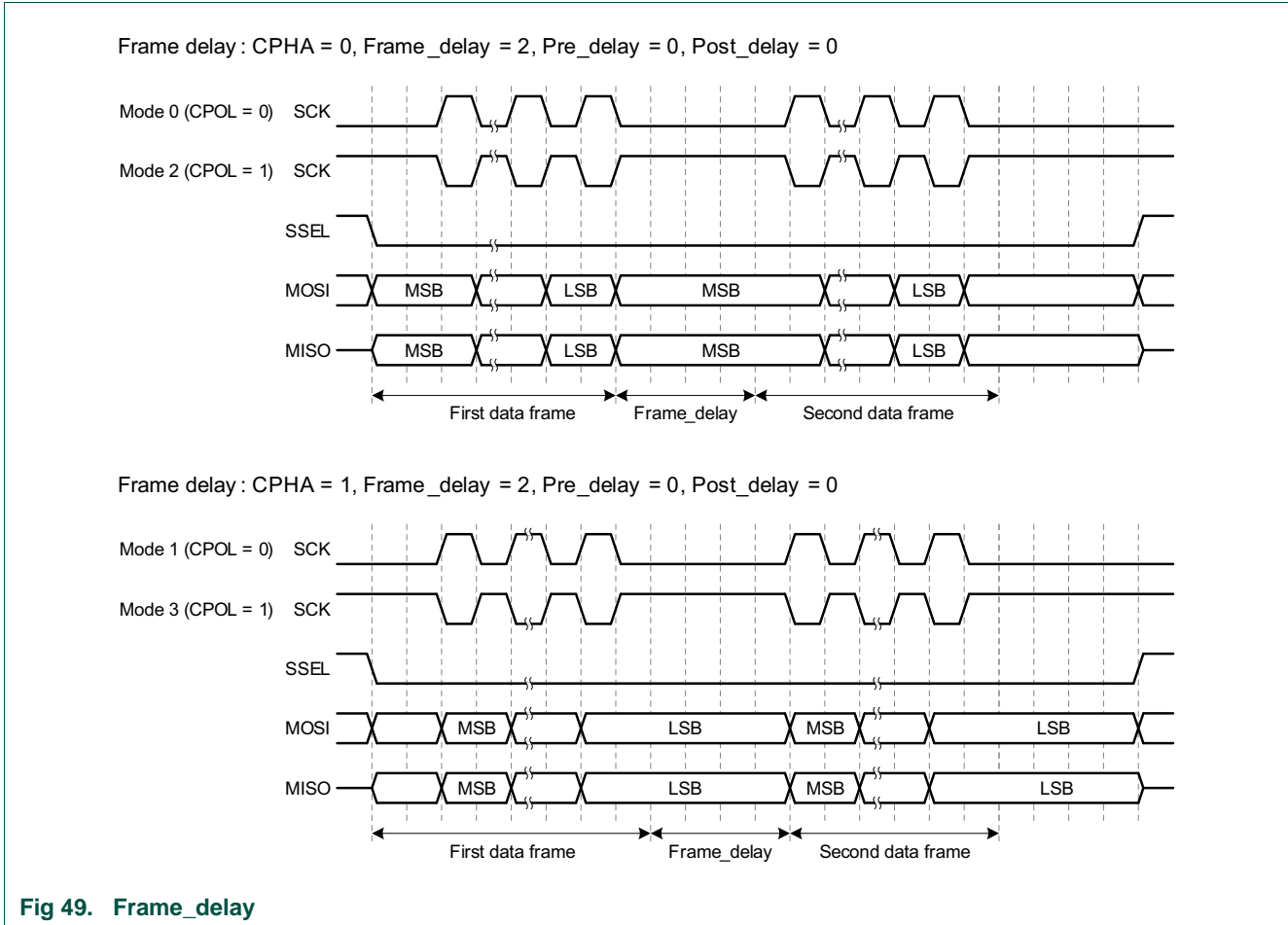
23.7.2.1 Pre_delay and Post_delay

Pre_delay and Post_delay are illustrated by the examples in [Figure 48](#). The Pre_delay value controls the amount of time between SSEL being asserted and the beginning of the subsequent data frame. The Post_delay value controls the amount of time between the end of a data frame and the deassertion of SSEL.



23.7.2.2 Frame_delay

The Frame_delay value controls the amount of time at the end of each frame. This delay is inserted when the EOF bit = 1. Frame_delay is illustrated by the examples in [Figure 49](#). Note that frame boundaries occur only where specified. This is because frame lengths can be any size, involving multiple data writes. See [Section 23.7.6](#) for more information.



23.7.2.3 Transfer_delay

The Transfer_delay value controls the minimum amount of time that SSEL is deasserted between transfers, because the EOT bit = 1. When Transfer_delay = 0, SSEL may be deasserted for a minimum of one SPI clock time. Transfer_delay is illustrated by the examples in [Figure 50](#).

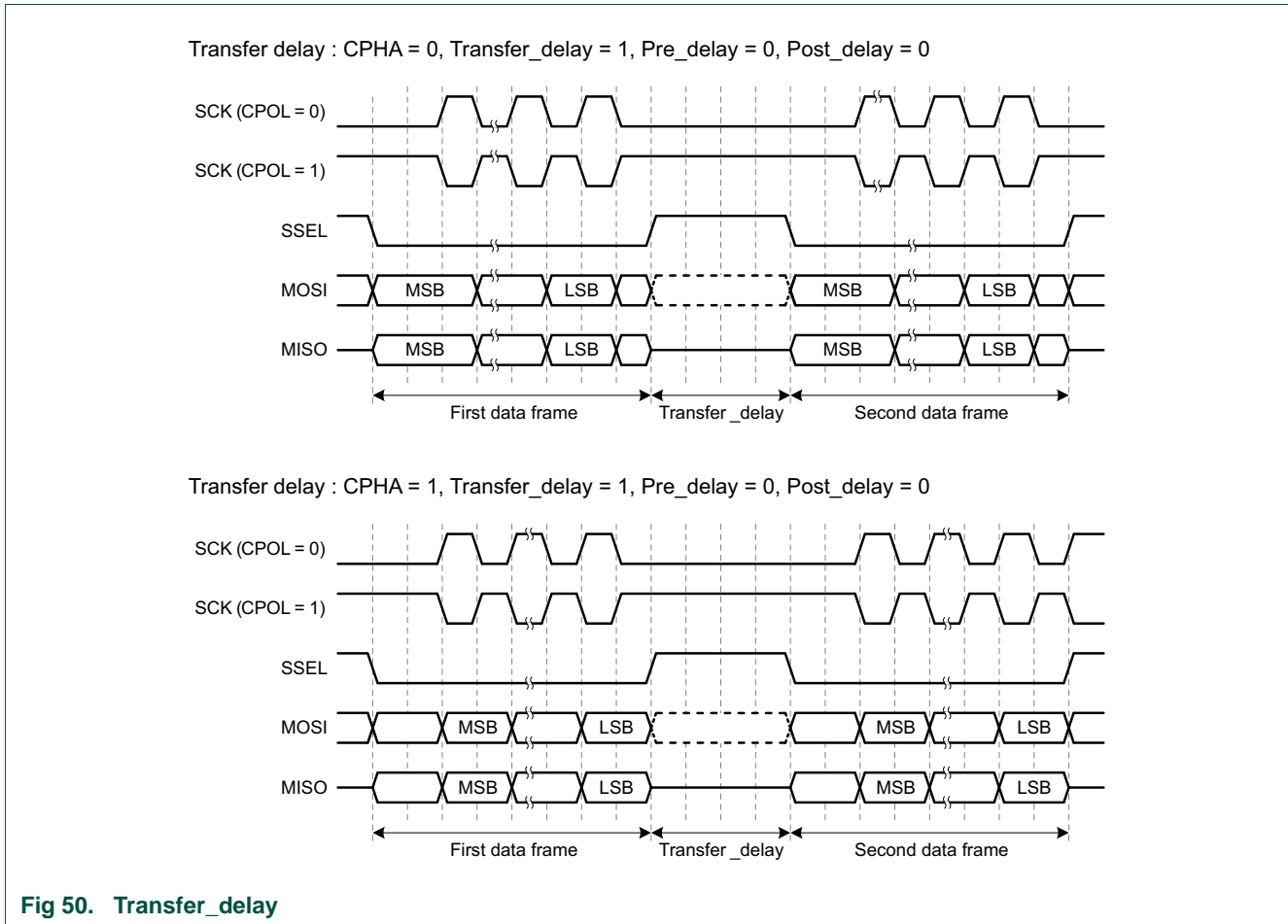


Fig 50. Transfer_delay

23.7.3 Clocking and data rates

In order to use the SPI, clocking details must be defined. This includes configuring the system clock and selection of the clock divider value in DIV. See [Figure 3 "Clock generation"](#).

23.7.3.1 Data rate calculations

The SPI interface is designed to operate asynchronously from any on-chip clocks, and without the need for overclocking.

In slave mode, this means that the SCK from the external master is used directly to run the transmit and receive shift registers and other logic.

In master mode, the SPI rate clock produced by the SPI clock divider is used directly as the outgoing SCK.

The SPI clock divider is an integer divider. The SPI in master mode can be set to run at the same speed as the selected PCLK, or at lower integer divide rates. The SPI rate will be $= \text{PCLK_SPIn} / \text{DIVVAL}$.

In slave mode, the clock is taken from the SCK input and the SPI clock divider is not used.

23.7.4 Slave select

The SPI block provides for four Slave Select inputs in slave mode or outputs in master mode. Each SSEL can be set for normal polarity (active low), or can be inverted (active high). Representation of the 4 SSELs in a register is always active low. If an SSEL is inverted, this is done as the signal leaves/enters the SPI block.

In slave mode, **any** asserted SSEL that is connected to a pin will activate the SPI. In master mode, all SSELs that are connected to a pin will be output as defined in the SPI registers. In the latter case, the SSELs could potentially be decoded externally in order to address more than four slave devices. Note that at least one SSEL is asserted when data is transferred in master mode.

In master mode, Slave Selects come from the TXSSEL bits in the CTL or DATCTL registers. In slave mode, the state of all four SSELs is saved along with received data in the RXSSEL_N field of the RXDAT register.

23.7.5 DMA operation

A DMA request is provided for each SPI direction, and can be used in lieu of interrupts for transferring data by configuring the DMA controller appropriately. The DMA controller provides an acknowledgement signal that clears the related request when it completes handling that request.

The transmitter DMA request is asserted when Tx DMA is enabled and the transmitter can accept more data.

The receiver DMA request is asserted when Rx DMA is enabled and received data is available to be read.

23.7.5.1 DMA master mode End-Of-Transfer

When using polled or interrupt mode to transfer data in master mode, the transition to end-of-transfer status (drive SSEL inactive) is straightforward. The TXDATCTL EOT bit can be set along with the last data item to be transmitted or the Status register's END TRANSFER bit can be set after the last data item is written to the TXDAT register.

When using the DMA in master mode, the end-of-transfer status (drive SSEL inactive) can be generated it 3 ways:

1. The simplest way is using 32 bit wide DMA transfers. This allows the TXCTL bits to be included with the data. The EOT bit of the last word transferred can be set, to de-assert the SSEL after the data is transmitted.
2. Another way is through the SPI Tx DMA interrupt handler. This interrupt handler can set the SPI Status register (STAT) END TRANSFER control bit at the completion of the DMA transfer.
3. A third way is to use the DMA controller's linked descriptor capability. The DMA controller provides for a linked list of DMA transfer control descriptors. The initial descriptor(s) can be used to transfer most of the data. A final DMA descriptor can be used to send a single 32 bit wide DMA transfer to TXDATCTL that includes EOT along with the last of the data.

23.7.6 Data lengths greater than 16 bits

The SPI interface handles data frame sizes from 1 to 16 bits directly. Larger sizes can be handled by splitting data up into groups of 16 bits or less. For example, 24 bits can be supported as 2 groups of 16 bits and 8 bits or 2 groups of 12 bits, among others. Frames of any size, including greater than 32 bits, can supported in the same way.

Details of how to handle larger data widths depend somewhat on other SPI configuration options. For instance, if it is intended for Slave Selects to be deasserted between frames, then this must be suppressed when a larger frame is split into more than one part. Sending 2 groups of 12 bits with SSEL deasserted between 24-bit increments, for instance, would require changing the value of the EOF bit on alternate 12-bit frames.

23.7.7 Data stalls

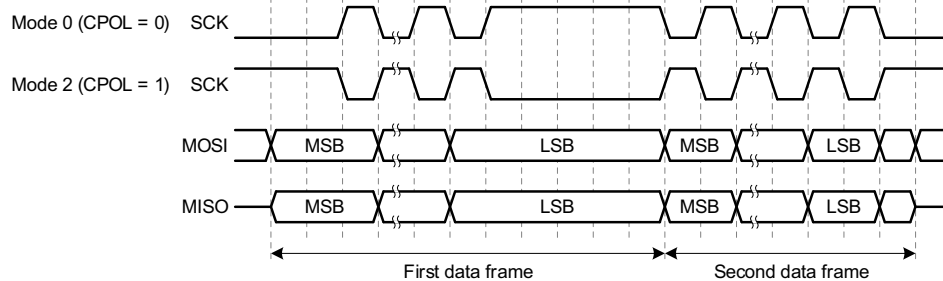
A stall for Master transmit data can happen in modes 0 and 2 when SCK cannot be returned to the rest state until the MSB of the next data frame can be driven on MOSI. In this case, the stall happens just before the final clock edge of data if the next piece of data is not yet available.

A stall for Master receive can happen when a receiver overrun would otherwise occur if the transmitter was not stalled. In modes 0 and 2, this occurs if the previously received data is not read before the end of the next piece of is received. This stall happens one clock edge earlier than the transmitter stall.

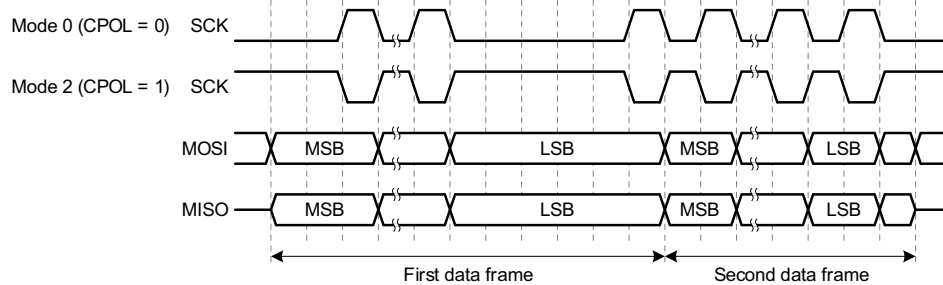
In modes 1 and 3, the same kind of receiver stall can occur, but just before the final clock edge of the received data. Also, a transmitter stall will not happen in modes 1 and 3 because the transmitted data is complete at the point where a stall would otherwise occur, so it is not needed.

Stalls are reflected in the STAT register by the Stalled status flag, which indicates the current SPI status.

Transmitter stall: CPHA = 0, Frame_delay = 0, Pre_delay = 0, Post_delay = 0, 2 clock stall



Receiver stall: CPHA = 0, Frame_delay = 0, Pre_delay = 0, Post_delay = 0, 2 clock stall



Receiver stall: CPHA = 1, Frame_delay = 0, Pre_delay = 0, Post_delay = 0, 2 clock stall

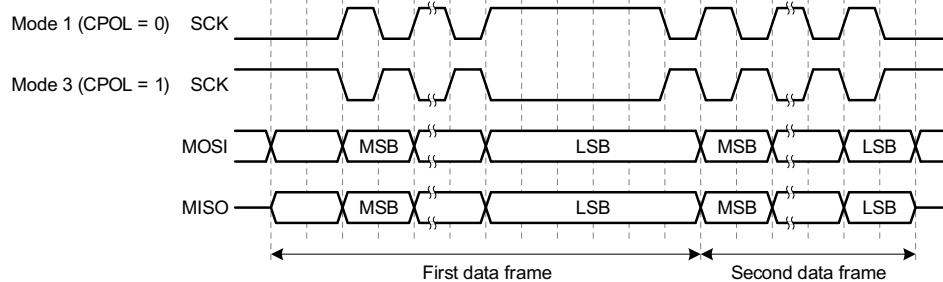


Fig 51. Examples of data stalls

24.1 How to read this chapter

I²C-bus interfaces are available on all parts.

Read this chapter to understand the I²C operation, software interface, and how to use the I²C for wake-up from reduced power modes.

24.2 Features

- Independent Master, Slave, and Monitor functions.
- Bus speeds supported:
 - Standard mode, up to 100 kbits/s.
 - Fast-mode, up to 400 kbits/s.
 - Fast-mode Plus, up to 1 Mbits/s
 - High speed mode, 3.4 Mbits/s as a Slave only.
- Supports both Multi-master and Multi-master with Slave functions.
- Multiple I²C slave addresses supported in hardware.
- One slave address can be selectively qualified with a bit mask or an address range in order to respond to multiple I²C bus addresses.
- 10-bit addressing supported with software assist.
- Supports System Management Bus (SMBus).
- Separate DMA requests for Master, Slave, and Monitor functions.
- No chip clocks are required in order to receive and compare an address as a Slave, so this event can wake up the device from Power-down mode.
- Supports the I²C-bus specification up to Fast-mode Plus (FM+, up to 1 MHz) in both master and slave modes. High-speed (HS, up to 3.4 MHz) I²C is support in slave mode only.

24.3 Pin description

The I²C pins are fixed-pin functions and enabled through IOCON. Refer to the IOCON settings table for I²C modes in [Section 8.5.2](#).

Table 331. I²C-bus pin description

| Function | Direction | Description | Connect to |
|----------|-----------|--------------------|------------|
| I2C0_SCL | I/O | I2C0 serial clock. | P0_23 |
| I2C0_SDA | I/O | I2C0 serial data. | P0_24 |
| I2C1_SCL | I/O | I2C1 serial clock. | P0_25 |
| I2C1_SDA | I/O | I2C1 serial data. | P0_26 |
| I2C2_SCL | I/O | I2C2 serial clock. | P0_27 |
| I2C2_SDA | I/O | I2C2 serial data. | P0_28 |

24.4 Basic configuration

Configure I²C blocks using the following registers:

- In the ASYNCAPBCLKCTRL register, set the appropriate bit(s) (see [Table 92](#)) to enable clocks to the register interface.
- Clear the I²C peripheral reset using the ASYNCPRESETCTRL register ([Table 89](#)).
- Enable/disable the related I²C interrupt slot in the NVIC.
- Configure the I²C pin functions through IOCON.
- The peripheral clock for the I²C is the asynchronous APB clock (see [Figure 3](#)).

24.4.1 I²C transmit/receive in master mode

In this example, I2C0 is configured as the master. The master sends 8 bits to the slave and then receives 8 bits from the slave. The system clock is set to 30 MHz and the bit rate is approximately 400 kHz. The I2C0_SCL and I2C0_SDA functions must be enabled on pins PIO0_22 and PIO0_23 through IOCON. See [Section 8.5.2](#).

The pins should be configured as required for the I²C-bus mode that will be used (SM, FM, FM+, HS) via the IOCON block. See [Section 8.5.2](#).

The transmission of the address and data bits is controlled by the state of the MSTPENDING status bit. Whenever the status is Master pending, the master can read or write to the MSTDAT register and go to the next step of the transmission protocol by writing to the MSTCTL register.

Configure the I²C bit rate:

- Divide the system clock (I2C_PCLK) by a factor of 19. See [Table 351 “I²C Clock Divider register \(CLKDIV, offset 0x14\) bit description”](#).
- Set the SCL high and low times to 2 clock cycles each. This is the default. See [Table 357 “Master Time register \(MSTTIME, address offset 0x024\) bit description”](#). The result is an SCL clock of $(30 \text{ MHz} / 19 / (2 + 2)) = 394.7 \text{ kHz}$.

24.4.1.1 Master write to slave

Configure I2C0 as a master: Set the MSTEN bit to 1 in the CFG register. See [Table 339](#).

Write data to the slave:

1. Write the slave address with the \overline{RW} bit set to 0 to the Master data register MSTDAT. See [Table 359](#).
2. Start the transmission by setting the MSTSTART bit to 1 in the Master control register. See [Table 355](#). The following happens:
 - The pending status is cleared and the I²C-bus is busy.
 - The I²C master sends the start bit and address with the \overline{RW} bit to the slave.
3. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
4. Write 8 bits of data to the MSTDAT register.
5. Continue with the transmission of data by setting the MSTCONT bit to 1 in the Master control register. See [Table 355](#). The following happens:
 - The pending status is cleared and the I²C-bus is busy.
 - The I²C master sends the data bits to the slave address.
6. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
7. Stop the transmission by setting the MSTSTOP bit to 1 in the Master control register. See [Table 355](#).

Table 332. Code example

Master write to slave

```
//Master write 1 byte to slave. Address 0x23, Data 0xdd. Polling mode.
I2C->CFG = I2C_CFG_MSTEN;
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
I2C->MSTDAT = (0x23 << 1) | 0; // address and 0 for R/W bit
I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_TX) abort();
I2C->MSTDAT = 0xdd; // send data
I2C->MSTCTL = I2C_MSTCTL_MSTCONTINUE; // continue transaction
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_TX) abort();
I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
```

24.4.1.2 Master read from slave

Configure I2C0 as a master: Set the MSTEN bit to 1 in the CFG register. See [Table 339](#).

Read data from the slave:

1. Write the slave address with the \overline{RW} bit set to 1 to the Master data register MSTDAT. See [Table 359](#).
2. Start the transmission by setting the MSTSTART bit to 1 in the Master control register. See [Table 355](#). The following happens:
 - The pending status is cleared and the I²C-bus is busy.
 - The I²C master sends the start bit and address with the \overline{RW} bit to the slave.
 - The slave sends 8 bit of data.
3. Wait for the pending status to be set (MSTPENDING = 1) by polling the STAT register.
4. Read 8 bits of data from the MSTDAT register.
5. Stop the transmission by setting the MSTSTOP bit to 1 in the Master control register. See [Table 355](#).

Table 333. Code example

Master read from slave

```
// Master read 1 byte from slave. Address 0x23. Polling mode. No error checking.
uint8_t data;
I2C->CFG = I2C_CFG_MSTEN;
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
I2C->MSTDAT = (0x23 << 1) | 1; // address and 1 for Rwn bit
I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_RX) abort();
data = I2C->MSTDAT; // read data
if(data != 0xdd) abort();
I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop
while(!(I2C->STAT & I2C_STAT_MSTPENDING));
if((I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
```

24.4.2 I²C receive/transmit in slave mode

In this example, I2C0 is configured as the slave. The slave receives 8 bits from the master and then sends 8 bits to the master. The system clock is set to 30 MHz and the bit rate is approximately 400 KHz. The I2C0_SCL and I2C0_SDA functions must be enabled on pins PIO0_22 and PIO0_23 through IOCON. See [Section 8.5.2](#).

The pins should be configured as required for the I²C-bus mode that will be used (SM, FM, FM+, HS) via the IOCON block. See [Section 8.5.2](#).

The transmission of the address and data bits is controlled by the state of the SLVPENDING status bit. Whenever the status is Slave pending, the slave can acknowledge (“ack”) or send or receive an address and data. The received data or the data to be sent to the master are available in the SLVDAT register. After sending and receiving data, continue to the next step of the transmission protocol by writing to the SLVCTL register.

24.4.2.1 Slave read from master

Configure I2C0 as slave with address x:

- Set the SLVEN bit to 1 in the CFG register. See [Table 339](#).
- Write the slave address x to the address 0 match register. See [Table 365](#).

Read data from the master:

1. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
2. Acknowledge (“ack”) the address by setting SLVCONTINUE = 1 in the slave control register. See [Table 361](#).
3. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
4. Read 8 bits of data from the SLVDAT register. See [Table 363](#).
5. Acknowledge (“ack”) the data by setting SLVCONTINUE = 1 in the slave control register. See [Table 361](#).

Table 334. Code example

Slave read from master

```
//Slave read 1 byte from master. Address 0x23. Polling mode.
uint8_t data;
I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register
I2C->CFG = I2C_CFG_SLVEN;
I2C->CFG;
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_RX) abort();
data = I2C->SLVDAT; // read data
if(data != 0xdd) abort();
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack data
```

24.4.2.2 Slave write to master

- Set the SLVEN bit to 1 in the CFG register. See [Table 339](#).
- Write the slave address x to the address 0 match register. See [Table 365](#).

Write data to the master:

1. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
2. ACK the address by setting SLVCONTINUE = 1 in the slave control register. See [Table 361](#).
3. Wait for the pending status to be set (SLVPENDING = 1) by polling the STAT register.
4. Write 8 bits of data to SLVDAT register. See [Table 363](#).
5. Continue the transaction by setting SLVCONTINUE = 1 in the slave control register. See [Table 361](#).

Table 335. Code example

Slave write to master

```
//Slave write 1 byte to master. Address 0x23, Data 0xdd. Polling mode.
I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register
I2C->CFG = I2C_CFG_SLVEN;
I2C->CFG;
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address
while(!(I2C->STAT & I2C_STAT_SLVPENDING));
if((I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_TX) abort();
I2C->SLVDAT = 0xdd; // write data
I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // continue transaction
```

24.4.3 Configure the I²C for wake-up

In sleep mode, any activity on the I²C-bus that triggers an I²C interrupt can wake up the part, provided that the interrupt is enabled in the INTENSET register and the NVIC. As long as the I²C clock I2C_PCLK remains active in sleep mode, the I²C can wake up the part independently of whether the I²C block is configured in master or slave mode.

In Deep-sleep or Power-down mode, the I²C clock is turned off as are all peripheral clocks. However, if the I²C is configured in slave mode and an external master on the I²C-bus provides the clock signal, the I²C block can create an interrupt asynchronously. This interrupt, if enabled in the NVIC and in the I²C block's INTENCLR register, can then wake up the core.

24.4.3.1 Wake-up from Sleep mode

- Enable the I²C interrupt in the NVIC.
- Enable the I²C wake-up event in the I2C INTENSET register. Wake-up on any enabled interrupts is supported (see the INTENSET register). Examples are the following events:
 - Master pending
 - Change to idle state

- Start/stop error
- Slave pending
- Address match (in slave mode)
- Data available/ready

24.4.3.2 Wake-up from Deep-sleep and Power-down modes

- Enable the I²C interrupt in the NVIC.
- Enable the I²C interrupt in the STARTER1 register in the SYSCON block to create the interrupt signal asynchronously while the core and the peripheral are not clocked. See [Table 76 “Start enable register 1 \(STARTER1, address 0x4000 0244\) bit description”](#).
- Configure the I²C in slave mode.
- Enable the I²C the interrupt in the I2C INTENCLR register which configures the interrupt as wake-up event. Examples are the following events:
 - Slave deselect
 - Slave pending (wait for read, write, or ACK)
 - Address match
 - Data available/ready for the monitor

24.5 General description

The architecture of the I²C-bus interface is shown in [Figure 52](#).

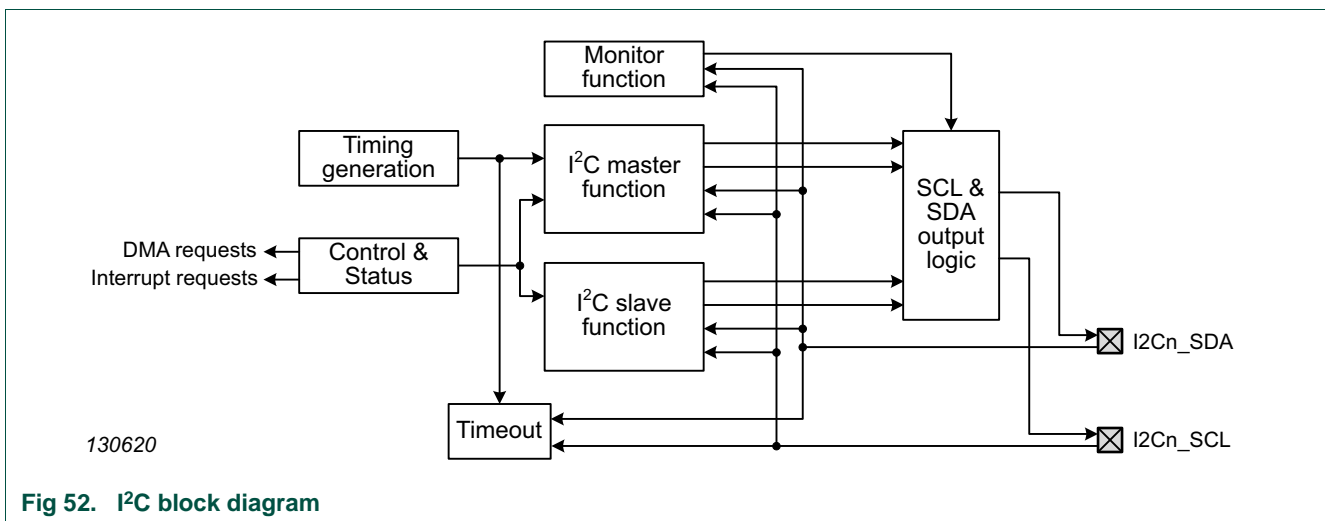


Fig 52. I²C block diagram

24.6 Register description

The base addresses of the I²C blocks are:

Table 336. I²C base addresses

| I ² C block | Base address |
|------------------------|--------------|
| I2C0 | 0x4009 4000 |
| I2C1 | 0x4009 8000 |
| I2C2 | 0x4009 C000 |

The register functions can be grouped as follows:

- Common registers:
 - [Table 339 “I2C Configuration register \(CFG, address offset 0x000\) bit description”](#)
 - [Table 341 “I²C Status register \(STAT, address offset 0x004\) bit description”](#)
 - [Table 353 “I²C Interrupt Status register \(INTSTAT, address offset 0x018\) bit description”](#)
 - [Table 345 “Interrupt Enable Set and read register \(INTENSET, address offset 0x008\) bit description”](#)
 - [Table 347 “Interrupt Enable Clear register \(INTENCLR, address offset 0x00C\) bit description”](#)
 - [Table 349 “Time-out value register \(TIMEOUT, address offset 0x010\) bit description”](#)
 - [Table 351 “I²C Clock Divider register \(CLKDIV, offset 0x14\) bit description”](#)
- Master function registers:
 - [Table 355 “Master Control register \(MSTCTL, address offset 0x020\) bit description”](#)
 - [Table 357 “Master Time register \(MSTTIME, address offset 0x024\) bit description”](#)
 - [Table 359 “Master Data register \(MSTDAT, address offset 0x028\) bit description”](#)
- Slave function registers:
 - [Table 361 “Slave Control register \(SLVCTL, address offset 0x040\) bit description”](#)
 - [Table 363 “Slave Data register \(SLVDAT, address offset 0x044\) bit description”](#)
 - [Table 365 “Slave Address registers \(SLVADR\[0:3\], address offset \[0x048:0x054\]\) bit description”](#)
 - [Table 367 “Slave address Qualifier 0 register \(SLVQUAL0, address offset 0x058\) bit description”](#)
- Monitor function register: [Table 369 “Monitor data register \(MONRXDAT, address offset 0x080\) bit description”](#)

Table 337: Register overview: I2C0/1/2 (register base addresses 0x4009 4000 (I2C0), 0x4009 8000 (I2C1), 0x4009 C000 (I2C2))

| Name | Access | Offset | Description | Reset value | Reference |
|----------|--------|--------|--|-------------|---------------------------|
| CFG | R/W | 0x00 | Configuration for shared functions. | 0 | Table 339 |
| STAT | R/W | 0x04 | Status register for Master, Slave, and Monitor functions. | 0x0801 | Table 341 |
| INTENSET | R/W | 0x08 | Interrupt Enable Set and read register. | 0 | Table 345 |
| INTENCLR | WO | 0x0C | Interrupt Enable Clear register. | NA | Table 347 |
| TIMEOUT | R/W | 0x10 | Time-out value register. | 0xFFFF | Table 349 |
| CLKDIV | R/W | 0x14 | Clock pre-divider for the entire I ² C block. This determines what time increments are used for the MSTTIME register, and controls some timing of the Slave function. | 0 | Table 351 |
| INTSTAT | RO | 0x18 | Interrupt Status register for Master, Slave, and Monitor functions. | 0 | Table 353 |
| MSTCTL | R/W | 0x20 | Master control register. | 0 | Table 355 |
| MSTTIME | R/W | 0x24 | Master timing configuration. | 0x77 | Table 357 |
| MSTDAT | R/W | 0x28 | Combined Master receiver and transmitter data register. | NA | Table 359 |
| SLVCTL | R/W | 0x40 | Slave control register. | 0 | Table 361 |
| SLVDAT | R/W | 0x44 | Combined Slave receiver and transmitter data register. | NA | Table 363 |
| SLVADR0 | R/W | 0x48 | Slave address 0. | 0x01 | Table 365 |
| SLVADR1 | R/W | 0x4C | Slave address 1. | 0x01 | Table 365 |
| SLVADR2 | R/W | 0x50 | Slave address 2. | 0x01 | Table 365 |
| SLVADR3 | R/W | 0x54 | Slave address 3. | 0x01 | Table 365 |
| SLVQUAL0 | R/W | 0x58 | Slave Qualification for address 0. | 0 | Table 367 |
| MONRXDAT | RO | 0x80 | Monitor receiver data register. | 0 | Table 369 |

24.6.1 I2C Configuration register

The CFG register contains mode settings that apply to Master, Slave, and Monitor functions.

Table 338. Address map CFG register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| I2C0 | 0x4009 4000 | 0x000 | - | 1 |
| I2C1 | 0x4009 8000 | 0x000 | - | 1 |
| I2C2 | 0x4009 C000 | 0x000 | - | 1 |

Table 339. I2C Configuration register (CFG, address offset 0x000) bit description

| Bit | Symbol | Value | Description | Reset Value |
|-----|-----------|-------|--|-------------|
| 0 | MSTEN | | Master Enable. When disabled, configurations settings for the Master function are not changed, but the Master function is internally reset. | 0 |
| | | 0 | Disabled. The I ² C Master function is disabled. | |
| | | 1 | Enabled. The I ² C Master function is enabled. | |
| 1 | SLVEN | | Slave Enable. When disabled, configurations settings for the Slave function are not changed, but the Slave function is internally reset. | 0 |
| | | 0 | Disabled. The I ² C slave function is disabled. | |
| | | 1 | Enabled. The I ² C slave function is enabled. | |
| 2 | MONEN | | Monitor Enable. When disabled, configurations settings for the Monitor function are not changed, but the Monitor function is internally reset. | 0 |
| | | 0 | Disabled. The I ² C monitor function is disabled. | |
| | | 1 | Enabled. The I ² C monitor function is enabled. | |
| 3 | TIMEOUTEN | | I ² C bus Time-out Enable. When disabled, the time-out function is internally reset. | 0 |
| | | 0 | Disabled. Time-out function is disabled. | |
| | | 1 | Enabled. Time-out function is enabled. Both types of time-out flags will be generated and will cause interrupts if they are enabled. Typically, only one time-out will be used in a system. | |
| 4 | MONCLKSTR | | Monitor function Clock Stretching. | 0 |
| | | 0 | Disabled. The monitor function will not perform clock stretching. Software or DMA may not always be able to read data provided by the monitor function before it is overwritten. This mode may be used when non-invasive monitoring is critical. | |
| | | 1 | Enabled. The monitor function will perform clock stretching in order to ensure that software or DMA can read all incoming data supplied by the monitor function. | |

Table 339. I2C Configuration register (CFG, address offset 0x000) bit description

| Bit | Symbol | Value | Description | Reset Value |
|------|-----------|-------|---|-------------|
| 5 | HSCAPABLE | | High-speed mode Capable enable. Since High Speed mode alters the way I2C pins drive and filter, as well as the timing for certain I ² C signalling, enabling High-speed mode applies to all functions: master, slave, and monitor. | 0 |
| | | 0 | Fast-mode plus. The I2C block will support Standard-mode, Fast-mode, and Fast-mode Plus, to the extent that the pin electronics support these modes. Any changes that need to be made to the pin controls, such as changing the drive strength or filtering, must be made by software via the IOCON register associated with each I ² C pin, | |
| | | 1 | High-speed. In addition to Standard-mode, Fast-mode, and Fast-mode Plus, the I ² C block will support High-speed mode to the extent that the pin electronics support these modes. See Section 24.7.1.2 for more information. | |
| 31:6 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

24.6.2 I2C Status register

The STAT register provides status flags and state information about all of the functions of the I²C block. Access to bits in this register varies. RO = Read-only, W1 = write 1 to clear.

Details on the master and slave states described in the MSTSTATE and SLVSTATE bits in this register are listed in [Table 342](#) and [Table 343](#).

Table 340. Address map STAT register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| I2C0 | 0x4009 4000 | 0x004 | - | 1 |
| I2C1 | 0x4009 8000 | 0x004 | - | 1 |
| I2C2 | 0x4009 C000 | 0x004 | - | 1 |

Table 341. I²C Status register (STAT, address offset 0x004) bit description

| Bit | Symbol | Value | Description | Reset value | Access |
|-----|-------------|-------|---|-------------|--------|
| 0 | MST PENDING | | Master Pending. Indicates that the Master is waiting to continue communication on the I ² C-bus (pending) or is idle. When the master is pending, the MSTSTATE bits indicate what type of software service if any the master expects. This flag will cause an interrupt when set if, enabled via the INTENSET register. The MSTPENDING flag is not set when the DMA is handling an event (if the MSTDMA bit in the MSTCTL register is set). If the master is in the idle state, and no communication is needed, mask this interrupt. | 1 | RO |
| | | 0 | In progress. Communication is in progress and the Master function is busy and cannot currently accept a command. | | |
| | | 1 | Pending. The Master function needs software service or is in the idle state. If the master is not in the idle state, it is waiting to receive or transmit data or the NACK bit. | | |
| 3:1 | MSTSTATE | | Master State code. The master state code reflects the master state when the MSTPENDING bit is set, that is the master is pending or in the idle state. Each value of this field indicates a specific required service for the Master function. All other values are reserved. See Table 342 for details of state values and appropriate responses. | 0 | RO |
| | | 0x0 | Idle. The Master function is available to be used for a new transaction. | | |
| | | 0x1 | Receive ready. Received data available (Master Receiver mode). Address plus Read was previously sent and Acknowledged by slave. | | |
| | | 0x2 | Transmit ready. Data can be transmitted (Master Transmitter mode). Address plus Write was previously sent and Acknowledged by slave. | | |
| | | 0x3 | NACK Address. Slave NACKed address. | | |
| | | 0x4 | NACK Data. Slave NACKed transmitted data. | | |
| 4 | MST ARBLOSS | | Master Arbitration Loss flag. This flag can be cleared by software writing a 1 to this bit. It is also cleared automatically a 1 is written to MSTCONTINUE. | 0 | W1 |
| | | 0 | No Arbitration Loss has occurred. | | |
| | | 1 | Arbitration loss. The Master function has experienced an Arbitration Loss. At this point, the Master function has already stopped driving the bus and gone to an idle state. Software can respond by doing nothing, or by sending a Start in order to attempt to gain control of the bus when it next becomes idle. | | |
| 5 | - | | Reserved. Read value is undefined, only zero should be written. | NA | NA |

Table 341. I²C Status register (STAT, address offset 0x004) bit description ...continued

| Bit | Symbol | Value | Description | Reset value | Access |
|------|-----------------|-------|--|-------------|--------|
| 6 | MST STSTPERR | | Master Start/Stop Error flag. This flag can be cleared by software writing a 1 to this bit. It is also cleared automatically a 1 is written to MSTCONTINUE. | 0 | W1 |
| | | 0 | No Start/Stop Error has occurred. | | |
| | | 1 | The Master function has experienced a Start/Stop Error. A Start or Stop was detected at a time when it is not allowed by the I ² C specification. The Master interface has stopped driving the bus and gone to an idle state, no action is required. A request for a Start could be made, or software could attempt to insure that the bus has not stalled. | | |
| 7 | - | | Reserved. Read value is undefined, only zero should be written. | NA | NA |
| 8 | SLV PENDING | | Slave Pending. Indicates that the Slave function is waiting to continue communication on the I ² C-bus and needs software service. This flag will cause an interrupt when set if enabled via INTENSET. The SLVPENDING flag is not set when the DMA is handling an event (if the SLVDMA bit in the SLVCTL register is set). The SLVPENDING flag is read-only and is automatically cleared when a 1 is written to the SLVCONTINUE bit in the SLVCTL register. The point in time when SlvPending is set depends on whether the I ² C block is in HSCAPABLE mode. See Section 24.7.1.2.2 . When the I ² C block is configured to be HSCAPABLE, HS master codes are detected automatically. Due to the requirements of the HS I ² C specification, slave addresses must also be detected automatically, since the address must be acknowledged before the clock can be stretched. | 0 | RO |
| | | 0 | In progress. The Slave function does not currently need service. | | |
| | | 1 | Pending. The Slave function needs service. Information on what is needed can be found in the adjacent SLVSTATE field. | | |
| 10:9 | SLVSTATE | | Slave State code. Each value of this field indicates a specific required service for the Slave function. All other values are reserved. See Table 343 for state values and actions. | 0 | RO |
| | | 0x0 | Slave address. Address plus R/W received. At least one of the four slave addresses has been matched by hardware. | | |
| | | 0x1 | Slave receive. Received data is available (Slave Receiver mode). | | |
| | | 0x2 | Slave transmit. Data can be transmitted (Slave Transmitter mode). | | |
| 11 | SLV NOTSTR | | Slave Not Stretching. Indicates when the slave function is stretching the I ² C clock. This is needed in order to gracefully invoke Deep Sleep or Power-down modes during slave operation. This read-only flag reflects the slave function status in real time. | 1 | RO |
| | | 0 | Stretching. The slave function is currently stretching the I ² C bus clock. Deep-Sleep or Power-down mode cannot be entered at this time. | | |
| | | 1 | Not stretching. The slave function is not currently stretching the I ² C bus clock. Deep-sleep or Power-down mode could be entered at this time. | | |

Table 341. I²C Status register (STAT, address offset 0x004) bit description ...continued

| Bit | Symbol | Value | Description | Reset value | Access |
|-------|------------|-------|---|-------------|--------|
| 13:12 | SLVIDX | | Slave address match Index. This field is valid when the I ² C slave function has been selected by receiving an address that matches one of the slave addresses defined by any enabled slave address registers, and provides an identification of the address that was matched. It is possible that more than one address could be matched, but only one match can be reported here. | 0 | RO |
| | | 0x0 | Address 0. Slave address 0 was matched. | | |
| | | 0x1 | Address 1. Slave address 1 was matched. | | |
| | | 0x2 | Address 2. Slave address 2 was matched. | | |
| | | 0x3 | Address 3. Slave address 3 was matched. | | |
| 14 | SLVSEL | | Slave selected flag. SLVSEL is set after an address match when software tells the Slave function to acknowledge the address. It is cleared when another address cycle presents an address that does not match an enabled address on the Slave function, when slave software decides to NACK a matched address, or when there is a Stop detected on the bus. SLVSEL is not cleared if software NACKs data. | 0 | RO |
| | | 0 | Not selected. The Slave function is not currently selected. | | |
| | | 1 | Selected. The Slave function is currently selected. | | |
| 15 | SLVDESEL | | Slave Deselected flag. This flag will cause an interrupt when set if enabled via INTENSET. This flag can be cleared by writing a 1 to this bit. | 0 | W1 |
| | | 0 | Not deselected. The Slave function has not become deselected. This does not mean that it is currently selected. That information can be found in the SLVSEL flag. | | |
| | | 1 | Deselected. The Slave function has become deselected. This is specifically caused by the SLVSEL flag changing from 1 to 0. See the description of SLVSEL for details on when that event occurs. | | |
| 16 | MONRDY | | Monitor Ready. This flag is cleared when the MONRXDAT register is read. | 0 | RO |
| | | 0 | No data. The Monitor function does not currently have data available. | | |
| | | 1 | Data waiting. The Monitor function has data waiting to be read. | | |
| 17 | MONOV | | Monitor Overflow flag. | 0 | W1 |
| | | 0 | No overrun. Monitor data has not overrun. | | |
| | | 1 | Overrun. A Monitor data overrun has occurred. This can only happen when Monitor clock stretching not enabled via the MONCLKSTR bit in the CFG register. Writing 1 to this bit clears the flag. | | |
| 18 | MON ACTIVE | | Monitor Active flag. Indicates when the Monitor function considers the I ² C bus to be active. Active is defined here as when some Master is on the bus: a bus Start has occurred more recently than a bus Stop. | 0 | RO |
| | | 0 | Inactive. The Monitor function considers the I ² C bus to be inactive. | | |
| | | 1 | Active. The Monitor function considers the I ² C bus to be active. | | |
| 19 | MONIDLE | | Monitor Idle flag. This flag is set when the Monitor function sees the I ² C bus change from active to inactive. This can be used by software to decide when to process data accumulated by the Monitor function. This flag will cause an interrupt when set if enabled via the INTENSET register. The flag can be cleared by writing a 1 to this bit. | 0 | W1 |
| | | 0 | Not idle. The I ² C bus is not idle, or this flag has been cleared by software. | | |
| | | 1 | Idle. The I ² C bus has gone idle at least once since the last time this flag was cleared by software. | | |

Table 341. I²C Status register (STAT, address offset 0x004) bit description ...continued

| Bit | Symbol | Value | Description | Reset value | Access |
|-------|------------------|-------|---|-------------|--------|
| 23:20 | - | | Reserved. Read value is undefined, only zero should be written. | NA | NA |
| 24 | EVENT TIMEOUT | | Event Time-out Interrupt flag. Indicates when the time between events has been longer than the time specified by the TIMEOUT register. Events include Start, Stop, and clock edges. The flag is cleared by writing a 1 to this bit. No time-out is created when the I ² C-bus is idle. | 0 | W1 |
| | | 0 | No time-out. I ² C bus events have not caused a time-out. | | |
| | | 1 | Event time-out. The time between I ² C bus events has been longer than the time specified by the I2C TIMEOUT register. | | |
| 25 | SCL TIMEOUT | | SCL Time-out Interrupt flag. Indicates when SCL has remained low longer than the time specific by the TIMEOUT register. The flag is cleared by writing a 1 to this bit. | 0 | W1 |
| | | 0 | No time-out. SCL low time has not caused a time-out. | | |
| | | 1 | Time-out. SCL low time has caused a time-out. | | |
| 31:26 | - | | Reserved. Read value is undefined, only zero should be written. | NA | NA |

Table 342. Master function state codes (MSTSTATE)

| MST STATE | Description | Actions | DMA allowed |
|-----------|---|--|-------------|
| 0x0 | Idle. The Master function is available to be used for a new transaction. | Send a Start or disable MSTPENDING interrupt if the Master function is not needed currently. | No |
| 0x1 | Received data is available (Master Receiver mode). Address plus Read was previously sent and Acknowledged by slave. | Read data and either continue, send a Stop, or send a Repeated Start. | Yes |
| 0x2 | Data can be transmitted (Master Transmitter mode). Address plus Write was previously sent and Acknowledged by slave. | Send data and continue, or send a Stop or Repeated Start. | Yes |
| 0x3 | Slave NACKed address. | Send a Stop or Repeated Start. | No |
| 0x4 | Slave NACKed transmitted data. | Send a Stop or Repeated Start. | No |

Table 343. Slave function state codes (SLVSTATE)

| SLVSTATE | Description | Actions | DMA allowed |
|----------|---|---|-------------|
| 0 | SLVST_ADDR Address plus R/W received. At least one of the 4 slave addresses has been matched by hardware. | Software can further check the address if needed, for instance if a subset of addresses qualified by SLVQUAL0 is to be used. Software can ACK or NACK the address by writing 1 to either SLVCONTINUE or SLVNACK. Also see Section 24.7.3 regarding 10-bit addressing. | No |
| 1 | SLVST_RX Received data is available (Slave Receiver mode). | Read data, reply with an ACK or a NACK. | Yes |
| 2 | SLVST_TX Data can be transmitted (Slave Transmitter mode). | Send data. Note that when the Master NACKs data transmitted by the slave, the slave becomes de-selected. | Yes |

24.6.3 Interrupt Enable Set and read register

The INTENSET register controls which I²C status flags generate interrupts. Writing a 1 to a bit position in this register enables an interrupt in the corresponding position in the STAT register (Table 341), if an interrupt is supported there. Reading INTENSET indicates which interrupts are currently enabled.

Table 344. Address map INTENSET register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| I2C0 | 0x4009 4000 | 0x008 | - | 1 |
| I2C1 | 0x4009 8000 | 0x008 | - | 1 |
| I2C2 | 0x4009 C000 | 0x008 | - | 1 |

Table 345. Interrupt Enable Set and read register (INTENSET, address offset 0x008) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|---------------|-------|---|-------------|
| 0 | MSTPENDINGEN | | Master Pending interrupt Enable. | 0 |
| | | 0 | Disabled. The MstPending interrupt is disabled. | |
| | | 1 | Enabled. The MstPending interrupt is enabled. | |
| 3:1 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 4 | MSTARBLOSSEN | | Master Arbitration Loss interrupt Enable. | 0 |
| | | 0 | Disabled. The MstArbLoss interrupt is disabled. | |
| | | 1 | Enabled. The MstArbLoss interrupt is enabled. | |
| 5 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 6 | MSTSTSTPERREN | | Master Start/Stop Error interrupt Enable. | 0 |
| | | 0 | Disabled. The MstStStpErr interrupt is disabled. | |
| | | 1 | Enabled. The MstStStpErr interrupt is enabled. | |
| 7 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 8 | SLVPENDINGEN | | Slave Pending interrupt Enable. | 0 |
| | | 0 | Disabled. The SlvPending interrupt is disabled. | |
| | | 1 | Enabled. The SlvPending interrupt is enabled. | |
| 10:9 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 11 | SLVNOTSTREN | | Slave Not Stretching interrupt Enable. | 0 |
| | | 0 | Disabled. The SlvNotStr interrupt is disabled. | |
| | | 1 | Enabled. The SlvNotStr interrupt is enabled. | |
| 14:12 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 15 | SLVDESELEN | | Slave Deselect interrupt Enable. | 0 |
| | | 0 | Disabled. The SlvDeSel interrupt is disabled. | |
| | | 1 | Enabled. The SlvDeSel interrupt is enabled. | |
| 16 | MONRDYEN | | Monitor data Ready interrupt Enable. | 0 |
| | | 0 | Disabled. The MonRdy interrupt is disabled. | |
| | | 1 | Enabled. The MonRdy interrupt is enabled. | |
| 17 | MONOVEN | | Monitor Overrun interrupt Enable. | 0 |
| | | 0 | Disabled. The MonOv interrupt is disabled. | |
| | | 1 | Enabled. The MonOv interrupt is enabled. | |
| 18 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

Table 345. Interrupt Enable Set and read register (INTENSET, address offset 0x008) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|----------------|-------|---|-------------|
| 19 | MONIDLEEN | | Monitor Idle interrupt Enable. | 0 |
| | | 0 | Disabled. The MonIdle interrupt is disabled. | |
| | | 1 | Enabled. The MonIdle interrupt is enabled. | |
| 23:20 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 24 | EVENTTIMEOUTEN | | Event time-out interrupt Enable. | 0 |
| | | 0 | Disabled. The Event time-out interrupt is disabled. | |
| | | 1 | Enabled. The Event time-out interrupt is enabled. | |
| 25 | SCLTIMEOUTEN | | SCL time-out interrupt Enable. | 0 |
| | | 0 | Disabled. The SCL time-out interrupt is disabled. | |
| | | 1 | Enabled. The SCL time-out interrupt is enabled. | |
| 31:26 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

24.6.4 Interrupt Enable Clear register

Writing a 1 to a bit position in INTENCLR clears the corresponding position in the INTENSET register, disabling that interrupt. INTENCLR is a write-only register.

Bits that do not correspond to defined bits in INTENSET are reserved and only zeroes should be written to them.

Table 346. Address map INTENCLR register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| I2C0 | 0x4009 4000 | 0x00C | - | 1 |
| I2C1 | 0x4009 8000 | 0x00C | - | 1 |
| I2C2 | 0x4009 C000 | 0x00C | - | 1 |

Table 347. Interrupt Enable Clear register (INTENCLR, address offset 0x00C) bit description

| Bit | Symbol | Description | Reset value |
|-------|---------------|---|-------------|
| 0 | MSTPENDINGCLR | Master Pending interrupt clear. Writing 1 to this bit clears the corresponding bit in the INTENSET register if implemented. | 0 |
| 3:1 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 4 | MSTARLOSSCLR | Master Arbitration Loss interrupt clear. | 0 |
| 5 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 6 | MSTSTSPERRCLR | Master Start/Stop Error interrupt clear. | 0 |
| 7 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 8 | SLVPENDINGCLR | Slave Pending interrupt clear. | 0 |
| 10:9 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 11 | SLVNOTSTRCLR | Slave Not Stretching interrupt clear. | 0 |
| 14:12 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 15 | SLVDESELCLR | Slave Deselect interrupt clear. | 0 |
| 16 | MONRDYCLR | Monitor data Ready interrupt clear. | 0 |
| 17 | MONOVCLR | Monitor Overrun interrupt clear. | 0 |
| 18 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 19 | MONIDLECLR | Monitor Idle interrupt clear. | 0 |

Table 347. Interrupt Enable Clear register (INTENCLR, address offset 0x00C) bit description ...continued

| Bit | Symbol | Description | Reset value |
|-------|-----------------|---|-------------|
| 23:20 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 24 | EVENTTIMEOUTCLR | Event time-out interrupt clear. | 0 |
| 25 | SCLTIMEOUTCLR | SCL time-out interrupt clear. | 0 |
| 31:26 | - | Reserved. Read value is undefined, only zero should be written. | NA |

24.6.5 Time-out value register

The TIMEOUT register allows setting an upper limit to certain I²C bus times, informing by status flag and/or interrupt when those times are exceeded.

Two time-outs are generated, and software can elect to use either of them.

1. EVENTTIMEOUT checks the time between bus events while the bus is not idle: Start, SCL rising, SCL falling, and Stop. The EVENTTIMEOUT status flag in the STAT register is set if the time between any two events becomes longer than the time configured in the TIMEOUT register. The EVENTTIMEOUT status flag can cause an interrupt if enabled to do so by the EVENTTIMEOUTEN bit in the INTENSET register.
2. SCLTIMEOUT checks only the time that the SCL signal remains low while the bus is not idle. The SCLTIMEOUT status flag in the STAT register is set if SCL remains low longer than the time configured in the TIMEOUT register. The SCLTIMEOUT status flag can cause an interrupt if enabled to do so by the SCLTIMEOUTEN bit in the INTENSET register. The SCLTIMEOUT can be used with the SMBus.

Also see [Section 24.7.2 "Time-out"](#).

Table 348. Address map TIMEOUT register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| I2C0 | 0x4009 4010 | 0x010 | - | 1 |
| I2C1 | 0x4009 8010 | 0x010 | - | 1 |
| I2C2 | 0x4009 C010 | 0x010 | - | 1 |

Table 349. Time-out value register (TIMEOUT, address offset 0x010) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|--|-------------|
| 3:0 | TOMIN | Time-out time value, bottom four bits. These are hard-wired to 0xF. This gives a minimum time-out of 16 I ² C function clocks and also a time-out resolution of 16 I ² C function clocks. | 0xF |
| 15:4 | TO | Time-out time value. Specifies the time-out interval value in increments of 16 I ² C function clocks, as defined by the CLKDIV register. To change this value while I ² C is in operation, disable all time-outs, write a new value to TIMEOUT, then re-enable time-outs. 0x000 = A time-out will occur after 16 counts of the I ² C function clock. 0x001 = A time-out will occur after 32 counts of the I ² C function clock. ... 0xFFFF = A time-out will occur after 65,536 counts of the I ² C function clock. | 0xFFFF |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | NA |

24.6.6 Clock Divider register

The CLKDIV register divides down the Peripheral Clock (PCLK) to produce the I²C function clock that is used to time various aspects of the I²C interface. The I²C function clock is used for some internal operations in the I²C block and to generate the timing required by the I²C bus specification, some of which are user configured in the MSTIME register for Master operation. Slave operation uses CLKDIV for some timing functions.

See [Section 24.7.1.1 “Rate calculations”](#) for details on bus rate setup.

Table 350. Address map CLDIV register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| I2C0 | 0x4009 4010 | 0x014 | - | 1 |
| I2C1 | 0x4009 8010 | 0x014 | - | 1 |
| I2C2 | 0x4009 C010 | 0x014 | - | 1 |

Table 351. I²C Clock Divider register (CLKDIV, offset 0x14) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 15:0 | DIVVAL | This field controls how the clock (PCLK) is used by the I ² C functions that need an internal clock in order to operate. 0x0000 = PCLK is used directly by the I ² C. 0x0001 = PCLK is divided by 2 before use. 0x0002 = PCLK is divided by 3 before use. ... 0xFFFF = PCLK is divided by 65,536 before use. | 0 |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | NA |

24.6.7 Interrupt Status register

The INTSTAT register provides register provides a view of those interrupt flags that are currently enabled. This can simplify software handling of interrupts. See [Table 341](#) for detailed descriptions of the interrupt flags.

Table 352. Address map INTSTAT register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| I2C0 | 0x4009 4010 | 0x018 | - | 1 |
| I2C1 | 0x4009 8010 | 0x018 | - | 1 |
| I2C2 | 0x4009 C010 | 0x018 | - | 1 |

Table 353. I²C Interrupt Status register (INTSTAT, address offset 0x018) bit description

| Bit | Symbol | Description | Reset value |
|-----|------------|---|-------------|
| 0 | MSTPENDING | Master Pending. | 1 |
| 3:1 | - | Reserved. | |
| 4 | MSTARBLOSS | Master Arbitration Loss flag. | 0 |
| 5 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 6 | MSTSTSPERR | Master Start/Stop Error flag. | 0 |
| 7 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 8 | SLVPENDING | Slave Pending. | 0 |

Table 353. I²C Interrupt Status register (INTSTAT, address offset 0x018) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------------|---|-------------|
| 10:9 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 11 | SLVNOTSTR | Slave Not Stretching status. | 1 |
| 14:12 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 15 | SLVDESEL | Slave Deselected flag. | 0 |
| 16 | MONRDY | Monitor Ready. | 0 |
| 17 | MONOV | Monitor Overflow flag. | 0 |
| 18 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 19 | MONIDLE | Monitor Idle flag. | 0 |
| 23:20 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 24 | EVENTTIMEOUT | Event time-out Interrupt flag. | 0 |
| 25 | SCLTIMEOUT | SCL time-out Interrupt flag. | 0 |
| 31:26 | - | Reserved. Read value is undefined, only zero should be written. | NA |

24.6.8 Master Control register

The MSTCTL register contains bits that control various functions of the I²C Master interface. Only write to this register when the master is pending (MSTPENDING = 1 in the STAT register, [Table 341](#)).

Software should always write a complete value to MSTCTL, and not OR new control bits into the register as is possible in other registers such as CFG. This is due to the fact that MSTSTART and MSTSTOP are not self-clearing flags. ORing in new data following a Start or Stop may cause undesirable side effects.

After an initial I²C Start, MSTCTL should generally only be written when the MSTPENDING flag in the STAT register is set, after the last bus operation has completed. An exception is when DMA is being used and a transfer completes. In this case there is no MSTPENDING flag, and the MSTDMA control bit would be cleared by software potentially at the same time as setting either the MSTSTOP or MSTSTART control bit.

Remark: When in the idle or slave NACKed states (see [Table 342](#)), set the MSTDMA bit either with or after the MSTCONTINUE bit. MSTDMA can be cleared at any time.

Table 354. Address map MSTCTL register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| I2C0 | 0x4009 4010 | 0x020 | - | 1 |
| I2C1 | 0x4009 8010 | 0x020 | - | 1 |
| I2C2 | 0x4009 C010 | 0x020 | - | 1 |

Table 355. Master Control register (MSTCTL, address offset 0x020) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|-------------|-------|---|-------------|
| 0 | MSTCONTINUE | | Master Continue. This bit is write-only. | 0 |
| | | 0 | No effect. | |
| | | 1 | Continue. Informs the Master function to continue to the next operation. This must be done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation. | |
| 1 | MSTSTART | | Master Start control. This bit is write-only. | 0 |
| | | 0 | No effect. | |
| | | 1 | Start. A Start will be generated on the I ² C bus at the next allowed time. | |
| 2 | MSTSTOP | | Master Stop control. This bit is write-only. | 0 |
| | | 0 | No effect. | |
| | | 1 | Stop. A Stop will be generated on the I ² C bus at the next allowed time, preceded by a NACK to the slave if the master is receiving data from the slave (Master Receiver mode). | |

Table 355. Master Control register (MSTCTL, address offset 0x020) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|--------|-------|--|-------------|
| 3 | MSTDMA | | Master DMA enable. Data operations of the I ² C can be performed with DMA. Protocol type operations such as Start, address, Stop, and address match must always be done with software, typically via an interrupt. When a DMA data transfer is complete, MSTDMA must be cleared prior to beginning the next operation, typically a Start or Stop. This bit is read/write. | 0 |
| | | 0 | Disable. No DMA requests are generated for master operation. | |
| | | 1 | Enable. A DMA request is generated for I ² C master data operations. When this I ² C master is generating Acknowledge bits in Master Receiver mode, the acknowledge is generated automatically. | |
| 31:4 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

24.6.9 Master Time register

The MSTTIME register allows programming of certain times that may be controlled by the Master function. These include the clock (SCL) high and low times, repeated Start setup time, and transmitted data setup time.

The I²C clock pre-divider is described in [Table 351](#).

Table 356. Address map MSTTIME register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| I2C0 | 0x4009 4010 | 0x024 | - | 1 |
| I2C1 | 0x4009 8010 | 0x024 | - | 1 |
| I2C2 | 0x4009 C010 | 0x024 | - | 1 |

Table 357. Master Time register (MSTTIME, address offset 0x024) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|----------|-------|---|-------------|
| 2:0 | MSTSCLOW | | Master SCL Low time. Specifies the minimum low time that will be asserted by this master on SCL. Other devices on the bus (masters or slaves) could lengthen this time. This corresponds to the parameter t_{LOW} in the I ² C bus specification. I ² C bus specification parameters t_{BUF} and $t_{SU;STA}$ have the same values and are also controlled by MSTSCLOW. | 0 |
| | | 0x0 | 2 clocks. Minimum SCL low time is 2 clocks of the I ² C clock pre-divider. | |
| | | 0x1 | 3 clocks. Minimum SCL low time is 3 clocks of the I ² C clock pre-divider. | |
| | | 0x2 | 4 clocks. Minimum SCL low time is 4 clocks of the I ² C clock pre-divider. | |
| | | 0x3 | 5 clocks. Minimum SCL low time is 5 clocks of the I ² C clock pre-divider. | |
| | | 0x4 | 6 clocks. Minimum SCL low time is 6 clocks of the I ² C clock pre-divider. | |
| | | 0x5 | 7 clocks. Minimum SCL low time is 7 clocks of the I ² C clock pre-divider. | |
| | | 0x6 | 8 clocks. Minimum SCL low time is 8 clocks of the I ² C clock pre-divider. | |
| | | 0x7 | 9 clocks. Minimum SCL low time is 9 clocks of the I ² C clock pre-divider. | |
| 3 | - | | Reserved. | 0 |

Table 357. Master Time register (MSTTIME, address offset 0x024) bit description ...continued

| Bit | Symbol | Value | Description | Reset value |
|------|------------|-------|--|-------------|
| 6:4 | MSTSCLHIGH | | Master SCL High time. Specifies the minimum high time that will be asserted by this master on SCL. Other masters in a multi-master system could shorten this time. This corresponds to the parameter t_{HIGH} in the I ² C bus specification. I ² C bus specification parameters $t_{SU;STO}$ and $t_{HD;STA}$ have the same values and are also controlled by MSTSCLHIGH. | 0 |
| | | 0x0 | 2 clocks. Minimum SCL high time is 2 clock of the I ² C clock pre-divider. | |
| | | 0x1 | 3 clocks. Minimum SCL high time is 3 clocks of the I ² C clock pre-divider . | |
| | | 0x2 | 4 clocks. Minimum SCL high time is 4 clock of the I ² C clock pre-divider. | |
| | | 0x3 | 5 clocks. Minimum SCL high time is 5 clock of the I ² C clock pre-divider. | |
| | | 0x4 | 6 clocks. Minimum SCL high time is 6 clock of the I ² C clock pre-divider. | |
| | | 0x5 | 7 clocks. Minimum SCL high time is 7 clock of the I ² C clock pre-divider. | |
| | | 0x6 | 8 clocks. Minimum SCL high time is 8 clock of the I ² C clock pre-divider. | |
| | | 0x7 | 9 clocks. Minimum SCL high time is 9 clocks of the I ² C clock pre-divider. | |
| 31:7 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

24.6.10 Master Data register

The MSTDAT register provides the means to read the most recently received data for the Master function, and to transmit data using the Master function.

Table 358. Address map MSTDAT register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| I2C0 | 0x4009 4010 | 0x028 | - | 1 |
| I2C1 | 0x4009 8010 | 0x028 | - | 1 |
| I2C2 | 0x4009 C010 | 0x028 | - | 1 |

Table 359. Master Data register (MSTDAT, address offset 0x028) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|--|-------------|
| 7:0 | DATA | Master function data register. Read: read the most recently received data for the Master function. Write: transmit data using the Master function. | 0 |
| 31:8 | - | Reserved. Read value is undefined, only zero should be written. | NA |

24.6.11 Slave Control register

The SLVCTL register contains bits that control various functions of the I²C Slave interface. Only write to this register when the slave is pending (SLVPENDING = 1 in the STAT register, [Table 341](#)).

Remark: When in the slave address state (slave state 0, see [Table 343](#)), set the SLVDMA bit either with or after the SLVCONTINUE bit. SLVDMA can be cleared at any time.

Table 360. Address map SLVCTL register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| I2C0 | 0x4009 4010 | 0x010 | - | 1 |
| I2C1 | 0x4009 8010 | 0x010 | - | 1 |
| I2C2 | 0x4009 C010 | 0x010 | - | 1 |

Table 361. Slave Control register (SLVCTL, address offset 0x040) bit description

| Bit | Symbol | Value | Description | Reset Value |
|------|-------------|-------|---|-------------|
| 0 | SLVCONTINUE | | Slave Continue. | 0 |
| | | 0 | No effect. | |
| | | 1 | Continue. Informs the Slave function to continue to the next operation, by clearing the SLVPENDING flag in the STAT register. This must be done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation. SLVCONTINUE should not be set unless SLVPENDING = 1. | |
| 1 | SLVNACK | | Slave NACK. | 0 |
| | | 0 | No effect. | |
| | | 1 | NACK. Causes the Slave function to NACK the master when the slave is receiving data from the master (Slave Receiver mode). | |
| 3 | SLVDMA | | Slave DMA enable. | 0 |
| | | 0 | Disabled. No DMA requests are issued for Slave mode operation. | |
| | | 1 | Enabled. DMA requests are issued for I ² C slave data transmission and reception. | |
| 31:4 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

24.6.12 Slave Data register

The SLVDAT register provides the means to read the most recently received data for the Slave function and to transmit data using the Slave function.

Table 362. Address map SLVDAT register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| I2C0 | 0x4009 4010 | 0x044 | - | 1 |
| I2C1 | 0x4009 8010 | 0x044 | - | 1 |
| I2C2 | 0x4009 C010 | 0x044 | - | 1 |

Table 363. Slave Data register (SLVDAT, address offset 0x044) bit description

| Bit | Symbol | Description | Reset Value |
|------|--------|---|-------------|
| 7:0 | DATA | Slave function data register. Read: read the most recently received data for the Slave function. Write: transmit data using the Slave function. | 0 |
| 31:8 | - | Reserved. Read value is undefined, only zero should be written. | NA |

24.6.13 Slave Address registers

The four SLVADR registers each allow enabling and defining one of the addresses that can be automatically recognized by the I²C slave hardware.

For SLVADR0, the comparison to the receive address can be affected by the setting of the SLVQUAL0 register (see [Section 24.6.14](#)). The other 3 address comparators do not include the address qualifier feature. For handling of the general call address, one of the 4 address registers can be programmed to respond to address 0.

Table 364. Address map SLVADR[0:3] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| I2C0 | 0x4009 4010 | [0x048:0x054] | 0x4 | 4 |
| I2C1 | 0x4009 8010 | [0x048:0x054] | 0x4 | 4 |
| I2C2 | 0x4009 C010 | [0x048:0x054] | 0x4 | 4 |

Table 365. Slave Address registers (SLVADR[0:3], address offset [0x048:0x054]) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|-----------|-------|---|-------------|
| 0 | SADISABLE | | Slave Address n Disable. | 1 |
| | | 0 | Enabled. Slave Address n is enabled. | |
| | | 1 | Ignored Slave Address n is ignored. | |
| 7:1 | SLVADR | | Slave Address. Seven bit slave address that is compared to received addresses if enabled. | 0 |
| 31:8 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

24.6.14 Slave address Qualifier 0 register

The SLVQUAL0 register can alter how Slave Address 0 (specified by the SLVADR0 register) is interpreted.

Table 366. Address map SLVQUAL0 register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| I2C0 | 0x4009 4010 | 0x058 | - | 1 |
| I2C1 | 0x4009 8010 | 0x058 | - | 1 |
| I2C2 | 0x4009 C010 | 0x058 | - | 1 |

Table 367. Slave address Qualifier 0 register (SLVQUAL0, address offset 0x058) bit description

| Bit | Symbol | Value | Description | Reset Value |
|------|-----------|-------|---|-------------|
| 0 | QUALMODE0 | | Qualify mode for slave address 0. | 0 |
| | | 0 | Mask. The SLVQUAL0 field is used as a logical mask for matching address 0. | |
| | | 1 | Extend. The SLVQUAL0 field is used to extend address 0 matching in a range of addresses. | |
| 7:1 | SLVQUAL0 | | Slave address Qualifier for address 0. A value of 0 causes the address in SLVADR0 to be used as-is, assuming that it is enabled. If QUALMODE0 = 0, any bit in this field which is set to 1 will cause an automatic match of the corresponding bit of the received address when it is compared to the SLVADR0 register. If QUALMODE0 = 1, an address range is matched for address 0. This range extends from the value defined by SLVADR0 to the address defined by SLVQUAL0 (address matches when $SLVADR0[7:1] \leq \text{received address} \leq SLVQUAL0[7:1]$). | 0 |
| 31:8 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

24.6.15 Monitor data register

The read-only MONRXDAT register provides information about events on the I²C bus, primarily to facilitate debugging of the I²C during application development. All data addresses and data passing on the bus and whether these were acknowledged, as well as Start and Stop events, are reported.

The Monitor function must be enabled by the MONEN bit in the CFG register. Monitor mode can be configured to stretch the I²C clock if data is not read from the MONRXDAT register in time to prevent it, via the MONCLKSTR bit in the CFG register. This can help ensure that nothing is missed but can cause the monitor function to be somewhat intrusive (by potentially adding clock delays, depending on software or DMA response time). In order to improve the chance of collecting all Monitor information if clock stretching is not enabled, Monitor data is buffered such that it is available until the end of the next piece of information from the I²C bus.

Details of clock stretching are different in HS mode, see [Section 24.7.1.2.2](#).

Table 368. Address map MONRXDAT register

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|--------|-----------|-----------|
| I2C0 | 0x4009 4010 | 0x080 | - | 1 |
| I2C1 | 0x4009 8010 | 0x080 | - | 1 |
| I2C2 | 0x4009 C010 | 0x080 | - | 1 |

Table 369. Monitor data register (MONRXDAT, address offset 0x080) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|------------|-------|--|-------------|
| 7:0 | MONRXDAT | | Monitor function Receiver Data. This reflects every data byte that passes on the I ² C pins. | 0 |
| 8 | MONSTART | 0 | No start detected. The monitor function has not detected a Start event on the I ² C bus. | 0 |
| | | 1 | Start detected. The monitor function has detected a Start event on the I ² C bus. | |
| 9 | MONRESTART | 0 | No repeated start detected. The monitor function has not detected a Repeated Start event on the I ² C bus. | 0 |
| | | 1 | Repeated start detected. The monitor function has detected a Repeated Start event on the I ² C bus. | |
| 10 | MONNACK | 0 | Acknowledged. The data currently being provided by the monitor function was acknowledged by at least one master or slave receiver. | 0 |
| | | 1 | Not acknowledged. The data currently being provided by the monitor function was not acknowledged by any receiver. | |
| 31:11 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

24.7 Functional description

24.7.1 Bus rates and timing considerations

Due to the nature of the I²C bus, it is generally not possible to guarantee a specific clock rate on the SCL pin. On the I²C-bus, the clock can be stretched by any slave device, extended by software overhead time, etc.

In a multi-master system, the master that provides the shortest SCL high time will cause that time to appear on SCL as long as that master is participating in I²C traffic (i.e. when it is the only master on the bus, or during arbitration between masters).

In addition, I²C implementations generally base subsequent actions on what actually happens on the bus lines. For instance, a bus master allows SCL to go high. It then monitors the line to make sure it actually did go high (this would be required in a multi-master system). This results in a small delay before the next action on the bus, caused by the rise time of the open drain bus line.

Rate calculations give a base frequency that represents the fastest that the I²C bus could operate if nothing slows it down.

24.7.1.1 Rate calculations

Master timing

SCL high time (in I²C function clocks) = $(CLKDIV + 1) * (MSTSCLEHIGH + 2)$

SCL low time (in I²C function clocks) = $(CLKDIV + 1) * (MSTSCLELOW + 2)$

Nominal SCL rate = I²C function clock rate / (SCL high time + SCL low time)

Slave timing

Most aspects of slave operation are controlled by SCL received from the I²C bus master. However, if the slave function stretches SCL to allow for software response, it must provide sufficient data setup time to the master before releasing the stretched clock. This is accomplished by inserting one clock time of CLKDIV at that point.

If CLKDIV is already configured for master operation, that is sufficient. If only the slave function is used, CLKDIV should be configured such that one clock time is greater than the tSU;DAT value noted in the I²C bus specification for the I²C mode that is being used.

24.7.1.2 Bus rate support

The I²C interface can support 4 modes from the I²C bus specification:

- Standard-mode (SM, rate up to 100 kbits/s)
- Fast-mode (FM, rate up to 400 kbits/s)
- Fast-mode Plus (FM+, rate up to 1 Mbits/s)
- High-speed mode (HS, rate up to 3.4 Mbits/s)

Refer to “I²C-bus specification and user manual” for details of I²C modes.

The I²C interface supports Standard-mode, Fast-mode, and Fast-mode Plus with the same software sequence, which also supports SMBus. High-speed mode is intrinsically incompatible with SMBus due to conflicting requirements and limitations for clock stretching, and therefore requires a slightly different software sequence.

24.7.1.2.1 High-speed mode support

High-speed mode requires different pin filtering, somewhat different timing, and a different drive strength on SCL for the master function. The changes needed for the handling of the acknowledge bit mean that SMBus cannot be supported when the I²C is configured to be HS capable. This limitation is intrinsic to the SMBus and High-speed I²C specifications.

Because of the timing of changes to pin drive strength and filtering, the I²C block is designed to directly control those pad characteristics when configured to be HS capable. The I²C also recognizes HS master codes and responds to programmed addresses when HS capable.

For software consistency, the changes required for handling of acknowledge and address recognition, and which affect when interrupts occur, are always in effect when the I²C is configured to be HS capable. This means that software does not need to know if a particular transfer is actually in HS mode or not.

24.7.1.2.2 Clock stretching

The I²C interface automatically stretches the clock when it does not have sufficient information on how to proceed, i.e. software has not supplied data and/or instructions to generate a start or stop. In principle, at least, I²C can allow the clock to be stretched by any bus participant at any time that SCL is low, in SM, FM, and MF+ modes.

In practice, the I2C interface described here may stretch SCL at the following times, in SM, FM, and MF+ modes:

- As a Slave:
 - after an address is received that complies with at least one slave address (before the address is acknowledged)
 - as a slave receiver, after each data byte received (software then acknowledges the data)
 - as a slave transmitter, after each data byte is sent and the matching acknowledge is received from the master
- As a master:
 - after each address is sent and the acknowledge bit has been received
 - as a master receiver, after each after each data byte is received (software then acknowledges the data)
 - as a master transmitter, after each data byte is sent and the matching acknowledge bit has been received from the slave

In HS mode:

- As a Slave (only slave functions in HS mode are supported on this device)
 - as a slave receiver, after each data byte is received and automatically acknowledged

- as a slave transmitter, after each after each data byte is sent and the matching acknowledge is received from the master

In each case, the relevant pending flag (MSTPENDING or SLVPENDING) is set at the point where clock stretching occurs.

24.7.2 Time-out

A time-out feature on an I²C interface can be used to detect a “stuck” bus and potentially do something to alleviate the condition. Two different types of time-out are supported. Both types apply whenever the I²C block and the time-out function are both enabled. Master, Slave, or Monitor functions do not need to be enabled.

In the first type of time-out, reflected by the EVENTTIMEOUT flag in the STAT register, the time between bus events governs the time-out check. These events include Start, Stop, and all changes on the I²C clock (SCL). This time-out is asserted when the time between any of these events is longer than the time configured in the TIMEOUT register. This time-out could be useful in monitoring an I²C bus within a system as part of a method to keep the bus running of problems occur.

The second type of I²C time-out is reflected by the SCLTIMEOUT flag in the STAT register. This time-out is asserted when the SCL signal remains low longer than the time configured in the TIMEOUT register. This corresponds to SMBus time-out parameter T_{TIMEOUT}. In this situation, a slave could reset its own I²C interface in case it is the offending device. If all listening slaves (including masters that can be addressed as slaves) do this, then the bus will be released unless it is a current master causing the problem. Refer to the SMBus specification for more details.

Both types of time-out are generated only when the I²C bus is considered busy, i.e. when there has been a Start condition more recently than a Stop condition.

24.7.3 Ten-bit addressing

Ten-bit addressing is accomplished by the I²C master sending a second address byte to extend a particular range of standard 7-bit addresses. In the case of the master writing to the slave, the I²C frame simply continues with data after the 2 address bytes. For the master to read from a slave, it needs to reverse the data direction after the second address byte. This is done by sending a Repeated Start, followed by a repeat of the same standard 7-bit address, with a Read bit. The slave must remember that it had been addressed by the previous write operation and stay selected for the subsequent read with the correct partial I²C address.

For the Master function, the I²C is simply instructed to perform the 2-byte addressing as a normal write operation, followed either by more write data, or by a Repeated Start with a repeat of the first part of the 10-bit slave address and then reading in the normal fashion.

For the Slave function, the first part of the address is automatically matched in the same fashion as 7-bit addressing. The Slave address qualifier feature (see [Section 24.6.14](#)) can be used to intercept all potential 10-bit addresses (first address byte values F0 through F6), or just one. In the case of Slave Receiver mode, data is received in the normal fashion after software matches the first data byte to the remaining portion of the 10-bit address. The Slave function should record the fact that it has been addressed, in case there is a follow-up read operation.

For Slave Transmitter mode, the slave function responds to the initial address in the same fashion as for Slave Receiver mode, and checks that it has previously been addressed with a full 10-bit address. If the address matched is address 0, and address qualification is enabled, software must check that the first part of the 10-bit address is a complete match to the previous address before acknowledging the address.

24.7.4 Clocking and power considerations

The Master function of the I²C always requires a peripheral clock to be running in order to operate. The Slave function can operate without any internal clocking when the slave is not currently addressed. This means that reduced power modes up to Power-down mode can be entered, and the device will wake up when the I²C Slave function recognizes an address. Monitor mode can similarly wake up the device from a reduced power mode when information becomes available.

24.7.5 Interrupt handling

The I²C provides a single interrupt output that handles all interrupts for Master, Slave, and Monitor functions.

24.7.6 DMA

DMA with the I²C is done only for data transfer, DMA cannot handle control of the I²C. Once DMA is transferring data, I²C acknowledges are handled implicitly. No CPU intervention is required while DMA is transferring data.

Generally, data transfers can be handled by DMA for Master mode after an address is sent and acknowledged by a slave, and for Slave mode after software has acknowledged an address. In either mode, software is always involved in the address portion of a message. In master and slave modes, data receive and transmit data can be transferred by the DMA. The DMA supports three DMA requests: data transfer in master mode, slave mode, and monitor mode.

A received NACK (from a slave in Master mode, or from a master in Slave mode) will cause DMA to stop and an interrupt to be generated. A Repeated Start sensed on the bus will similarly cause DMA to stop and an interrupt to be generated.

24.7.6.1 DMA as a Master transmitter

A basic sequence for a Master transmitter:

- Software sets up DMA to transmit a message.
- Software causes a slave address with write command to be sent and checks that the address was acknowledged.
- Software turns on DMA mode in the I²C.
- DMA transfers data and eventually completes the transfer.
- Software causes a stop (or repeated start) to be sent.

Software will be invoked to handle any exceptions to the standard transfer, such as the slave sending a NACK before the end of the transfer.

24.7.6.2 DMA as a Master receiver

A basic sequence for a Master receiver:

- Software sets up DMA to receive a message.
- Software causes a slave address with read command to be sent and checks that the address was acknowledged.
- Software starts DMA.
- DMA completes.
- Software causes a stop or repeated start to be sent.

Software will be invoked to handle any exceptions to the standard transfer.

24.7.6.3 DMA as a Slave transmitter

A basic sequence for a Slave transmitter:

- Software acknowledges an I²C address.
- Software sets up DMA to transmit a message.
- Software starts DMA.
- DMA completes.

24.7.6.4 DMA as a Slave receiver

A basic sequence for a Slave receiver:

- Software receives an interrupt for a slave address received, and acknowledges the address.
- Software sets up DMA to receive a message, less the final data byte.
- Software starts DMA.
- DMA completes.
- Software sets SLVNACK prior to receiving the final data byte.
- Software receives the final data byte.

25.1 How to read this chapter

Read this chapter for a description of the optional FIFOs for the USART and SPI interfaces.

25.2 Basic configuration

The System FIFO is configured using the following registers:

- Use the AHBCLKCTRL1 register ([Table 52](#)) to enable the clock to the System FIFO register interface.
- Clear the System FIFO peripheral reset using the PRESETCTRL1 register ([Table 36](#)).
- Interrupts: System FIFO interrupts are shared with peripheral interrupts. The VIFO and the serviced peripherals should be configured such that only one generates any particular data interrupt. Non-data interrupt may come directly from the peripheral. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.
- DMA: transmit and receive functions handled by the System FIFO can also be operated with the system DMA controller (see [Chapter 13](#)). FIFOs that will be used with DMA must be enabled via the FIFOCTRL register, see [Table 63](#).

25.3 Features

- Performs transmit and receive FIFO operations for all USARTs and SPIs on a device, supporting software or DMA access to each peripheral transmit and receive functions.
- FIFOs provide additional timing elasticity, which is extended when they are used in conjunction with DMA.
- Buffer space is provided separately for each peripheral type (USART and SPI) and can be allocated to each peripheral within the type by the user. there are 16 FIFO entries for USART transmit, 16 for USART receive, 8 for SPI transmit, and 8 for SPI receive.
- The System FIFO accesses APB peripherals at the lower speed (which depends on the peripheral clock rate) and allows the CPU and/or DMA to access data at AHB rates, with no stalls, regardless of how slow the peripheral bus clock is running.
- Each peripheral transmit and receive FIFO has a software settable threshold ranging from 1 to FIFO full.
- Each peripheral transmit and receive FIFO provides a count of entries ranging from empty to full.
- The System FIFO provides status flags for peripheral receive data availability and transmit FIFO availability. these can be used to generate interrupts or to signal the system DMA.
- A timeout feature (for USART and SPI) provides a means to get data left for a time in a FIFO that has not reached its threshold to be transferred.

25.4 Architecture

The architecture of the System FIFO is shown in [Figure 53](#). System connection for the FIFO are shown in [Figure 54](#).

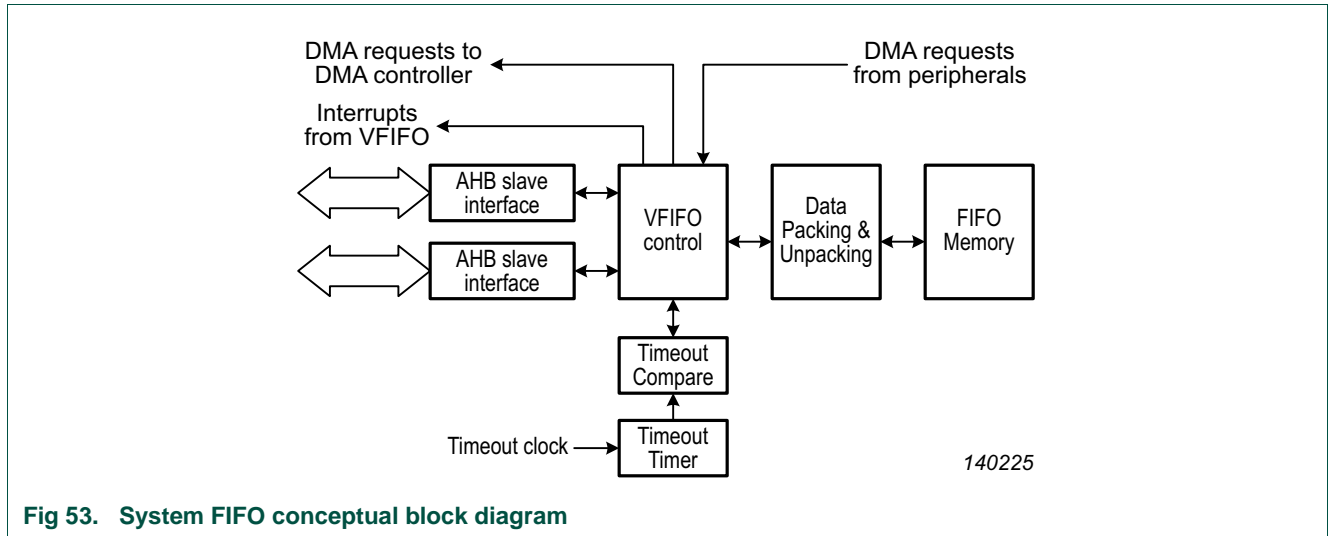


Fig 53. System FIFO conceptual block diagram

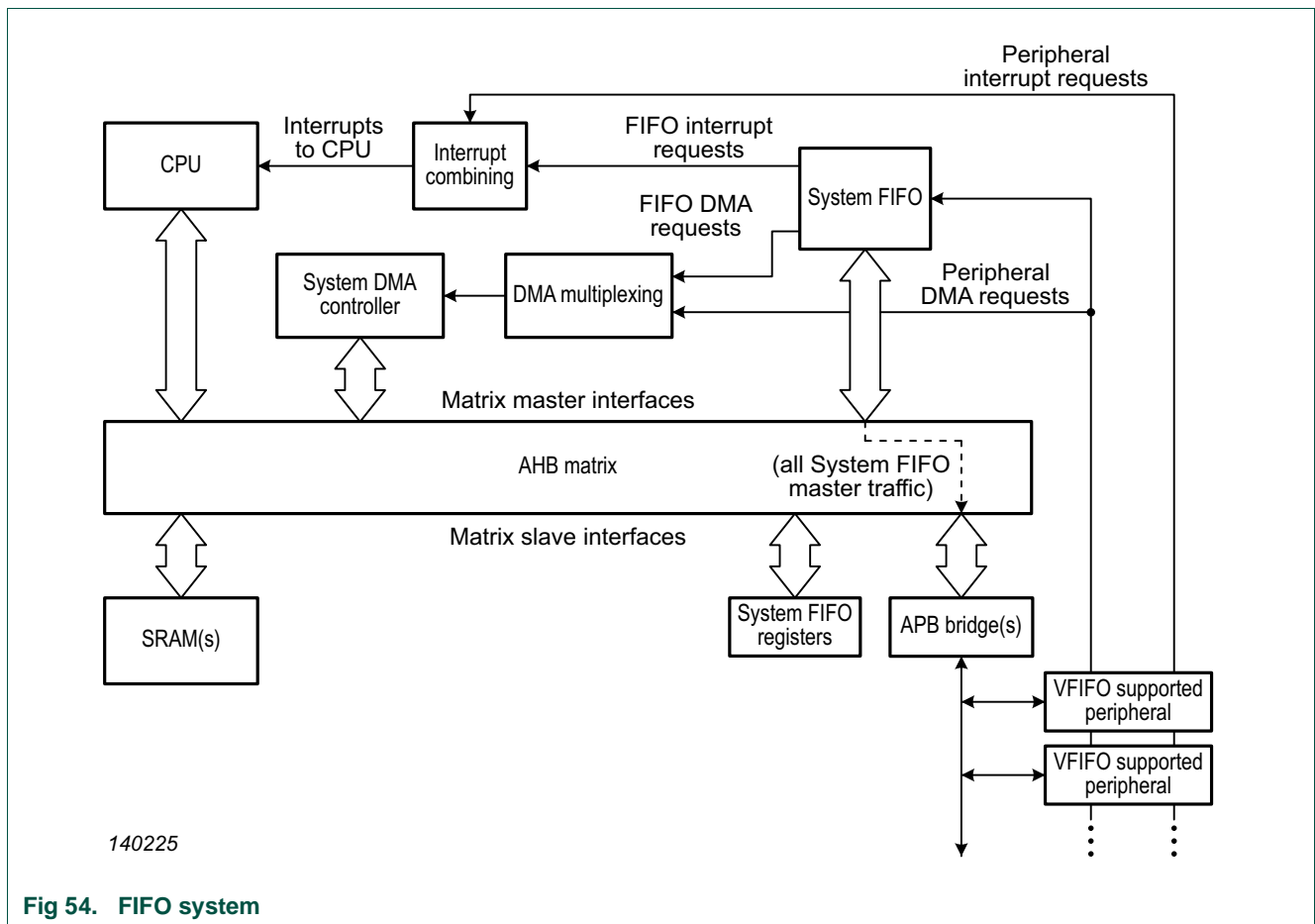


Fig 54. FIFO system

25.5 Register description

Table 370. Register overview: FIFO register map (base address 0x1C03 8000)

| Name | Access | Address Offset | Description | Reset Value ^[1] | Reference |
|-------------------------------------|--------|----------------|---|----------------------------|---------------------|
| Global System FIFO registers | | | | | |
| FIFOCTLUSART | R/W | 0x0100 | USART FIFO global control register. These registers are byte, halfword, and word addressable. The upper 16 bits of these registers provide information about the System FIFO configuration, and are specific to each device type. | 0x707 | 371 |
| FIFOUPDATEUSART | W1 | 0x0104 | USART FIFO global update register | NA | 372 |
| FIFOCFGUSART0 | R/W | 0x0110 | FIFO configuration register for USART0 | 0 | 374 |
| FIFOCFGUSART1 | R/W | 0x0114 | FIFO configuration register for USART1 | 0 | 374 |
| FIFOCFGUSART2 | R/W | 0x0118 | FIFO configuration register for USART2 | 0 | 374 |
| FIFOCFGUSART3 | R/W | 0x011C | FIFO configuration register for USART3 | 0 | 374 |
| FIFOCTLSPI | R/W | 0x0200 | SPI FIFO global control register. These registers are byte, halfword, and word addressable. The upper 16 bits of these registers provide information about the System FIFO configuration, and are specific to each device type. | 0x707 | 375 |
| FIFOUPDATESPI | W1 | 0x0204 | SPI FIFO global update register | NA | 376 |
| FIFOCFGSPI0 | R/W | 0x0210 | FIFO configuration register for SPI0 | 0 | 378 |
| FIFOCFGSPI1 | R/W | 0x0214 | FIFO configuration register for SPI0 | 0 | 378 |
| USART specific registers | | | | | |
| CFGUSART0 | R/W | 0x1000 | USART0 configuration | 0 | 380 |
| STATUSART0 | R/W | 0x1004 | USART0 status | 0x300 | 382 |
| INTSTATUSART0 | RO | 0x1008 | USART0 interrupt status | 0x300 | 384 |
| CTLSETUSART0 | RO/W1 | 0x100C | USART0 control read and set register. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set. | 0 | 386 |
| CTLCLRUSART0 | W1 | 0x1010 | USART0 control clear register. Writing a 1 to any implemented bit position causes the corresponding bit in the related CTLSET register to be cleared. | NA | 388 |
| RXDATUSART0 | RO | 0x1014 | USART0 received data | NA | 390 |
| RXDATSTATUSART0 | RO | 0x1018 | USART0 received data with status | NA | 392 |
| TXDATUSART0 | WO | 0x101C | USART0 transmit data | 0 | 394 |
| CFGUSART1 | R/W | 0x1100 | USART1 configuration | 0 | 380 |
| STATUSART1 | R/W | 0x1104 | USART1 status | 0x300 | 382 |
| INTSTATUSART1 | RO | 0x1108 | USART1 interrupt status | 0x300 | 384 |
| CTLSETUSART1 | RO/W1 | 0x110C | USART1 control read and set register. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set. | 0 | 386 |
| CTLCLRUSART1 | W1 | 0x1110 | USART1 control clear register. Writing a 1 to any implemented bit position causes the corresponding bit in the related CTLSET register to be cleared. | NA | 388 |
| RXDATUSART1 | RO | 0x1114 | USART1 received data | NA | 390 |
| RXDATSTATUSART1 | RO | 0x1118 | USART1 received data with status | NA | 392 |
| TXDATUSART1 | WO | 0x111C | USART1 transmit data | 0 | 394 |

Table 370. Register overview: FIFO register map (base address 0x1C03 8000)

| Name | Access | Address Offset | Description | Reset Value ^[1] | Reference |
|-------------------------------|--------|----------------|---|----------------------------|---------------------|
| CFGUSART2 | R/W | 0x1200 | USART2 configuration | 0 | 380 |
| STATUSART2 | R/W | 0x1204 | USART2 status | 0x300 | 382 |
| INTSTATUSART2 | RO | 0x1208 | USART2 interrupt status | 0x300 | 384 |
| CTLSETUSART2 | RO/W1 | 0x120C | USART2 control read and set register. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set. | 0 | 386 |
| CTLCLRUSART2 | W1 | 0x1210 | USART2 control clear register. Writing a 1 to any implemented bit position causes the corresponding bit in the related CTLSET register to be cleared. | NA | 388 |
| RXDATUSART2 | RO | 0x1214 | USART2 received data | NA | 390 |
| RXDATSTATUSART2 | RO | 0x1218 | USART2 received data with status | NA | 392 |
| TXDATUSART2 | WO | 0x121C | USART2 transmit data | 0 | 394 |
| CFGUSART3 | R/W | 0x1300 | USART3 configuration | 0 | 380 |
| STATUSART3 | R/W | 0x1304 | USART3 status | 0x300 | 382 |
| INTSTATUSART3 | RO | 0x1308 | USART3 interrupt status | 0x300 | 384 |
| CTLSETUSART3 | RO/W1 | 0x130C | USART3 control read and set register. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set. | 0 | 386 |
| CTLCLRUSART3 | W1 | 0x1310 | USART3 control clear register. Writing a 1 to any implemented bit position causes the corresponding bit in the related CTLSET register to be cleared. | NA | 388 |
| RXDATUSART3 | RO | 0x1314 | USART3 received data | NA | 390 |
| RXDATSTATUSART3 | RO | 0x1318 | USART3 received data with status | NA | 392 |
| TXDATUSART3 | WO | 0x131C | USART3 transmit data | 0 | 394 |
| SPI specific registers | | | | | |
| CFGSPI0 | R/W | 0x2000 | SPI0 configuration | 0 | 396 |
| STATSPI0 | R/W | 0x2004 | SPI0 status | 0x300 | 398 |
| INTSTATSPI0 | RO | 0x2008 | SPI0 interrupt status | 0x300 | 400 |
| CTLSETSPI0 | RO/W1 | 0x200C | SPI0 control read and set register. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set. | 0 | 402 |
| CTLCLRSPi0 | W1 | 0x2010 | SPI0 control clear register. Writing a 1 to any implemented bit position causes the corresponding bit in the related CTLSET register to be cleared. | NA | 404 |
| RXDATSPi0 | RO | 0x2014 | SPI0 received data. These registers are half word addressable. | NA | 406 |
| TXDATCTLSPi0 | WO | 0x2018 | SPI0 transmit data. These registers are half word addressable. | NA | 408 |
| CFGSPi1 | R/W | 0x2100 | SPI1 configuration | 0 | 396 |
| STATSPi1 | R/W | 0x2104 | SPI1 status | 0x300 | 398 |
| INTSTATSPi1 | RO | 0x2108 | SPI1 interrupt status | 0x300 | 400 |
| CTLSETSPi1 | RO/W1 | 0x210C | SPI1 control read and set register. A complete value may be read from this register. Writing a 1 to any implemented bit position causes that bit to be set. | 0 | 402 |

Table 370. Register overview: FIFO register map (base address 0x1C03 8000)

| Name | Access | Address Offset | Description | Reset Value ^[1] | Reference |
|-------------|--------|----------------|---|----------------------------|---------------------|
| CTLCLRSP1 | W1 | 0x2110 | SPI1 control clear register. Writing a 1 to any implemented bit position causes the corresponding bit in the related CTLSET register to be cleared. | NA | 404 |
| RXDATSP1 | RO | 0x2114 | SPI1 received data. These registers are half word addressable. | NA | 406 |
| TXDATCTLSP1 | WO | 0x2118 | SPI1 transmit data. These registers are half word addressable. | NA | 408 |

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

25.5.1 USART FIFO global control register

The FIFCTLUSART register contains information about the configuration of USART support for the specific instance of the System FIFO as well as global control and status for the USART FIFOs.

Table 371. USART FIFO global control register (FIFCTLUSART, address offset 0x0100) bit description

| Bit | Symbol | Description | Access | Reset Value |
|-------|-------------|--|--------|-------------|
| 0 | RXPAUSE | Pause all USARTs receive FIFO operations. This can be used to prepare the System FIFO to reconfigure FIFO allocations among the USART receivers. | R/W | 1 |
| 1 | RXPAUSED | All USART receive FIFOs are paused. | RO | 1 |
| 2 | RXEMPTY | All USART receive FIFOs are empty. | RO | 1 |
| 7:3 | - | Reserved. Read value is undefined, only zero should be written. | - | NA |
| 8 | TXPAUSE | Pause all USARTs transmit FIFO operations. This can be used to prepare the System FIFO to reconfigure FIFO allocations among the USART transmitters. | R/W | 1 |
| 9 | TXPAUSED | All USART transmit FIFOs are paused. | RO | 1 |
| 10 | TXEMPTY | All USART transmit FIFOs are empty. | RO | 1 |
| 15:11 | - | Reserved. Read value is undefined, only zero should be written. | - | NA |
| 23:16 | RXFIFOTOTAL | Reports the receive FIFO space available for USARTs on this FIFO. The reset value is device specific. | RO | - |
| 31:24 | TXFIFOTOTAL | Reports the transmit FIFO space available for USARTs on this FIFO. The reset value is device specific. | RO | - |

25.5.2 USART FIFO global update register

The FIFOUPTUSART register is used to update FIFO sizes when changes are made to any FIFOCFGUSART register(s). When a FIFO is updated, all internal pointers are reset to their default values and the FIFO will be empty. All USART Rx and/or Tx FIFOs should be updated together.

Table 372. USART FIFO global reset register (FIFOUPTUSART, address offset 0x0104) bit description

| Bit | Symbol | Description | Reset Value |
|------|------------------------|---|-------------|
| 0 | USART0RX UPDATESIZE | Writing 1 updates USART0 Rx FIFO size to match the USART0 RXSIZE. Must be done for all USARTs when any USART RXSIZE is changed. | 0 |
| 1 | USART1RX UPDATESIZE | Writing 1 updates USART1 Rx FIFO size to match the USART1 RXSIZE. Must be done for all USARTs when any USART RXSIZE is changed. | 0 |
| 2 | USART2RX UPDATESIZE | Writing 1 updates USART2 Rx FIFO size to match the USART2 RXSIZE. Must be done for all USARTs when any USART RXSIZE is changed. | 0 |
| 3 | USART3RX UPDATESIZE | Writing 1 updates USART3 Rx FIFO size to match the USART3 RXSIZE. Must be done for all USARTs when any USART RXSIZE is changed. | 0 |
| 15:4 | - | Reserved. Read value is undefined, only zero should be written. | - |
| 16 | USART0TX UPDATESIZE | Writing 1 updates USART0 Tx FIFO size to match the USART0 TXSIZE. Must be done for all USARTs when any USART TXSIZE is changed. | 0 |
| 17 | USART1TX UPDATESIZE | Writing 1 updates USART1 Tx FIFO size to match the USART1 TXSIZE. Must be done for all USARTs when any USART TXSIZE is changed. | 0 |

Table 372. USART FIFO global reset register (FIFOUPDATEUSART, address offset 0x0104) bit description

| Bit | Symbol | Description | Reset Value |
|-------|------------------------|---|-------------|
| 18 | USART2TX UPDATESIZE | Writing 1 updates USART2 Tx FIFO size to match the USART2 TXSIZE. Must be done for all USARTs when any USART TXSIZE is changed. | 0 |
| 19 | USART3TX UPDATESIZE | Writing 1 updates USART3 Tx FIFO size to match the USART3 TXSIZE. Must be done for all USARTs when any USART TXSIZE is changed. | 0 |
| 31:20 | - | Reserved. Read value is undefined, only zero should be written. | NA |

25.5.3 FIFO configuration register for USART0 to USART3

The FIFOCFGUSART register configure the FIFO sizes for the related USART receiver and transmitter. Each USART has a dedicated FIFOCFG register.

Remark: To reconfigure transmit or receive FIFOs, all USART transmit or receive functions must be paused and their FIFOs empty. Before any are un-paused, all USART FIFO sizes must be configured such that no more than the available FIFO space is allocated. That is, the sum of all FIFO sizes must not exceed the related FIFOTOTAL value in the FIFOCTLUSART register. After configuration, the FIFOs must be reset. See [Section 25.6.1 “Configuring peripheral FIFOs”](#).

Table 373. Address map FIFOCFGUSART[0:3] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| VFIFO | 0x1C03 8000 | [0x110:0x11C] | 0x4 | 4 |

Table 374. FIFO configuration register for USARTn (FIFOCFGUSART[0:3], address offset [0x0110:0x011C]) bit description

| Bit | Symbol | Description | Reset Value |
|-------|--------|---|-------------|
| 7:0 | RXSIZE | Configures the USART receive FIFO size. A zero values provides no System FIFO service for the related USART receiver. | 0 |
| 15:8 | TXSIZE | Configures the USART transmit FIFO size. A zero values provides no System FIFO service for the related USART transmitter. | 0 |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | NA |

25.5.4 SPI FIFO global control register

The FIFCTLSPi register contains information about the configuration of SPI support for the specific instance of the System FIFO as well as global control and status for the SPI FIFOs.

Table 375. SPI FIFO global control register (FIFCTLSPi, address offset 0x0200) bit description

| Bit | Symbol | Description | Access | Reset Value |
|-------|-------------|--|--------|-------------|
| 0 | RXPAUSE | Pause all SPIs receive FIFO operations. This can be used to prepare the System FIFO to reconfigure FIFO allocations among the SPI receivers. | R/W | 1 |
| 1 | RXPAUSED | All SPI receive FIFOs are paused. | RO | 1 |
| 2 | RXEMPTY | All SPI receive FIFOs are empty. | RO | 1 |
| 7:3 | - | Reserved. Read value is undefined, only zero should be written. | - | NA |
| 8 | TXPAUSE | Pause all SPIs transmit FIFO operations. This can be used to prepare the System FIFO to reconfigure FIFO allocations among the SPI transmitters. | R/W | 1 |
| 9 | TXPAUSED | All SPI transmit FIFOs are paused. | RO | 1 |
| 10 | TXEMPTY | All SPI transmit FIFOs are empty. | RO | 1 |
| 15:11 | - | Reserved. Read value is undefined, only zero should be written. | - | NA |
| 23:16 | RXFIFOTOTAL | Reports the receive FIFO space available for SPIs on the System FIFO. The reset value is device specific. | RO | - |
| 31:24 | TXFIFOTOTAL | Reports the transmit FIFO space available for SPIs on the System FIFO. The reset value is device specific. | RO | - |

25.5.5 SPI FIFO global reset register

The FIFUUPDATESPI register is used to update FIFO sizes when changes are made to any FIFOCFGSPI register(s). When a FIFO is updated, all internal pointers are reset to their default values and the FIFO will be empty. All SPI Rx and/or Tx FIFOs should be updated together.

Table 376. SPI FIFO global reset register (FIFUUPDATESPI, address offset 0x0204) bit description

| Bit | Symbol | Description | Reset Value |
|-------|----------------------|---|-------------|
| 0 | SPI0RX UPDATESIZE | Writing 1 updates SPI0 Rx FIFO size to match the SPI0 RXSIZE. Must be done for all SPIs when any SPI RXSIZE is changed. | 0 |
| 1 | SPI1RX UPDATESIZE | Writing 1 updates SPI1 Rx FIFO size to match the SPI1 RXSIZE. Must be done for all SPIs when any SPI RXSIZE is changed. | 0 |
| 15:3 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 16 | SPI0TX UPDATESIZE | Writing 1 updates SPI0 Tx FIFO size to match the SPI0 TXSIZE. Must be done for all SPIs when any SPI TXSIZE is changed. | 0 |
| 17 | SPI1TX UPDATESIZE | Writing 1 updates SPI1 Tx FIFO size to match the SPI1 TXSIZE. Must be done for all SPIs when any SPI TXSIZE is changed. | 0 |
| 31:18 | - | Reserved. Read value is undefined, only zero should be written. | NA |

25.5.6 FIFO configuration register for SPI0 and SPI1

The FIFOCFGSPI register configure the FIFO sizes for the related SPI receiver and transmitter. Each SPI has a dedicated FIFOCFG register.

Remark: To reconfigure transmit or receive FIFOs, all SPI transmit or receive functions must be paused and their FIFOs empty. Before any are un-paused, all SPI FIFO sizes must be configured such that no more than the available FIFO space is allocated. That is, the sum of all FIFO sizes must not exceed the related FIFOTOTAL value in the FIFCTLSPI register. After configuration, the FIFOs must be reset. See [Section 25.6.1 “Configuring peripheral FIFOs”](#).

Table 377. Address map FIFOCFGSPI[0:1] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| VFIFO | 0x1C03 8000 | [0x210:0x214] | 0x4 | 2 |

Table 378. FIFO configuration register for SPIn (FIFOCFGSPI[0:1], address offset [0x210:0x214]) bit description

| Bit | Symbol | Description | Reset Value |
|-------|--------|---|-------------|
| 7:0 | RXSIZE | Configures the SPI receive FIFO size. A zero values provides no System FIFO service for the related SPI receiver. | 0 |
| 15:8 | TXSIZE | Configures the SPI transmit FIFO size. A zero values provides no System FIFO service for the related SPI transmitter. | 0 |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | NA |

25.5.7 Configuration register for USARTn

The CFGUSART register allows configuration of the transmit and receive FIFO thresholds and the receive FIFO timeout. Each USART has a dedicated CFGUSART register.

Table 379. Address map CFGUSART[0:3] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| VFIFO | 0x1C03 8000 | [0x1100:0x1300] | 0x100 | 4 |

Table 380. Configuration register for USARTn (CFGUSART[0:3], address offset [0x1000:0x1300]) bit description

| Bit | Symbol | Description | Reset Value |
|-------|----------------------------|--|-------------|
| 3:0 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 4 | TIMEOUT CONT ONWRITE | Timeout Continue On Write. When 0, the timeout for the related peripheral is reset every time data is transferred from the peripheral into the receive FIFO. When 1, the timeout for the related peripheral is not reset every time data is transferred into the receive FIFO. This allows the timeout to be applied to accumulated data, perhaps related to the FIFO threshold. | 0 |
| 5 | TIMEOUT CONT ONEMPTY | Timeout Continue On Empty. When 0, the timeout for the related peripheral is reset when the receive FIFO becomes empty. When 1, the timeout for the related peripheral is not reset when the receive FIFO becomes empty. This allows the timeout to be used to flag idle peripherals, and could potentially be used to indicate the end of a transmission of indeterminate length. | 0 |
| 7:6 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 11:8 | TIMEOUT BASE | Specifies the least significant timer bit to compare to TimeoutValue. See Section 25.5.7.1 below. Value can be 0 through 15. | 0 |
| 15:12 | TIMEOUT VALUE | Specifies the maximum time value for timeout at the timer position identified by TimeoutBase. Minimum time TimeoutValue - 1. is See Section 25.5.7.1 below. TimeoutValue should not be 0 or 1 when timeout is enabled. | 0 |
| 23:16 | RX THRESHOLD | Receive FIFO Threshold. The System FIFO indicates that the threshold has been reached when the number of entries in the receive FIFO is greater than this value. For example, when RxThreshold = 0, the threshold is exceeded when there is at least one entry in the receive FIFO. An interrupt can be generated when the RxThreshold has been reached (see Section 25.5.10), but has no effect on DMA requests, which are generated whenever the receiver FIFO is not empty. | 0 |
| 31:24 | TX THRESHOLD | Transmit FIFO Threshold. The System FIFO indicates that the threshold has been reached when the number of free entries in the transmit FIFO is less than or equal to this value. For example, when TxThreshold = 0, the threshold is exceeded when there is at least one free entry in the transmit FIFO. An interrupt can be generated when the TxThreshold has been reached (see Section 25.5.10), but has no effect on DMA requests, which are generated whenever the transmit FIFO has any free entries. | 0 |

25.5.7.1 Receiver Timeout

A single 19-bit timeout timer is used for all receiver FIFOs. TimeoutBase chooses a bit of that timer to be used as the place to begin comparing the timer to TimeoutValue, from bit 0 up to bit 15. The compare is always 4 bits in size. TimeoutValue can be any value from 2 to 15. This gives a maximum timeout range of 2 counts (too small to be useful) at the bottom end, up to 15 * 32,786 (491,152) counts at the upper end. TimeoutValue of 0 and 1 should not be used. The timeout is enabled for each peripheral when the related interrupt is enabled.

For instance, if TimeoutBase is set to 5, then 4 bits of the timeout timer starting at bit 5 are compared to TimeoutValue. Since bit 5 changes every 32 counts of the timeout timer, the maximum time for a timeout to occur (since the timer may change immediately after data is entered into the FIFO) is TimeoutValue (a range of 2 to 15) * 32 clocks. The source of the timeout clock is device dependent.

25.5.8 Status register for USARTn

The STATUSART register provides information about the current state of the USART FIFOs. Each USART has a dedicated STATUSART register.

Table 381. Address map STATUSART[0:3] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| VFIFO | 0x1C03 8000 | [0x1004:0x1304] | 0x100 | 4 |

Table 382. Status register for USARTn (STATUSART[0:3], address offset [0x1004:0x1304]) bit description

| Bit | Symbol | Description | Reset Value |
|-------|---------------|--|-------------|
| 0 | RXTH | Receive FIFO Threshold. When 1, the receive FIFO threshold has been reached. This is a read-only bit. | 0 |
| 1 | TXTH | Transmit FIFO Threshold. When 1, the transmit FIFO threshold has been reached. This is a read-only bit. | 0 |
| 3:2 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 4 | RX TIMEOUT | Receive FIFO Timeout. When 1, the receive FIFO has timed out, based on the timeout configuration in the CFGUSART register. The timeout condition can be cleared by writing a 1 to this bit, by enabling or disabling the timeout interrupt, or by writing a 1 to the timeout interrupt enable. | 0 |
| 6:5 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 7 | BUSERR | Bus Error. When 1, a bus error has occurred while processing data for USARTn. The bus error flag can be cleared by writing a 1 to this bit. | 0 |
| 8 | RXEMPTY | Receive FIFO Empty. When 1, the receive FIFO is currently empty. This is a read-only bit. | 1 |
| 9 | TXEMPTY | Transmit FIFO Empty. When 1, the transmit FIFO is currently empty. This is a read-only bit. | 1 |
| 15:10 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 23:16 | RXCOUNT | Receive FIFO Count. Indicates how many entries may be read from the receive FIFO. 0 = FIFO empty. This is a read-only field. | 0 |
| 31:24 | TXCOUNT | Transmit FIFO Count. Indicates how many entries may be written to the transmit FIFO. 0 = FIFO full. This is a read-only field that is valid only when the TxFIFO is fully configured and enabled. | 0 |

25.5.9 Interrupt status register for USARTn

The read-only INTSTATUSART register provides information about a peripheral being serviced by the System FIFO that may be needed by an interrupt service routine. Each USART has a dedicated INTSTATUSART register.

Table 383. Address map INTSTATUSART[0:3] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| VFIFO | 0x1C03 8000 | [0x1008:0x1308] | 0x100 | 4 |

Table 384. Interrupt status register for USARTn (INTSTATUSART[0:3], address offset [0x1008:0x1308]) bit description

| Bit | Symbol | Description | Reset Value |
|-------|---------------|---|-------------|
| 0 | RXTH | Receive FIFO Threshold. When 1, the receive FIFO threshold has been reached, and the related interrupt is enabled. | 0 |
| 1 | TXTH | Transmit FIFO Threshold. When 1, the transmit FIFO threshold has been reached, and the related interrupt is enabled. | 0 |
| 3:2 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 4 | RX TIMEOUT | Receive Timeout. When 1, the receive FIFO has timed out, based on the timeout configuration in the CFGUSART register, and the related interrupt is enabled. | 0 |
| 6:5 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 7 | BUSERR | Bus Error. This is simply a copy of the same bit in the STATUSART register. The bus error interrupt is always enabled. | 0 |
| 8 | RXEMPTY | Receive FIFO Empty. This is simply a copy of the same bit in the STATUSART register. | 1 |
| 9 | TXEMPTY | Transmit FIFO Empty. This is simply a copy of the same bit in the STATUSART register. | 1 |
| 15:10 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 23:16 | RXCOUNT | Receive FIFO Count. This is simply a copy of the same field in the STATUSART register, included here so an ISR can read all needed status information in one read. | 0 |
| 31:24 | TXCOUNT | Transmit FIFO Available. This is simply a copy of the same field in the STATUSART register, included here so an ISR can read all needed status information in one read. | 0 |

25.5.10 Control read and set register for USARTn

The CTLSETUSART register determines which interrupts are passed on to the system interrupt controller. Writing a 1 to a defined bit causes that bit to be set, enabling the related interrupt. Writing 0 has no effect. When read, the state of the interrupt enables is provided. Each USART has a dedicated CTLSETUSART register.

Table 385. Address map CTLSETUSART[0:3] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| VFIFO | 0x1C03 8000 | [0x100C:0x130C] | 0x100 | 4 |

Table 386. Control read and set register for USARTn (CTLSETUSART[0:3], address offset [0x100C:0x130C]) bit description

| Bit | Symbol | Description | Reset Value |
|-----|-----------|---|-------------|
| 0 | RXTHINTEN | Receive FIFO Threshold Interrupt Enable. | 0 |
| 1 | TXTHINTEN | Transmit FIFO Threshold Interrupt Enable. | 0 |
| 3:2 | - | Reserved. Read value is undefined, only zero should be written. | NA |

Table 386. Control read and set register for USARTn (CTLSETUSART[0:3], address offset [0x100C:0x130C]) bit description

| Bit | Symbol | Description | Reset Value |
|-------|-----------------|--|-------------|
| 4 | RXTIMEOUT INTEN | Receive FIFO Timeout Interrupt Enable. When enabled, this also enables the timeout for this USART. Writing a 1 to this bit resets the USART timeout logic. | 0 |
| 7:5 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 8 | RXFLUSH | Receive FIFO flush. Writing a 1 to this bit forces the receive FIFO to be empty. | 0 |
| 9 | TXFLUSH | Transmit FIFO flush. Writing a 1 to this bit forces the transmit FIFO to be empty. | 0 |
| 31:10 | - | Reserved. Read value is undefined, only zero should be written. | NA |

25.5.11 Control clear register for USARTn

The write-only CTLCLRUSART register allows disabling selected FIFO interrupts. Writing a 1 to a defined bit causes the related bit in CTLSETUSART to be set, disabling the related interrupt. Writing 0 has no effect. Each USART has a dedicated CTLCLRUSART register.

Table 387. Address map CTLCLRUSARTTT[0:3] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| VFIFO | 0x1C03 8000 | [0x1010:0x1310] | 0x100 | 4 |

Table 388. Control read and clear register for USARTn (CTLCLRUSART[0:3], address offset [0x1010:0x1310]) bit description

| Bit | Symbol | Description | Reset Value |
|-------|-----------------|---|-------------|
| 0 | RXTHINTCLR | Receive FIFO Threshold Interrupt clear. | NA |
| 1 | TXTHINTCLR | Transmit FIFO Threshold Interrupt clear. | NA |
| 3:2 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 4 | RXTIMEOUTINTCLR | Receive FIFO Time-out Interrupt clear. | NA |
| 7:5 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 8 | RXFLUSHCLR | Receive FIFO flush clear. | 0 |
| 9 | TXFLUSHCLR | Transmit FIFO flush clear. | 0 |
| 31:10 | - | Reserved. Read value is undefined, only zero should be written. | NA |

25.5.12 Received data register for USARTn

The RXDATUSART register reads receive FIFO data that is an image of the USART's RXDAT register. If USART status is meant to be read along with data, the RXDATSTATUSART register should be used instead of RXDATUSART. Each USART has a dedicated RXDATUSART register.

Table 389. Address map RXDATUSART[0:3] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| VFIFO | 0x1C03 8000 | [0x1014:0x1314] | 0x100 | 4 |

Table 390. Received data register for USARTn (RXDATUSART[0:3], address offset [0x1014:0x1314]) bit description

| Bit | Symbol | Description | Reset Value |
|------|--------|--|-------------|
| 8:0 | RXDAT | The UART Receiver Data register contains the next received character. The number of bits that are relevant depends on the UART configuration settings. | 0 |
| 31:9 | - | Reserved, the value read from a reserved bit is not defined. | NA |

25.5.13 Received data with status register for USARTn

The RXDATSTATUSART register reads receive FIFO data that is an image of the USART's RXDATSTAT register. Each USART has a dedicated RXDATSTATUSART register.

Table 391. Address map RXDATSTATUSART[0:3] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| VFIFO | 0x1C03 8000 | [0x1018:0x1318] | 0x100 | 4 |

Table 392. Received data with status register for USARTn (RXDATSTATUSART[0:3], address offset [0x1018:0x1318]) bit description

| Bit | Symbol | Description | Reset Value |
|-------|-----------|--|-------------|
| 8:0 | RXDAT | The UART Receiver Data register contains the next received character. The number of bits that are relevant depends on the UART configuration settings. | 0 |
| 12:9 | - | Reserved, the value read from a reserved bit is not defined. | NA |
| 13 | FRAMERR | Framing Error status flag. This bit is valid when there is a character to be read in the RXDAT register and reflects the status of that character. This bit will set when the character in RXDAT was received with a missing stop bit at the expected location. This could be an indication of a baud rate or configuration mismatch with the transmitting source. | 0 |
| 14 | PARITYERR | Parity Error status flag. This bit is valid when there is a character to be read in the RXDAT register and reflects the status of that character. This bit will be set when a parity error is detected in a received character. | 0 |
| 15 | RXNOISE | Received Noise flag. | 0 |
| 31:16 | - | Reserved, the value read from a reserved bit is not defined. | NA |

25.5.14 Transmit data register for USARTn

The TXDATUSART register allows writing data to the transmit FIFO that will later be written by the System FIFO to the USART TXDAT register. Each USART has a dedicated TXDATUSART register.

Table 393. Address map TXDATUSART[0:3] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| VFIFO | 0x1C03 8000 | [0x101C:0x131C] | 0x100 | 4 |

Table 394. Transmit data register for USARTn (TXDATUSART[0:3], address offset [0x101C:0x131C]) bit description

| Bit | Symbol | Description | Reset Value |
|------|--------|---|-------------|
| 8:0 | TXDAT | Writing to the UART Transmit Data Register causes the data to be transmitted as soon as the transmit shift register is available and the condition for transmitting data is met: TXDIS bit = 0. | 0 |
| 31:9 | - | Reserved. Only zero should be written. | NA |

25.5.15 Configuration register for SPI0 and SPI1

The CFGSPI register allows configuration of the transmit and receive FIFO thresholds and the receive FIFO timeout. Each SPI has a dedicated CFGSPI register.

See [Section 25.5.7.1](#) for details on the TIMEOUTBASE configuration.

See [Section 25.5.15.1](#) for details on the TIMEOUTVALUE configuration.

See [Section 25.5.10](#) for details on the RXTHRESHOLD and TXTHRESHOLD configuration.

Table 395. Address map CFGSPI[0:1] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| VFIFO | 0x1C03 8000 | [0x2000:0x2100] | 0x100 | 2 |

Table 396. Configuration register for SPIn (CFGSPI[0:1], address offset [0x2000:0x2100]) bit description

| Bit | Symbol | Description | Reset Value |
|-------|----------------------|---|-------------|
| 3:0 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 4 | TIMEOUT CONT ONWRITE | Timeout Continue On Write. When 0, the timeout for the related peripheral is reset every time data is transferred from the peripheral into the receive FIFO. When 1, the timeout for the related peripheral is not reset every time data is transferred into the receive FIFO. This allows the timeout to be applied to accumulated data, perhaps related to the FIFO threshold. | 0 |
| 5 | TIMEOUT CONT ONEMPTY | Timeout Continue On Empty. When 0, the timeout for the related peripheral is reset when the receive FIFO becomes empty. When 1, the timeout for the related peripheral is not reset when the receive FIFO becomes empty. This allows the timeout to be used to flag idle peripherals, and could potentially be used to indicate the end of a transmission of indeterminate length. | 0 |
| 7:6 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 11:8 | TIMEOUT BASE | Specifies the least significant timer bit to compare to TimeoutValue. Value can be 0 through 15. | 0 |
| 15:12 | TIMEOUT VALUE | Specifies the maximum time value for timeout at the timer position identified by TimeoutBase. Minimum time TimeoutValue - 1. TimeoutValue should not be 0 or 1 when timeout is enabled. | 0 |
| 23:16 | RX THRESHOLD | Receive FIFO Threshold. The System FIFO indicates that the threshold has been reached when the number of entries in the receive FIFO is greater than this value. For example, when RxThreshold = 0, the threshold is exceeded when there is at least one entry in the receive FIFO. An interrupt can be generated when the RxThreshold has been reached, but has no effect on DMA requests, which are generated whenever the receiver FIFO is not empty. | 0 |
| 31:24 | TX THRESHOLD | Transmit FIFO Threshold. The System FIFO indicates that the threshold has been reached when the number of free entries in the transmit FIFO is less than or equal to this value. For example, when TxThreshold = 0, the threshold is exceeded when there is at least one free entry in the transmit FIFO. An interrupt can be generated when the TxThreshold has been reached, but has no effect on DMA requests, which are generated whenever the transmit FIFO has any free entries. | 0 |

25.5.15.1 Receiver Timeout

A single 19-bit timeout timer is used for all receiver FIFOs. TimeoutBase chooses a bit of that timer to be used as the place to begin comparing the timer to TimeoutValue, from bit 0 up to bit 15. The compare is always 4 bits in size. TimeoutValue can be any value from 2 to 15. This gives a maximum timeout range of 2 counts (too small to be useful) at the bottom end, up to 15 * 32,786 (491,152) counts at the upper end. TimeoutValue of 0 and 1 should not be used. The timeout is enabled for each peripheral when the related interrupt is enabled.

For instance, if TimeoutBase is set to 5, then 4 bits of the timeout timer starting at bit 5 are compared to TimeoutValue. Since bit 5 changes every 32 counts of the timeout timer, the maximum time for a timeout to occur (since the timer may change immediately after data is entered into the FIFO) is TimeoutValue (a range of 2 to 15) * 32 clocks. The source of the timeout clock is device dependent.

25.5.16 Status register for SPIs

The STATSPI register provides information about the current state of the SPI FIFOs. Each SPI has a dedicated STATSPI register.

Table 397. Address map STATSPI[0:1] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| VFIFO | 0x1C03 8000 | [0x2004:0x2104] | 0x100 | 2 |

Table 398. Status register for SPIn (STATSPI[0:1], address offset [0x2004:0x2104]) bit description

| Bit | Symbol | Description | Reset Value |
|-------|------------|--|-------------|
| 0 | RXTH | Receive FIFO Threshold. When 1, the receive FIFO threshold has been reached. This is a read-only bit. | 0 |
| 1 | TXTH | Transmit FIFO Threshold. When 1, the transmit FIFO threshold has been reached. This is a read-only bit. | 0 |
| 3:2 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 4 | RX TIMEOUT | Receive FIFO Timeout. When 1, the receive FIFO has timed out, based on the timeout configuration in the CFGSPI register. The timeout condition can be cleared by writing a 1 to this bit, by enabling or disabling the timeout interrupt, or by writing a 1 to the timeout interrupt enable. | 0 |
| 6:5 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 7 | BUSERR | Bus Error. When 1, a bus error has occurred while processing data for SPI. The bus error flag can be cleared by writing a 1 to this bit. | 0 |
| 8 | RXEMPTY | Receive FIFO Empty. When 1, the receive FIFO is currently empty. This is a read-only bit. | 1 |
| 9 | TXEMPTY | Transmit FIFO Empty. When 1, the transmit FIFO is currently empty. This is a read-only bit. | 1 |
| 15:10 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 23:16 | RXCOUNT | Receive FIFO Count. Indicates how many entries may be read from the receive FIFO. 0 = FIFO empty. This is a read-only field. | 0 |
| 31:24 | TXCOUNT | Transmit FIFO Count. Indicates how many entries may be written to the transmit FIFO. 0 = FIFO full. This is a read-only field that is valid only when the TxFIFO is fully configured and enabled. | 0 |

25.5.17 Interrupt status register for SPI0 and SPI1

The read-only INTSTATSPI register provides information about a peripheral being serviced by the System FIFO that may be needed by an interrupt service routine. Each SPI has a dedicated INTSTATSPI register.

Table 399. Address map INTSTATSPI[0:1] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| VFIFO | 0x1C03 8000 | [0x2008:0x2108] | 0x100 | 2 |

Table 400. Interrupt status register for SPIn (INTSTATSPI[0:1], address offset [0x2008:0x2108]) bit description

| Bit | Symbol | Description | Reset Value |
|-------|---------------|---|-------------|
| 0 | RXTH | Receive FIFO Threshold. When 1, the receive FIFO threshold has been reached, and the related interrupt is enabled. | 0 |
| 1 | TXTH | Transmit FIFO Threshold. When 1, the transmit FIFO threshold has been reached, and the related interrupt is enabled. | 0 |
| 3:2 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 4 | RX TIMEOUT | Receive Timeout. When 1, the receive FIFO has timed out, based on the timeout configuration in the CFGSPI register, and the related interrupt is enabled. | 0 |
| 6:5 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 7 | BUSERR | Bus Error. This is simply a copy of the same bit in the STATSPI register. The bus error interrupt is always enabled. | 0 |
| 8 | RXEMPTY | Receive FIFO Empty. This is simply a copy of the same bit in the STATSPI register. | 1 |
| 9 | TXEMPTY | Transmit FIFO Empty. This is simply a copy of the same bit in the STATSPI register. | 1 |
| 15:10 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 23:16 | RXCOUNT | Receive FIFO Count. This is simply a copy of the same field in the STATSPI register, included here so an ISR can read all needed status information in one read. | 0 |
| 31:24 | TXCOUNT | Transmit FIFO Available. This is simply a copy of the same field in the STATSPI register, included here so an ISR can read all needed status information in one read. | 0 |

25.5.18 Control read and set register for SPIn

The CTLSETSPI register determines which interrupts are passed on to the system interrupt controller. Writing a 1 to a defined bit causes that bit to be set, enabling the related interrupt. Writing 0 has no effect. When read, the state of the interrupt enables is provided. Each SPI has a dedicated CTLSETSPI register.

Table 401. Address map CTLSETSPI[0:1] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| VFIFO | 0x1C03 8000 | [0x200C:0x210C] | 0x100 | 2 |

Table 402. Control read and set register for SPIn (CTLSETSPI[0:1], address offset [0x200C:0x210C]) bit description

| Bit | Symbol | Description | Reset Value |
|-----|--------------------|--|-------------|
| 0 | RXTHINTEN | Receive FIFO Threshold Interrupt Enable. | 0 |
| 1 | TXTHINTEN | Transmit FIFO Threshold Interrupt Enable. | 0 |
| 3:2 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 4 | RXTIMEOUT INTEN | Receive FIFO Timeout Interrupt Enable. When enabled, this also enables the timeout for this SPI. Writing a 1 to this bit resets the SPI timeout logic. | 0 |
| 7:5 | - | Reserved. Read value is undefined, only zero should be written. | NA |

Table 402. Control read and set register for SPI_n (CTLSETSPI[0:1], address offset [0x200C:0x210C]) bit description

| Bit | Symbol | Description | Reset Value |
|-------|---------|--|-------------|
| 8 | RXFLUSH | Receive FIFO flush. Writing a 1 to this bit forces the receive FIFO to be empty. | 0 |
| 9 | TXFLUSH | Transmit FIFO flush. Writing a 1 to this bit forces the transmit FIFO to be empty. | 0 |
| 31:10 | - | Reserved. Read value is undefined, only zero should be written. | NA |

25.5.19 Control clear register for SPI₀ and SPI₁

The write-only CTLCLRSPI register allows disabling selected System FIFO interrupts. Writing a 1 to a defined bit causes the related bit in CTLSETSPI to be cleared, disabling the related interrupt. Writing 0 has no effect. Each SPI has a dedicated CTLCLRSPI register.

Table 403. Address map CTLCLRSPI[0:1] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| VFIFO | 0x1C03 8000 | [0x2010:0x2110] | 0x100 | 2 |

Table 404. Control read and clear register for SPI_n (CTLCLRSPI[0:1], address offset [0x2010:0x2110]) bit description

| Bit | Symbol | Description | Reset Value |
|-------|---------------------|--|-------------|
| 31:0 | - | Writing ones to this register clears the corresponding bit or bits in the CTLSETSPI register, if they are implemented. Bits that do not correspond to defined bits in CTLSETSPI are reserved and only zeroes should be written to them. | NA |
| 0 | RXTHINTCLR | Receive FIFO Threshold Interrupt clear. | 0 |
| 1 | TXTHINTCLR | Transmit FIFO Threshold Interrupt clear. | 0 |
| 3:2 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 4 | RXTIMEOUT INTCLR | Receive FIFO Timeout Interrupt clear. | 0 |
| 7:5 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 8 | RXFLUSHCLR | Receive FIFO flush clear. Allows the Rx FIFO to operate if it is being flushed via the RXFLUSH bit in CTLSETSPI. | 0 |
| 9 | TXFLUSHCLR | Transmit FIFO flush clear. Allows the Tx FIFO to operate if it is being flushed via the TXFLUSH bit in CTLSETSPI. | 0 |
| 31:10 | - | Reserved. Read value is undefined, only zero should be written. | NA |

25.5.20 Received data register for SPI₀ and SPI₁

The RXDATSPI register reads receive FIFO data that is an image of the SPI's RXDAT register. If SPI status is meant to be read along with data, a word read provides both. A halfword read provides only the data without status. Each SPI has a dedicated RXDATSPI register.

Table 405. Address map RXDATSPI[0:1] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| VFIFO | 0x1C03 8000 | [0x2014:0x2114] | 0x100 | 2 |

Table 406. Received data register for SPIn (RXDATSPI[0:1], address offset [0x2014:0x2114]) bit description

| Bit | Symbol | Description | Reset Value |
|-------|-----------|---|-------------|
| 15:0 | RXDAT | Receiver Data. This contains the next piece of received data. The number of bits that are used depends on the LEN setting in TXCTL / TXDATCTL. | NA |
| 16 | RXSSEL0_N | Slave Select for receive. This field allows the state of the SSEL0 pin to be saved along with received data. The value will reflect the SSEL0 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG. | NA |
| 17 | RXSSEL1_N | Slave Select for receive. This field allows the state of the SSEL1 pin to be saved along with received data. The value will reflect the SSEL1 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG. | NA |
| 18 | RXSSEL2_N | Slave Select for receive. This field allows the state of the SSEL2 pin to be saved along with received data. The value will reflect the SSEL2 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG. | NA |
| 19 | RXSSEL3_N | Slave Select for receive. This field allows the state of the SSEL3 pin to be saved along with received data. The value will reflect the SSEL3 pin for both master and slave operation. A zero indicates that a slave select is active. The actual polarity of each slave select pin is configured by the related SPOL bit in CFG. | NA |
| 20 | SOT | Start of Transfer flag. This flag will be 1 if this is the first data after the SSELs went from deasserted to asserted (i.e., any previous transfer has ended). This information can be used to identify the first piece of data in cases where the transfer length is greater than 16 bit. | NA |
| 31:21 | - | Reserved, the value read from a reserved bit is not defined. | NA |

25.5.21 Transmit data register for SPIn

The TXDATCTLSPI register allows writing data to the transmit FIFO that will later be written by the System FIFO to the SPI TXDAT register. Each SPI has a dedicated TXDATCTLSPI register.

TXDATCTLSPI can be written as a halfword or as a word, which in practice corresponds to the writing to the SPI TXDAT, TXCTL, and TXDATCTL registers as follows:

- A word write to TXDATCTLSPI contains both data and control information for the SPI.
- A halfword write to the lower half of TXDATCTLSPI contains only data for the SPI.
- A halfword write to the upper half of TXDATCTLSPI contains only control information for the SPI.

Control information is saved within the FIFO and appended to all data writes to the SPI.

Table 407. Address map TXDATCTLSPI[0:1] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|-----------------|-----------|-----------|
| VFIFO | 0x1C03 8000 | [0x2018:0x2118] | 0x100 | 2 |

Table 408. Transmit data register for SPIn (TXDATCTLSPIn[0:1], address offset [0x2018:0x2118]) bit description

| Bit | Symbol | Description | Reset Value |
|------|-----------|---|---|
| 15:0 | TXDAT | Transmit Data. This field provides from 1 to 16 bits of data to be transmitted. | 0 |
| 16 | TXSSEL0_N | Transmit Slave Select. This field asserts SSEL0 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL0 pin is configured by bits in the CFG register. | 0 |
| | | 0 | Asserted. SSEL0 asserted. |
| | | 1 | Not asserted. SSEL0 not asserted. |
| 17 | TXSSEL1_N | Transmit Slave Select. This field asserts SSEL1 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL1 pin is configured by bits in the CFG register. | 0 |
| | | 0 | Asserted. SSEL1 asserted. |
| | | 1 | Not asserted. SSEL1 not asserted. |
| 18 | TXSSEL2_N | Transmit Slave Select. This field asserts SSEL2 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL2 pin is configured by bits in the CFG register. | 0 |
| | | 0 | Asserted. SSEL2 asserted. |
| | | 1 | Not asserted. SSEL2 not asserted. |
| 19 | TXSSEL3_N | Transmit Slave Select. This field asserts SSEL3 in master mode. The output on the pin is active LOW by default. Remark: The active state of the SSEL3 pin is configured by bits in the CFG register. | 0 |
| | | 0 | Asserted. SSEL3 asserted. |
| | | 1 | Not asserted. SSEL3 not asserted. |
| 20 | EOT | End of Transfer. The asserted SSEL will be deasserted at the end of a transfer, and remain so for at least the time specified by the Transfer_delay value in the DLY register. | 0 |
| | | 0 | Not deasserted. SSEL not deasserted. This piece of data is not treated as the end of a transfer. SSEL will not be deasserted at the end of this data. |
| | | 1 | Deasserted. SSEL deasserted. This piece of data is treated as the end of a transfer. SSEL will be deasserted at the end of this piece of data. |
| 21 | EOF | End of Frame. Between frames, a delay may be inserted, as defined by the FRAME_DELAY value in the DLY register. The end of a frame may not be particularly meaningful if the FRAME_DELAY value = 0. This control can be used as part of the support for frame lengths greater than 16 bits. | 0 |
| | | 0 | Data not EOF. This piece of data transmitted is not treated as the end of a frame. |
| | | 1 | Data EOF. This piece of data is treated as the end of a frame, causing the FRAME_DELAY time to be inserted before subsequent data is transmitted. |

Table 408. Transmit data register for SPIn (TXDATCTLSPI[0:1], address offset [0x2018:0x2118]) bit description

| Bit | Symbol | Description | Reset Value |
|-------|----------|--|-------------|
| 22 | RXIGNORE | Receive Ignore. This allows data to be transmitted using the SPI without the need to read unneeded data from the receiver to simplify the transmit process and can be used with the DMA. | 0 |
| | | 0 Read received data. Received data must be read in order to allow transmission to progress. In slave mode, an overrun error will occur if received data is not read before new data is received. | |
| | | 1 Ignore received data. Received data is ignored, allowing transmission without reading unneeded received data. No receiver flags are generated. | |
| 23 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 27:24 | LEN | Data Length. Specifies the data length from 1 to 16 bits. Note that transfer lengths greater than 16 bits are supported by implementing multiple sequential data transmits. 0x0 = Data transfer is 1 bit in length. 0x1 = Data transfer is 2 bits in length. 0x2 = Data transfer is 3 bits in length. ... 0xF = Data transfer is 16 bits in length. | 0x0 |
| 31:28 | - | Reserved. Read value is undefined, only zero should be written. | NA |

25.6 Operational details

This section describes details of System FIFO operation.

25.6.1 Configuring peripheral FIFOs

To configure a FIFO for use or re-configure a FIFO after prior use:

1. Pause the desired directions (transmit and/or receive) of the desired peripheral types (USART and/or SPI).
2. Wait for that function to reach the paused and empty state.
3. Write the FIFO configurations for all of the desired directions and peripheral types.
4. Reset the FIFOs for all of the desired directions and peripheral types.
5. Un-pause the desired directions and peripheral types.

25.6.2 Peripheral status

The System FIFO reports status for each peripheral in the registers (e.g. STATUSART0 for USART0, etc.). This includes flags to indicate whether the transmitter and/or receiver have reached the programmed threshold, and similarly if they are empty. Those status flags can all cause interrupts if enabled to do so. DMA does not make use of the thresholds: DMA requests are generated for the receiver whenever the receiver FIFO is not empty and for the transmitter whenever the transmit FIFO is not full. DMA will respond if it has been configured to support that peripheral channel.

25.6.3 Receiving data

DMA can be configured to read received data from a peripheral FIFO and generate an interrupt when a transfer is completed. An interrupt service routine could read in received data instead of DMA. Potentially being aware of the actual threshold for the channel, the ISR could always read in that amount of data when an interrupt occurs, or it could always just read until the FIFO is empty.

25.6.3.1 Receiver Timeouts

A timeout mechanism is provided that can be used to prevent data from languishing in a receiver FIFO when a data packet completes, or a channel is otherwise idle for an extended time before the FIFO has reached the programmed threshold.

The receiver timeout can be specified through for a wide range of possibilities. See [Section 25.5.7.1](#).

The timeout feature can be configured to continue timing when new received data is brought in to the FIFO from the peripheral by writing a 1 to the TimeoutContOnWrite control bit. In essence, this means that unread data in the FIFO is timed from the oldest entry to determine the timeout. When TimeoutContOnWrite is 0, the timeout restarts every time new data is entered into the receiver FIFO.

The timeout feature can be configured to continue timing even when the receiver FIFO is empty, by writing a 1 to the TimeoutContOnEmpty control bit. This can be used to determine that a receive data channel is idle unexpectedly, or possibly to indicate the end of data packet of unknown length.

When DMA is being used to receive data from the FIFO, the timeout has a limited value. A DMA request is always generated whenever the FIFO is not empty, so there should never be idle data in the FIFO for any length of time. The TimeoutContOnEmpty mode could potentially still be used to indicate an idle receiver.

25.6.4 Transmitting data

DMA can be configured to send transmit data to a peripheral FIFO and generate an interrupt when a transfer is completed. An interrupt service routine could send transmit data instead of DMA. An ISR could always send data until the transmit FIFO is full, if enough is available. When a complete data packet has been written to the transmit FIFO, the interrupt could be disabled to prevent further interrupts.

25.6.5 Channel Priority

Receive requests are given priority over transmit requests, in order to minimize potential loss of data. A simple first come, first served priority scheme is used to manage overlapping service requests within the two aforementioned groups. Simultaneously asserted requests within the same group are handled by giving priority to the lower numbered channel.

25.6.6 Errors

A bus error flag is provided for each peripheral FIFO. This flag simply reflects any internal bus error during a System FIFO data transfer. It is very unlikely that a bus error will occur in practice, but potentially problematic if not reported at all.

25.7 Generic FIFO setup

Before the FIFOs can be used, they require some basic setup.

25.7.1 System FIFO activation

- Write 0x200 to AHB_CLK_CTRLSET1 in Syscon, enabling the clock to the FIFOs.
- Write 0x200 to PRESETCTRLSET1 in Syscon, resetting the FIFOs.
- Write 0x200 to PRESETCTRLCLR1 in Syscon, releasing reset to the FIFOs.
- Write 0x3F3F to FIFOCTRL in Syscon, enabling FIFO DMA support for all USARTs and SPIs.

25.7.2 FIFO setup for USARTs

These steps will prepare the USART FIFOs, with the FIFO space evenly allocated among the 4 USARTs (4 FIFO entries for each USART receiver, and 4 FIFO entries for each USART transmitter):

- Write 0x303 to FIFOCFGUSART0, selecting sizes for the USART0 RX and TX FIFOs.
- Write 0x303 to FIFOCFGUSART1, selecting sizes for the USART1 RX and TX FIFOs.
- Write 0x303 to FIFOCFGUSART2, selecting sizes for the USART2 RX and TX FIFOs.
- Write 0x303 to FIFOCFGUSART3, selecting sizes for the USART3 RX and TX FIFOs.
- Write 0xF 000F to FIFOUPTATEUSART, causing the selected USART FIFO sizes to go into effect.
- Write 0 to FIFOCTRLUSART, un-pausing all USART FIFOs.
- Continue to setup USART clocking, and each USART individually.

25.7.3 FIFO setup for SPIs

These steps will prepare the SPI FIFOs, with the FIFO space evenly allocated among the 2 SPIs (4 FIFO entries for each SPI receiver, and 4 FIFO entries for each SPI transmitter):

- Write 0x303 to FIFOCFGSPI0, selecting sizes for the SPI0 RX and TX FIFOs.
- Write 0x303 to FIFOCFGSPI1, selecting sizes for the SPI1 RX and TX FIFOs.
- Write 0xF 000F to FIFOUPTATESPI, causing the selected SPI FIFO sizes to go into effect.
- Write 0 to FIFOCTRLSPI, un-pausing all SPI FIFOs.
- Continue to setup SPI clocking, and each SPI individually.

26.1 How to read this chapter

The ADC controller is available on all parts. The number of ADC channels available is dependent on the package size.

26.2 Features

- 12-bit successive approximation analog to digital converter.
- Input multiplexing among up to 12 pins.
- Two configurable conversion sequences with independent triggers.
- Optional automatic high/low threshold comparison and “zero crossing” detection.
- Measurement range V_{REFN} to V_{REFP} (typically 3 V; not to exceed V_{DDA} voltage level).
- 12-bit conversion rate of 5 MHz. Options for reduced resolution at higher conversion rates.
- Burst conversion mode for single or multiple inputs.
- Synchronous or asynchronous operation. Asynchronous operation maximizes flexibility in choosing the ADC clock frequency, Synchronous mode minimizes trigger latency and can eliminate uncertainty and jitter in response to a trigger.

26.3 Basic configuration

The ADC may be utilized entirely via the ADC API (available from NXP), or by directly controlling the registers of the ADC.

Configure the ADC as follows:

- Clear the PDEN_ADC0 bit in the PDRUNCFG register ([Table 72](#)) in order to enable the analog portion of the ADC.
- Set the ADC0 bit in the AHBCLKCTRL0 register ([Table 51](#)) to enable the clock to the ADC0 register interface and the ADC0 clock.
- Clear the ADC0 peripheral reset using the PRESETCTRL0 register ([Table 35](#)).
- The ADC block creates four interrupts which are connected to slot #26 (ADC0_SEQA), slot #27 (ADC0_SEQB), slot #28 (ADC0_THCMP and ADC0_OVR) in the NVIC. The sequence interrupts can also be configured as DMA triggers through the INPUT MUX (see [Table 128](#)) for each DMA channel and as inputs to the SCT.
- Use IOCON ([Chapter 8](#)) to connect and enable analog function on the ADC input pins.
- Calibration is required every time the ADC is powered off, unless offset error is not a concern and the BYPASSCAL bit in the CTRL register is set. Calibration would be done (at a minimum) after power-up or wake-up from a reduced power mode where the ADC was turned off to save power. Re-calibration is also recommended if the temperature or voltage operating conditions have changed (including if the chip has

been in a low-power sleep mode for a considerable period of time). Re-calibration should also be performed if the ADC clock rate is changed. To perform a calibration, use the ADC API.

- There are two options in the ADC CTRL register to clock ADC conversions:
 - Use the system clock to clock the ADC in synchronous mode. This option allows exact timing of triggers but requires a system clock of 80 MHz to obtain the full ADC conversion speed.
 - Use the ADC clock, determined by the ADCCLKSEL register ([Table 47](#)) and the ADCCLKDIV register ([Table 59](#)). Some clock sources are independent of the system clock, and may require extra time to synchronize ADC trigger inputs.

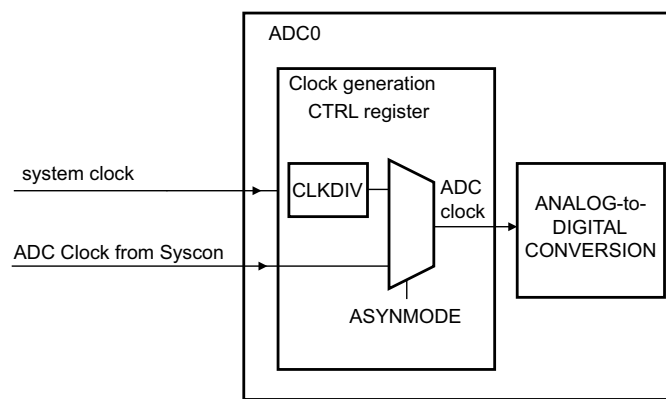


Fig 55. ADC clocking

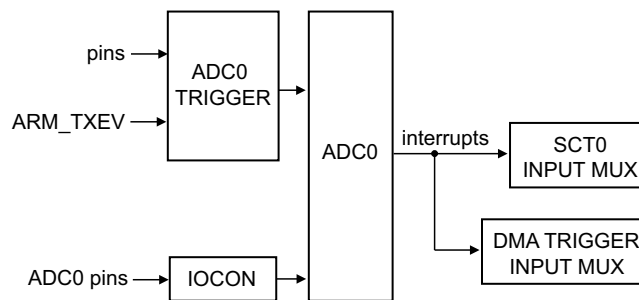


Fig 56. ADC connections

26.4 Pin description

The ADC can measure the voltage on any of the input signals on the analog input channel. Digital signals must be disconnected from the ADC input pins when the ADC function is to be used by setting DIGIMODE = 0 on those pins in the related IOCON registers.

Warning: If the ADC is used, signal levels on analog input pins must not be above the level of V_{DDA} at any time. Otherwise, ADC readings will be invalid. If the ADC is not used in an application, then the pins associated with ADC inputs can be used as digital I/O pins.

The ADC pin triggers are movable (digital) functions. To use external pin triggers for ADC conversions, assign the pin triggers to a pin via IOCON. In addition to assigning the pin triggers to a pin, they must also be selected in the conversion sequence registers for each ADC conversion sequence defined.

The V_{REFP} and V_{REFN} pins provide a positive and negative reference voltage input. The result of the conversion is $(4095 \times \text{input voltage } V_{IN}) / (V_{REFP} - V_{REFN})$. The result of an input voltage below V_{REFN} is 0, and the result of an input voltage above V_{REFP} is 4095 (0xFFFF).

Analog Power and Ground should typically be the same voltages as V_{DD} and V_{SS} , but should be isolated to minimize noise and error. If the ADC is not used, V_{DDA} and V_{REFP} should be tied to V_{DD} , and V_{SSA} and V_{REFN} should be tied to V_{SS} .

Recommended IOCON settings are shown in [Table 411](#).

Table 409. ADC common supply and reference pins

| Pin | Description |
|-------|---|
| VDDA | Analog supply voltage. V_{REFP} must not exceed the voltage level on V_{DDA} . This pin should be tied to V_{DD} (not left floating) if the ADC is not used. Remark: The supply voltage V_{DD} must be equal or lower than V_{DDA} . |
| VSSA | Analog ground. This pin should be tied to V_{SS} (not left floating) if the ADC is not used. |
| VREFP | Positive reference voltage. To operate the ADC within specifications at the maximum sampling rate, ensure that $V_{REFP} = V_{DDA}$. This pin should be tied to V_{DD} (not left floating) if the ADC is not used. Remark: Remark: Note that V_{REFP} is internally connected (not separately pinned) with V_{DDA} for some packages/part numbers. |
| VREFN | Negative reference voltage. The voltage level should typically be equal V_{SS} and V_{SSA} . This pin should be tied to V_{SS} (not left floating) if the ADC is not used. Remark: Note that V_{REFN} is internally connected (not separately pinned) with V_{SSA} for some packages/part numbers. |

Table 410. ADC0 pin description

| Function | Connect to | Description |
|----------|------------|-------------------------|
| ADC0_0 | PIO0_29 | Analog input channel 0. |
| ADC0_1 | PIO0_30 | Analog input channel 1. |
| ADC0_2 | PIO0_31 | Analog input channel 2. |
| ADC0_3 | PIO1_0 | Analog input channel 3. |
| ADC0_4 | PIO1_1 | Analog input channel 4. |
| ADC0_5 | PIO1_2 | Analog input channel 5. |

Table 410. ADC0 pin description

| Function | Connect to | Description |
|---------------|------------|--|
| ADC0_6 | PIO1_3 | Analog input channel 6. |
| ADC0_7 | PIO1_4 | Analog input channel 7. |
| ADC0_8 | PIO1_5 | Analog input channel 8. |
| ADC0_9 | PIO1_6 | Analog input channel 9. |
| ADC0_10 | PIO1_7 | Analog input channel 10. |
| ADC0_11 | PIO1_8 | Analog input channel 11. |
| ADC0_PINTRIG0 | PINT0 | ADC0 pin trigger input 0, from Pin Interrupt 0. Select in SEQA_CTRL or SEQB_CTRL register. |
| ADC0_PINTRIG1 | PINT1 | ADC0 pin trigger input 1, from Pin Interrupt 1. Select in SEQA_CTRL or SEQB_CTRL register. |

Table 411: Suggested ADC input pin settings

| IOCON bit(s) | Type D pin | Type A pin | Type I pin |
|-----------------|-------------------------|--|-------------------------|
| 10 | NA | OD: Set to 0. | NA |
| 9 | NA | Not used, set to 0. | NA |
| 8 | NA | FILTEROFF: Set to 1. | NA |
| 7 | NA | DIGIMODE: Set to 0. | NA |
| 6 | NA | INVERT: Set to 0. | NA |
| 5 | NA | Not used, set to 0. | NA |
| 4:3 | NA | MODE: Set to 00. | NA |
| 2:0 | NA | FUNC: Select GPIO as the pin function. | NA |
| General comment | Not applicable for ADC. | Only potential choice for ADC inputs. | Not applicable for ADC. |

26.5 General description

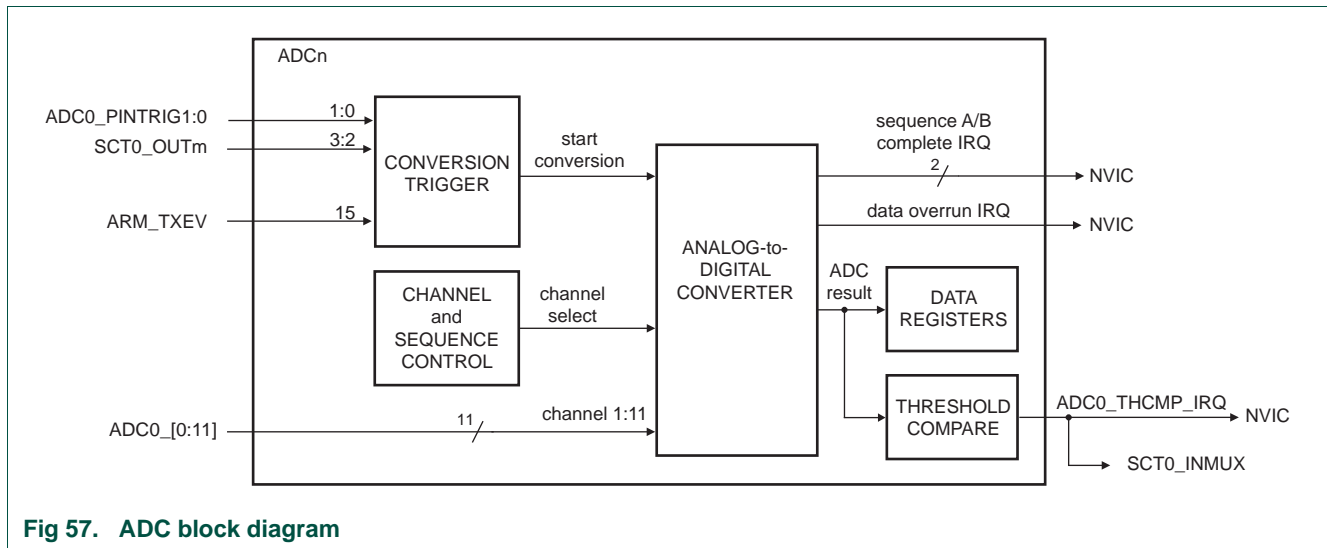


Fig 57. ADC block diagram

The ADC controller provides a great deal of flexibility in launching and controlling sequences of ADC conversions using the associated 12-bit, successive approximation ADC converter. ADC conversion sequences can be initiated under software control or in response to a selected hardware trigger.

Once the triggers are set up (software and hardware triggers can be mixed), the ADC runs through the pre-defined conversion sequences converting a sample whenever a trigger signal arrives until the sequence is disabled.

The ADC controller uses the system clock as a bus clock. The system clock or the asynchronous ADC clock (see [Figure 55](#)) can be used to create the ADC clock which drives the successive approximation process:

- In the synchronous operating mode, this ADC clock is derived from the system clock. In this mode, a programmable divider is included to scale the system clock to the maximum ADC clock rate of 80 MHz.
- In the asynchronous mode, an independent clock source is used as the ADC clock source without any further divider in the ADC. The maximum ADC clock rate is 80 MHz as well. **In this mode, the ADC clock frequency must not exceed ten times the system clock.**

A fully accurate conversion requires 15 ADC clocks.

26.6 Register description

The reset value reflects the data stored in used bits only. It does not include reserved bits content.

Table 412. Register overview: ADC (base address 0x1C03 4000)

| Name | Access | Address offset | Description | Reset value | Reference |
|-----------|--------|----------------|--|-------------|---------------------------|
| CTRL | R/W | 0x000 | ADC Control Register. Contains the clock divide value, resolution selection, sampling time selection, and mode controls. | 0x600 | Table 413 |
| SEQA_CTRL | R/W | 0x008 | ADC Conversion Sequence-A control Register: Controls triggering and channel selection for conversion sequence-A. Also specifies interrupt mode for sequence-A. | 0 | Table 414 |
| SEQB_CTRL | R/W | 0x00C | ADC Conversion Sequence-B Control Register: Controls triggering and channel selection for conversion sequence-B. Also specifies interrupt mode for sequence-B. | 0 | Table 415 |
| SEQA_GDAT | RO | 0x010 | ADC Sequence-A Global Data Register. This register contains the result of the most recent ADC conversion performed under sequence-A. | NA | Table 416 |
| SEQB_GDAT | RO | 0x014 | ADC Sequence-B Global Data Register. This register contains the result of the most recent ADC conversion performed under sequence-B. | NA | Table 417 |
| DAT0 | RO | 0x020 | ADC Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 0. | NA | Table 419 |
| DAT1 | RO | 0x024 | ADC Channel 1 Data Register. This register contains the result of the most recent conversion completed on channel 1. | NA | Table 419 |
| DAT2 | RO | 0x028 | ADC Channel 2 Data Register. This register contains the result of the most recent conversion completed on channel 2. | NA | Table 419 |
| DAT3 | RO | 0x02C | ADC Channel 3 Data Register. This register contains the result of the most recent conversion completed on channel 3. | NA | Table 419 |
| DAT4 | RO | 0x030 | ADC Channel 4 Data Register. This register contains the result of the most recent conversion completed on channel 4. | NA | Table 419 |
| DAT5 | RO | 0x034 | ADC Channel 5 Data Register. This register contains the result of the most recent conversion completed on channel 5. | NA | Table 419 |
| DAT6 | RO | 0x038 | ADC Channel 6 Data Register. This register contains the result of the most recent conversion completed on channel 6. | NA | Table 419 |
| DAT7 | RO | 0x03C | ADC Channel 7 Data Register. This register contains the result of the most recent conversion completed on channel 7. | NA | Table 419 |
| DAT8 | RO | 0x040 | ADC Channel 8 Data Register. This register contains the result of the most recent conversion completed on channel 7. | NA | Table 419 |
| DAT9 | RO | 0x044 | ADC Channel 9 Data Register. This register contains the result of the most recent conversion completed on channel 7. | NA | Table 419 |
| DAT10 | RO | 0x048 | ADC Channel 10 Data Register. This register contains the result of the most recent conversion completed on channel 7. | NA | Table 419 |
| DAT11 | RO | 0x04C | ADC Channel 11 Data Register. This register contains the result of the most recent conversion completed on channel 7. | NA | Table 419 |
| THR0_LOW | R/W | 0x050 | ADC Low Compare Threshold Register 0: Contains the lower threshold level for automatic threshold comparison for any channels linked to threshold pair 0. | 0x0 | Table 420 |

Table 412. Register overview: ADC (base address 0x1C03 4000)

| Name | Access | Address offset | Description | Reset value | Reference |
|-------------|--------|----------------|--|-------------|---------------------------|
| THR1_LOW | R/W | 0x054 | ADC Low Compare Threshold Register 1: Contains the lower threshold level for automatic threshold comparison for any channels linked to threshold pair 1. | 0 | Table 421 |
| THR0_HIGH | R/W | 0x058 | ADC High Compare Threshold Register 0: Contains the upper threshold level for automatic threshold comparison for any channels linked to threshold pair 0. | 0 | Table 422 |
| THR1_HIGH | R/W | 0x05C | ADC High Compare Threshold Register 1: Contains the upper threshold level for automatic threshold comparison for any channels linked to threshold pair 1. | 0 | Table 423 |
| CHAN_THRSEL | R/W | 0x060 | ADC Channel-Threshold Select Register. Specifies which set of threshold compare registers are to be used for each channel | 0 | Table 424 |
| INTEN | R/W | 0x064 | ADC Interrupt Enable Register. This register contains enable bits that enable the sequence-A, sequence-B, threshold compare and data overrun interrupts to be generated. | 0 | Table 425 |
| FLAGS | RO | 0x068 | ADC Flags Register. Contains the four interrupt/DMA trigger flags and the individual component overrun and threshold-compare flags. (The overrun bits replicate information stored in the result registers). | 0 | Table 426 |
| STARTUP | R/W | 0x6C | ADC Startup Register (typically only used by the ADC API). | 0 | Table 427 |
| CALIB | R/W/RO | 0x70 | ADC Calibration Register. | 0 | Table 428 |

26.6.1 ADC Control Register

This register specifies the clock divider value to be used to generate the ADC clock in **synchronous mode** and general operating mode bits including resolution and sampling time.

Table 413. ADC Control Register (CTRL, address offset 0x0) bit description

| Bit | Symbol | Value | Description | Reset value |
|------|----------|-------|---|-------------|
| 7:0 | CLKDIV | | <p>In synchronous mode only, the system clock is divided by this value plus one to produce the clock for the ADC converter, which should be less than or equal to 80 MHz.</p> <p>Typically, software should program the smallest value in this field that yields this maximum clock rate or slightly less, but in certain cases (such as a high-impedance analog source) a slower clock may be desirable.</p> <p>Remark: This field is ignored in the asynchronous operating mode.</p> | 0 |
| 8 | ASYNMODE | | Select clock mode. | 0 |
| | | 0 | <p>Synchronous mode. The ADC clock is derived from the system clock based on the divide value selected in the CLKDIV field. The ADC clock will be started in a controlled fashion in response to a trigger to eliminate any uncertainty in the launching of an ADC conversion in response to any synchronous (on-chip) trigger.</p> <p>In Synchronous mode with the SYNCPASS bit (in a sequence control register) set, sampling of the ADC input and start of conversion will initiate 2 system clocks after the leading edge of a (synchronous) trigger pulse.</p> | |
| | | 1 | Asynchronous mode. The ADC clock is based on the output of the ADC clock divider ADCCLKSEL in the SYSCON block. | |
| 10:9 | RESOL | | <p>The number of bits of ADC resolution. Accuracy can be reduced to achieve higher conversion rates. A single conversion (including one conversion in a burst or sequence) requires the selected number of bits of resolution plus 3 ADC clocks.</p> <p>Remark: This field must only be altered when the ADC is fully idle. Changing it during any kind of ADC operation may have unpredictable results.</p> <p>Remark: ADC clock frequencies for various resolutions must not exceed:</p> <ul style="list-style-type: none"> - 5x the system clock rate for 12-bit resolution - 4.3x the system clock rate for 10-bit resolution - 3.6x the system clock for 8-bit resolution - 3x the bus clock rate for 6-bit resolution | 0 |
| | | 0x0 | 6-bit resolution. An ADC conversion requires 9 ADC clocks, plus any clocks specified by the TSAMP field. | |
| | | 0x1 | 8-bit resolution. An ADC conversion requires 11 ADC clocks, plus any clocks specified by the TSAMP field. | |
| | | 0x2 | 10-bit resolution. An ADC conversion requires 13 ADC clocks, plus any clocks specified by the TSAMP field. | |
| | | 0x3 | 12-bit resolution. An ADC conversion requires 15 ADC clocks, plus any clocks specified by the TSAMP field. | |

Table 413. ADC Control Register (CTRL, address offset 0x0) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|-----------|-------|--|-------------|
| 11 | BYPASSCAL | | Bypass Calibration. This bit may be set to avoid the need to calibrate if offset error is not a concern in the application. | 0 |
| | | 0 | Calibrate. The stored calibration value will be applied to the ADC during conversions to compensated for offset error. A calibration cycle must be performed each time the chip is powered-up. Re-calibration may be warranted periodically - especially if operating conditions have changed. | |
| | | 1 | Bypass calibration. Calibration is not utilized. Less time is required when enabling the ADC - particularly following chip power-up. Attempts to launch a calibration cycle are blocked when this bit is set. | |
| 14:12 | TSAMP | | <p>Sample Time. The default sampling period (TSAMP = "000") at the start of each conversion is 2.5 ADC clock periods. Depending on a variety of factors, including operating conditions and the output impedance of the analog source, longer sampling times may be required. See Section 26.7.10.</p> <p>The TSAMP field specifies the number of additional ADC clock cycles, from zero to seven, by which the sample period will be extended. The total conversion time will increase by the same number of clocks.</p> <p>000 - The sample period will be the default 2.5 ADC clocks. A complete conversion with 12-bits of accuracy will require 15 clocks.</p> <p>001- The sample period will be extended by one ADC clock to a total of 3.5 clock periods. A complete 12-bit conversion will require 16 clocks.</p> <p>010 - The sample period will be extended by two clocks to 4.5 ADC clock cycles. A complete 12-bit conversion will require 17 ADC clocks.</p> <p>:</p> <p>111 - The sample period will be extended by seven clocks to 9.5 ADC clock cycles. A complete 12-bit conversion will require 22 ADC clocks.</p> | 0 |
| 31:15 | | | Reserved. Read value is undefined, only zero should be written. | 0 |

26.6.2 ADC Conversion Sequence A Control Register

There are two independent conversion sequences that can be configured, each consisting of a set of conversions on one or more channels. This control register specifies the channel selection and trigger conditions for the A sequence and contains bits to allow software to initiate that conversion sequence.

Remark: Set the BURST and SEQU_ENA bits at the same time.

Table 414: ADC Conversion Sequence A Control Register (SEQA_CTRL, address offset 0x08) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|------------|-------|---|-------------|
| 11:0 | CHANNELS | | <p>Selects which one or more of the ADC channels will be sampled and converted when this sequence is launched. A 1 in any bit of this field will cause the corresponding channel to be included in the conversion sequence, where bit 0 corresponds to channel 0, bit 1 to channel 1 and so forth.</p> <p>When this conversion sequence is triggered, either by a hardware trigger or via software command, ADC conversions will be performed on each enabled channel, in sequence, beginning with the lowest-ordered channel.</p> <p>Remark: This field can ONLY be changed while SEQA_ENA (bit 31) is LOW. It is allowed to change this field and set bit 31 in the same write.</p> | 0x00 |
| 17:12 | TRIGGER | | <p>Selects which of the available hardware trigger sources will cause this conversion sequence to be initiated. Program the trigger input number in this field. See Table 429.</p> <p>Remark: In order to avoid generating a spurious trigger, it is recommended writing to this field only when SEQA_ENA (bit 31) is low. It is safe to change this field and set bit 31 in the same write.</p> | 0x0 |
| 18 | TRIGPOL | | <p>Select the polarity of the selected input trigger for this conversion sequence.</p> <p>Remark: In order to avoid generating a spurious trigger, it is recommended writing to this field only when SEQA_ENA (bit 31) is low. It is safe to change this field and set bit 31 in the same write.</p> | 0 |
| | | 0 | Negative edge. A negative edge launches the conversion sequence on the selected trigger input. | |
| | | 1 | Positive edge. A positive edge launches the conversion sequence on the selected trigger input. | |
| 19 | SYNCBYPASS | | <p>Setting this bit allows the hardware trigger input to bypass synchronization flip-flop stages and therefore shorten the time between the trigger input signal and the start of a conversion. There are slightly different criteria for whether or not this bit can be set depending on the clock operating mode:</p> <p>Synchronous mode (the ASYNMODE in the CTRL register = 0): Synchronization may be bypassed (this bit may be set) if the selected trigger source is already synchronous with the main system clock (eg. coming from an on-chip, system-clock-based timer). Whether this bit is set or not, a trigger pulse must be maintained for at least one system clock period.</p> <p>Asynchronous mode (the ASYNMODE in the CTRL register = 1): Synchronization may be bypassed (this bit may be set) if it is certain that the duration of a trigger input pulse will be at least one cycle of the ADC clock (regardless of whether the trigger comes from an on-chip or off-chip source). If this bit is NOT set, the trigger pulse must at least be maintained for one system clock period.</p> | 0 |
| | | 0 | Enable trigger synchronization. The hardware trigger bypass is not enabled. | |
| | | 1 | Bypass trigger synchronization. The hardware trigger bypass is enabled. | |
| 25:20 | - | | Reserved. Read value is undefined, only zero should be written. | N/A |

Table 414: ADC Conversion Sequence A Control Register (SEQA_CTRL, address offset 0x08) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|------------|-------|--|-------------|
| 26 | START | | <p>Writing a 1 to this field will launch one pass through this conversion sequence. The behavior will be identical to a sequence triggered by a hardware trigger. Do not write 1 to this bit if the BURST bit is set.</p> <p>Remark: This bit is only set to a 1 momentarily when written to launch a conversion sequence. It will consequently always read back as a zero.</p> | 0 |
| 27 | BURST | | <p>Writing a 1 to this bit will cause this conversion sequence to be continuously cycled through. Other sequence A triggers will be ignored while this bit is set. Repeated conversions can be halted by clearing this bit. The sequence currently in progress will be completed before conversions are terminated. Note that a new sequence could begin just before BURST is cleared.</p> | 0 |
| 28 | SINGLESTEP | | <p>When this bit is set, a hardware trigger or a write to the START bit will launch a single conversion on the next channel in the sequence instead of the default response of launching an entire sequence of conversions. Once all of the channels comprising a sequence have been converted, a subsequent trigger will repeat the sequence beginning with the first enabled channel.</p> <p>Interrupt generation will still occur either after each individual conversion or at the end of the entire sequence, depending on the state of the MODE bit.</p> | 0 |
| 29 | LOWPRIO | 0 | Low priority. Any B trigger which occurs while an A conversion sequence is active will be ignored and lost. | 0 |
| | | 1 | <p>High priority. Setting this bit to a 1 will permit any enabled B sequence trigger (including a B sequence software start) to immediately interrupt sequence A and launch a B sequence in it's place. The conversion currently in progress will be terminated.</p> <p>The A sequence that was interrupted will automatically resume after the B sequence completes. The channel whose conversion was terminated will be re-sampled and the conversion sequence will resume from that point.</p> | |
| 30 | MODE | 0 | <p>Indicates whether the primary method for retrieving conversion results for this sequence will be accomplished via reading the global data register (SEQA_GDAT) at the end of each conversion, or the individual channel result registers at the end of the entire sequence.</p> <p>Impacts when conversion-complete interrupt/DMA trigger for sequence-A will be generated and which overrun conditions contribute to an overrun interrupt as described below.</p> | 0 |
| | | 0 | <p>End of conversion. The sequence A interrupt/DMA trigger will be set at the end of each individual ADC conversion performed under sequence A. This flag will mirror the DATAVALID bit in the SEQA_GDAT register.</p> <p>The OVERRUN bit in the SEQA_GDAT register will contribute to generation of an overrun interrupt/DMA trigger if enabled.</p> | |
| | | 1 | <p>End of sequence. The sequence A interrupt/DMA trigger will be set when the entire set of sequence-A conversions completes. This flag will need to be explicitly cleared by software or by the DMA-clear signal in this mode.</p> <p>The OVERRUN bit in the SEQA_GDAT register will NOT contribute to generation of an overrun interrupt/DMA trigger since it is assumed this register may not be utilized in this mode.</p> | |

Table 414: ADC Conversion Sequence A Control Register (SEQA_CTRL, address offset 0x08) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|----------|-------|--|-------------|
| 31 | SEQA_ENA | | Sequence Enable. In order to avoid spuriously triggering the sequence, care should be taken to only set the SEQA_ENA bit when the selected trigger input is in its INACTIVE state (as defined by the TRIGPOL bit). If this condition is not met, the sequence will be triggered immediately upon being enabled. Remark: In order to avoid spuriously triggering the sequence, care should be taken to only set the SEQA_ENA bit when the selected trigger input is in its INACTIVE state (as defined by the TRIGPOL bit). If this condition is not met, the sequence will be triggered immediately upon being enabled. | 0 |
| | | 0 | Disabled. Sequence A is disabled. Sequence A triggers are ignored. If this bit is cleared while sequence A is in progress, the sequence will be halted at the end of the current conversion. After the sequence is re-enabled, a new trigger will be required to restart the sequence beginning with the next enabled channel. | |
| | | 1 | Enabled. Sequence A is enabled. | |

26.6.3 ADC Conversion Sequence B Control Register

There are two independent conversion sequences that can be configured, each consisting of a set of conversions on one or more channels. This control register specifies the channel selection and trigger conditions for the B sequence, as well bits to allow software to initiate that conversion sequence.

Table 415: ADC Conversion Sequence B Control Register (SEQB_CTRL, address offset 0x0C) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|------------|-------|---|-------------|
| 11:0 | CHANNELS | | <p>Selects which one or more of the ADC channels will be sampled and converted when this sequence is launched. A 1 in any bit of this field will cause the corresponding channel to be included in the conversion sequence, where bit 0 corresponds to channel 0, bit 1 to channel 1 and so forth.</p> <p>When this conversion sequence is triggered, either by a hardware trigger or via software command, ADC conversions will be performed on each enabled channel, in sequence, beginning with the lowest-ordered channel.</p> <p>Remark: This field can ONLY be changed while SEQB_ENA (bit 31) is LOW. It is allowed to change this field and set bit 31 in the same write.</p> | 0x00 |
| 17:12 | TRIGGER | | <p>Selects which of the available hardware trigger sources will cause this conversion sequence to be initiated. Program the trigger input number in this field. See Table 429.</p> <p>Remark: In order to avoid generating a spurious trigger, it is recommended writing to this field only when SEQB_ENA (bit 31) is low. It is safe to change this field and set bit 31 in the same write.</p> | 0x0 |
| 18 | TRIGPOL | | <p>Select the polarity of the selected input trigger for this conversion sequence.</p> <p>Remark: In order to avoid generating a spurious trigger, it is recommended writing to this field only when SEQB_ENA (bit 31) is low. It is safe to change this field and set bit 31 in the same write.</p> | 0 |
| | | 0 | Negative edge. A negative edge launches the conversion sequence on the selected trigger input. | |
| | | 1 | Positive edge. A positive edge launches the conversion sequence on the selected trigger input. | |
| 19 | SYNCBYPASS | | <p>Setting this bit allows the hardware trigger input to bypass synchronization flip-flop stages and therefore shorten the time between the trigger input signal and the start of a conversion. There are slightly different criteria for whether or not this bit can be set depending on the clock operating mode:</p> <p>Synchronous mode (the ASYNMODE in the CTRL register = 0): Synchronization may be bypassed (this bit may be set) if the selected trigger source is already synchronous with the main system clock (eg. coming from an on-chip, system-clock-based timer). Whether this bit is set or not, a trigger pulse must be maintained for at least one system clock period.</p> <p>Asynchronous mode (the ASYNMODE in the CTRL register = 1): Synchronization may be bypassed (this bit may be set) if it is certain that the duration of a trigger input pulse will be at least one cycle of the ADC clock (regardless of whether the trigger comes from an on-chip or off-chip source). If this bit is NOT set, the trigger pulse must at least be maintained for one system clock period.</p> | 0 |
| | | 0 | Enable synchronization. The hardware trigger bypass is not enabled. | |
| | | 1 | Bypass synchronization. The hardware trigger bypass is enabled. | |
| 25:20 | - | | Reserved. Read value is undefined, only zero should be written. | N/A |

Table 415: ADC Conversion Sequence B Control Register (SEQB_CTRL, address offset 0x0C) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|------------|-------|---|-------------|
| 26 | START | | <p>Writing a 1 to this field will launch one pass through this conversion sequence. The behavior will be identical to a sequence triggered by a hardware trigger. Do not write 1 to this bit if the BURST bit is set.</p> <p>Remark: This bit is only set to a 1 momentarily when written to launch a conversion sequence. It will consequently always read back as a zero.</p> | 0 |
| 27 | BURST | | <p>Writing a 1 to this bit will cause this conversion sequence to be continuously cycled through. Other sequence B triggers will be ignored while this bit is set.</p> <p>Repeated conversions can be halted by clearing this bit. The sequence currently in progress will be completed before conversions are terminated.</p> | 0 |
| 28 | SINGLESTEP | | <p>When this bit is set, a hardware trigger or a write to the START bit will launch a single conversion on the next channel in the sequence instead of the default response of launching an entire sequence of conversions. Once all of the channels comprising a sequence have been converted, a subsequent trigger will repeat the sequence beginning with the first enabled channel.</p> <p>Interrupt generation will still occur either after each individual conversion or at the end of the entire sequence, depending on the state of the MODE bit.</p> | 0 |
| 29 | - | | Reserved. Read value is undefined, only zero should be written. | N/A |
| 30 | MODE | | <p>Indicates whether the primary method for retrieving conversion results for this sequence will be accomplished via reading the global data register (SEQB_GDAT) at the end of each conversion, or the individual channel result registers at the end of the entire sequence.</p> <p>Impacts when conversion-complete interrupt/DMA trigger for sequence-B will be generated and which overrun conditions contribute to an overrun interrupt as described below.</p> | 0 |
| | | 0 | <p>End of conversion. The sequence B interrupt/DMA trigger will be set at the end of each individual ADC conversion performed under sequence B. This flag will mirror the DATAVALID bit in the SEQB_GDAT register.</p> <p>The OVERRUN bit in the SEQB_GDAT register will contribute to generation of an overrun interrupt/DMA trigger if enabled.</p> | |
| | | 1 | <p>End of sequence. The sequence B interrupt/DMA trigger will be set when the entire set of sequence-B conversions completes. This flag will need to be explicitly cleared by software or by the DMA-clear signal in this mode.</p> <p>The OVERRUN bit in the SEQB_GDAT register will NOT contribute to generation of an overrun interrupt/DMA trigger since it is assumed this register may not be utilized in this mode.</p> | |
| 31 | SEQB_ENA | | <p>Sequence Enable. In order to avoid spuriously triggering the sequence, care should be taken to only set the SEQB_ENA bit when the selected trigger input is in its INACTIVE state (as defined by the TRIGPOL bit). If this condition is not met, the sequence will be triggered immediately upon being enabled.</p> <p>Remark: In order to avoid spuriously triggering the sequence, care should be taken to only set the SEQB_ENA bit when the selected trigger input is in its INACTIVE state (as defined by the TRIGPOL bit). If this condition is not met, the sequence will be triggered immediately upon being enabled.</p> | 0 |
| | | 0 | <p>Disabled. Sequence B is disabled. Sequence B triggers are ignored. If this bit is cleared while sequence B is in progress, the sequence will be halted at the end of the current conversion. After the sequence is re-enabled, a new trigger will be required to restart the sequence beginning with the next enabled channel.</p> | |
| | | 1 | <p>Enabled. Sequence B is enabled.</p> | |

26.6.4 ADC Global Data Register A and B

The ADC Global Data Registers contain the result of the most recent ADC conversion completed under each conversion sequence.

Results of ADC conversions can be read in one of two ways. One is to use these ADC Global Data Registers to read data from the ADC at the end of each ADC conversion. Another is to read the individual ADC Channel Data Registers, typically after the entire sequence has completed. It is recommended to use one method consistently for a given conversion sequence.

The global registers are useful in conjunction with DMA operation - particularly when the channels selected for conversion are not sequential (hence the addresses of the individual result registers will not be sequential, making it difficult for the DMA engine to address them). For interrupt-driven code it will more likely be advantageous to wait for an entire sequence to complete and then retrieve the results from the individual channel registers.

Remark: The method to be employed for each sequence should be reflected in the MODE bit in the corresponding SEQn_CTRL register since this will impact interrupt and overrun flag generation.

Table 416: ADC Sequence A Global Data Register (SEQA_GDAT, address offset 0x10) bit description

| Bit | Symbol | Description | Reset value |
|-------|-------------|---|-------------|
| 3:0 | - | Reserved. | NA |
| 15:4 | RESULT | This field contains the 12-bit ADC conversion result from the most recent conversion performed under conversion sequence associated with this register. The result is a binary fraction representing the voltage on the currently-selected input channel as it falls within the range of VREFP to VREFN. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on VREFN, while 0xFFF indicates that the voltage on the input was close to, equal to, or greater than that on VREFP. DATAVALID = 1 indicates that this result has not yet been read. | NA |
| 17:16 | THCMP RANGE | Indicates whether the result of the last conversion performed was above, below or within the range established by the designated threshold comparison registers (THRn_LOW and THRn_HIGH). | |
| 19:18 | THCMP CROSS | Indicates whether the result of the last conversion performed represented a crossing of the threshold level established by the designated LOW threshold comparison register (THRn_LOW) and, if so, in what direction the crossing occurred. | |
| 25:20 | - | Reserved. | NA |
| 29:26 | CHN | These bits contain the channel from which the RESULT bits were converted (e.g. 0000 identifies channel 0, 0001 channel 1, etc.). | NA |
| 30 | OVER RUN | This bit is set if a new conversion result is loaded into the RESULT field before a previous result has been read - i.e. while the DATAVALID bit is set. This bit is cleared, along with the DATAVALID bit, whenever this register is read. This bit will contribute to an overrun interrupt/DMA trigger if the MODE bit (in SEQAA_CTRL) for the corresponding sequence is set to '0' (and if the overrun interrupt is enabled). | 0 |
| 31 | DATA VALID | This bit is set to '1' at the end of each conversion when a new result is loaded into the RESULT field. It is cleared whenever this register is read. This bit will cause a conversion-complete interrupt for the corresponding sequence if the MODE bit (in SEQA_CTRL) for that sequence is set to 0 (and if the interrupt is enabled). | 0 |

Table 417: ADC Sequence B Global Data Register (SEQB_GDAT, address offset 0x14) bit description

| Bit | Symbol | Description | Reset value |
|-------|-------------|--|-------------|
| 3:0 | - | Reserved. | NA |
| 15:4 | RESULT | <p>This field contains the 12-bit ADC conversion result from the most recent conversion performed under conversion sequence associated with this register.</p> <p>The result is a binary fraction representing the voltage on the currently-selected input channel as it falls within the range of VREFP to VREFN. Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on VREFN, while 0xFFF indicates that the voltage on the input was close to, equal to, or greater than that on VREFP.</p> <p>DATAVALID = 1 indicates that this result has not yet been read.</p> | NA |
| 17:16 | THCMP RANGE | Indicates whether the result of the last conversion performed was above, below or within the range established by the designated threshold comparison registers (THRn_LOW and THRn_HIGH). | |
| 19:18 | THCMP CROSS | Indicates whether the result of the last conversion performed represented a crossing of the threshold level established by the designated LOW threshold comparison register (THRn_LOW) and, if so, in what direction the crossing occurred. | |
| 25:20 | - | Reserved. | NA |
| 29:26 | CHN | These bits contain the channel from which the RESULT bits were converted (e.g. 0000 identifies channel 0, 0001 channel 1, etc.). | NA |
| 30 | OVER RUN | <p>This bit is set if a new conversion result is loaded into the RESULT field before a previous result has been read - i.e. while the DATAVALID bit is set. This bit is cleared, along with the DATAVALID bit, whenever this register is read.</p> <p>This bit will contribute to an overrun interrupt/DMA trigger if the MODE bit (in SEQB_CTRL) for the corresponding sequence is set to '0' (and if the overrun interrupt is enabled).</p> | 0 |
| 31 | DATA VALID | <p>This bit is set to '1' at the end of each conversion when a new result is loaded into the RESULT field. It is cleared whenever this register is read.</p> <p>This bit will cause a conversion-complete interrupt for the corresponding sequence if the MODE bit (in SEQB_CTRL) for that sequence is set to 0 (and if the interrupt is enabled).</p> | 0 |

26.6.5 ADC Channel Data Registers 0 to 11

The ADC Channel Data Registers hold the result of the last conversion completed for each ADC channel. They also include status bits to indicate when a conversion has been completed, when a data overrun has occurred, and where the most recent conversion fits relative to the range dictated by the high and low threshold registers.

Results of ADC conversion can be read in one of two ways. One is to use the ADC Global Data Registers for each of the sequences to read data from the ADC at the end of each ADC conversion. Another is to use these individual ADC Channel Data Registers, typically after the entire sequence has completed. It is recommended to use one method consistently for a given conversion sequence.

Remark: The method to be employed for each sequence should be reflected in the MODE bit in the corresponding SEQ_CTRL register since this will impact interrupt and overrun flag generation.

The information presented in the DAT registers always pertains to the most recent conversion completed on that channel regardless of what sequence requested the conversion or which trigger caused it.

The OVERRUN fields for each channel are also replicated in the FLAGS register.

Table 418. Address map DAT[0:11] registers

| Peripheral | Base address | Offset | Increment | Dimension |
|------------|--------------|---------------|-----------|-----------|
| ADC | 0x1C03 2000 | [0x020:0x04C] | 0x4 | 12 |

Table 419. ADC Data Registers (DAT[0:11], address offset [0x020:0x04C]) bit description

| Bit | Symbol | Description | Reset value |
|-------|-------------|--|-------------|
| 3:0 | - | Reserved. | NA |
| 15:4 | RESULT | This field contains the 12-bit ADC conversion result from the last conversion performed on this channel. This will be a binary fraction representing the voltage on the AD0[n] pin, as it falls within the range of V_{REFP} to V_{REFN} . Zero in the field indicates that the voltage on the input pin was less than, equal to, or close to that on V_{REFN} , while 0xFFF indicates that the voltage on the input was close to, equal to, or greater than that on V_{REFP} . | NA |
| 17:16 | THCMP RANGE | Threshold Range Comparison result. 0x0 = In Range: The last completed conversion was greater than or equal to the value programmed into the designated LOW threshold register (THRn_LOW) but less than or equal to the value programmed into the designated HIGH threshold register (THRn_HIGH). 0x1 = Below Range: The last completed conversion on was less than the value programmed into the designated LOW threshold register (THRn_LOW). 0x2 = Above Range: The last completed conversion was greater than the value programmed into the designated HIGH threshold register (THRn_HIGH). 0x3 = Reserved. | NA |

Table 419. ADC Data Registers (DAT[0:11], address offset [0x020:0x04C]) bit description

| Bit | Symbol | Description | Reset value |
|-------|----------------|---|-------------|
| 19:18 | THCMP CROSS | <p>Threshold Crossing Comparison result.</p> <p>0x0 = No threshold Crossing detected: The most recent completed conversion on this channel had the same relationship (above or below) to the threshold value established by the designated LOW threshold register (THRn_LOW) as did the previous conversion on this channel.</p> <p>0x1 = Reserved.</p> <p>0x2 = Downward Threshold Crossing Detected. Indicates that a threshold crossing in the downward direction has occurred - i.e. the previous sample on this channel was above the threshold value established by the designated LOW threshold register (THRn_LOW) and the current sample is below that threshold.</p> <p>0x3 = Upward Threshold Crossing Detected. Indicates that a threshold crossing in the upward direction has occurred - i.e. the previous sample on this channel was below the threshold value established by the designated LOW threshold register (THRn_LOW) and the current sample is above that threshold.</p> | NA |
| 25:20 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 29:26 | CHANNEL | This field is hard-coded to contain the channel number that this particular register relates to (i.e. this field will contain 0b0000 for the DAT0 register, 0b0001 for the DAT1 register, etc) | NA |
| 30 | OVER RUN | <p>This bit will be set to a 1 if a new conversion on this channel completes and overwrites the previous contents of the RESULT field before it has been read - i.e. while the DONE bit is set.</p> <p>This bit is cleared, along with the DONE bit, whenever this register is read or when the data related to this channel is read from either of the global SEQn_GDAT registers.</p> <p>This bit (in any of the 12 registers) will cause an overrun interrupt/DMA trigger to be asserted if the overrun interrupt is enabled.</p> <p>Remark: While it is allowed to include the same channels in both conversion sequences, doing so may cause erratic behavior of the DONE and OVERRUN bits in the data registers associated with any of the channels that are shared between the two sequences. Any erratic OVERRUN behavior will also affect overrun interrupt generation, if enabled.</p> | NA |
| 31 | DATA VALID | <p>This bit is set to 1 when an ADC conversion on this channel completes.</p> <p>This bit is cleared whenever this register is read or when the data related to this channel is read from either of the global SEQn_GDAT registers.</p> <p>Remark: While it is allowed to include the same channels in both conversion sequences, doing so may cause erratic behavior of the DONE and OVERRUN bits in the data registers associated with any of the channels that are shared between the two sequences. Any erratic OVERRUN behavior will also affect overrun interrupt generation, if enabled.</p> | NA |

26.6.6 ADC Compare Low Threshold Registers 0 and 1

These registers set the LOW threshold levels against which ADC conversions on all channels will be compared.

Each channel will either be compared to the THR0_LOW/HIGH registers or to the THR1_LOW/HIGH registers depending on what is specified for that channel in the CHAN_THRSEL register.

A conversion result LESS THAN this value on any channel will cause the THCMP_RANGE status bits for that channel to be set to 0b01. This result will also generate an interrupt/DMA trigger if enabled to do so via the ADCMPINTEN bits associated with each channel in the INTEN register.

If, for two successive conversions on a given channel, one result is below this threshold and the other is equal-to or above this threshold, then a threshold crossing has occurred. In this case the MSB of the THCMP_CROSS status bits will indicate that a threshold crossing has occurred and the LSB will indicate the direction of the crossing. A threshold crossing event will also generate an interrupt/DMA trigger if enabled to do so via the ADCMPINTEN bits associated with each channel in the INTEN register.

Table 420. ADC Compare Low Threshold register 0 (THR0_LOW, address offset 0x50) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 3:0 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 15:4 | THRLOW | Low threshold value against which ADC results will be compared | 0x000 |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | NA |

Table 421. ADC Compare Low Threshold register 1 (THR1_LOW, address offset 0x54) bit description

| Bit | Symbol | Description | Reset value |
|-------|--------|---|-------------|
| 3:0 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 15:4 | THRLOW | Low threshold value against which ADC results will be compared | 0x000 |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | NA |

26.6.7 ADC Compare High Threshold Registers 0 and 1

These registers set the HIGH threshold level against which ADC conversions on all channels will be compared.

Each channel will either be compared to the THR0_LOW/HIGH registers or to the THR1_LOW/HIGH registers depending on what is specified for that channel in the CHAN_THRSEL register.

A conversion result greater than this value on any channel will cause the THCMP status bits for that channel to be set to 0b10. This result will also generate an interrupt/DMA trigger if enabled to do so via the ADCMPINTEN bits associated with each channel in the INTEN register.

Table 422: Compare High Threshold register0 (THR0_HIGH, address offset 0x58) bit description

| Bit | Symbol | Description | Reset value |
|-------|---------|---|-------------|
| 3:0 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 15:4 | THRHIGH | High threshold value against which ADC results will be compared | 0x000 |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | NA |

Table 423: Compare High Threshold register 1 (THR1_HIGH, address offset 0x5C) bit description

| Bit | Symbol | Description | Reset value |
|-------|---------|---|-------------|
| 3:0 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 15:4 | THRHIGH | High threshold value against which ADC results will be compared | 0x000 |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | NA |

26.6.8 ADC Channel Threshold Select register

For each channel, this register indicates which pair of threshold registers conversion results should be compared to.

Table 424: ADC Channel Threshold Select register (CHAN_THRSEL, address offset 0x60) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|-------------|-------|--|-------------|
| 0 | CH0_THRSEL | | Threshold select for channel 0. | 0 |
| | | 0 | Threshold 0. Results for this channel will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers. | |
| | | 1 | Threshold 1. Results for this channel will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers. | |
| 1 | CH1_THRSEL | | Threshold select for channel 1. See description for channel 0. | 0 |
| 2 | CH2_THRSEL | | Threshold select for channel 2. See description for channel 0. | 0 |
| 3 | CH3_THRSEL | | Threshold select for channel 3. See description for channel 0. | 0 |
| 4 | CH4_THRSEL | | Threshold select for channel 4. See description for channel 0. | 0 |
| 5 | CH5_THRSEL | | Threshold select for channel 5. See description for channel 0. | 0 |
| 6 | CH6_THRSEL | | Threshold select for channel 6. See description for channel 0. | 0 |
| 7 | CH7_THRSEL | | Threshold select for channel 7. See description for channel 0. | 0 |
| 8 | CH8_THRSEL | | Threshold select for channel 8. See description for channel 0. | 0 |
| 9 | CH9_THRSEL | | Threshold select for channel 9. See description for channel 0. | 0 |
| 10 | CH10_THRSEL | | Threshold select for channel 10. See description for channel 0. | 0 |
| 11 | CH11_THRSEL | | Threshold select for channel 11. See description for channel 0. | 0 |
| 31:12 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

26.6.9 ADC Interrupt Enable Register

There are four separate interrupt requests generated by the ADC: conversion, these are -complete or sequence-complete interrupts for each of the two sequences, a threshold-comparison out-of-range interrupt, and a data overrun interrupt. The two conversion/sequence-complete interrupts can also serve as DMA triggers. The threshold and data overrun interrupts share a slot in the NVIC.

These interrupts may be combined into one request on some chips if there is a limited number of interrupt slots. This register contains the interrupt-enable bits for each interrupt.

In this register, threshold events selected in the ADCMPINTENn bits are described as follows:

- Disabled: Threshold comparisons on channel n will not generate an ADC threshold-compare interrupt/DMA trigger.
- Outside threshold: A conversion result on channel n which is outside the range specified by the designated HIGH and LOW threshold registers will set the channel n THCMP flag in the FLAGS register and generate an ADC threshold-compare interrupt/DMA trigger.
- Crossing threshold: Detection of a threshold crossing on channel n will set the channel n THCMP flag in the FLAGS register and generate an ADC threshold-compare interrupt/DMA trigger.

Remark: Overrun and threshold-compare interrupts related to a particular channel will occur regardless of which sequence was in progress at the time the conversion was performed or what trigger caused the conversion.

Table 425: ADC Interrupt Enable register (INTEN, address offset 0x64) bit description

| Bit | Symbol | Value | Description | Reset value |
|-----|------------|-------|---|-------------|
| 0 | SEQA_INTEN | | Sequence A interrupt enable. | 0 |
| | | 0 | Disabled. The sequence A interrupt/DMA trigger is disabled. | |
| | | 1 | Enabled. The sequence A interrupt/DMA trigger is enabled and will be asserted either upon completion of each individual conversion performed as part of sequence A, or upon completion of the entire A sequence of conversions, depending on the MODE bit in the SEQA_CTRL register. | |
| 1 | SEQB_INTEN | | Sequence B interrupt enable. | 0 |
| | | 0 | Disabled. The sequence B interrupt/DMA trigger is disabled. | |
| | | 1 | Enabled. The sequence B interrupt/DMA trigger is enabled and will be asserted either upon completion of each individual conversion performed as part of sequence B, or upon completion of the entire B sequence of conversions, depending on the MODE bit in the SEQB_CTRL register. | |
| 2 | OVR_INTEN | | Overrun interrupt enable. | 0 |
| | | 0 | Disabled. The overrun interrupt is disabled. | |
| | | 1 | Enabled. The overrun interrupt is enabled. Detection of an overrun condition on any of the 12 channel data registers will cause an overrun interrupt/DMA trigger. In addition, if the MODE bit for a particular sequence is 0, then an overrun in the global data register for that sequence will also cause this interrupt/DMA trigger to be asserted. | |

Table 425: ADC Interrupt Enable register (INTEN, address offset 0x64) bit description

| Bit | Symbol | Value | Description | Reset value |
|-------|--------------|-------|--|-------------|
| 4:3 | ADCMPINTEN0 | | Threshold comparison interrupt enable for channel 0. | 00 |
| | | 0x0 | Disabled. | |
| | | 0x1 | Outside threshold. | |
| | | 0x2 | Crossing threshold. | |
| | | 0x3 | Reserved | |
| 6:5 | ADCMPINTEN1 | | Channel 1 threshold comparison interrupt enable. See description for channel 0. | 00 |
| 8:7 | ADCMPINTEN2 | | Channel 2 threshold comparison interrupt enable. See description for channel 0. | 00 |
| 10:9 | ADCMPINTEN3 | | Channel 3 threshold comparison interrupt enable. See description for channel 0. | 00 |
| 12:11 | ADCMPINTEN4 | | Channel 4 threshold comparison interrupt enable. See description for channel 0. | 00 |
| 14:13 | ADCMPINTEN5 | | Channel 5 threshold comparison interrupt enable. See description for channel 0. | 00 |
| 16:15 | ADCMPINTEN6 | | Channel 6 threshold comparison interrupt enable. See description for channel 0. | 00 |
| 18:17 | ADCMPINTEN7 | | Channel 7 threshold comparison interrupt enable. See description for channel 0. | 00 |
| 20:19 | ADCMPINTEN8 | | Channel 8 threshold comparison interrupt enable. See description for channel 0. | 00 |
| 22:21 | ADCMPINTEN9 | | Channel 9 threshold comparison interrupt enable. See description for channel 0. | 00 |
| 24:23 | ADCMPINTEN10 | | Channel 10 threshold comparison interrupt enable. See description for channel 0. | 00 |
| 26:25 | ADCMPINTEN11 | | Channel 21 threshold comparison interrupt enable. See description for channel 0. | 00 |
| 31:27 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

26.6.10 ADC Flags register

The ADC Flags registers contains the four interrupt/DMA trigger flags along with the individual overrun flags that contribute to an overrun interrupt and the component threshold-comparison flags that contribute to that interrupt. Note that the threshold and overrun interrupts have a slot in the NVIC.

The channel OVERRUN flags mirror those appearing in the individual DAT registers for each channel and indicate a data overrun in each of those registers.

Likewise, the SEQA_OVR and SEQB_OVR bits mirror the OVERRUN bits in the two global data registers (SEQA_GDAT and SEQB_GDAT).

Remark: The SEQn_INT conversion/sequence-complete flags also serve as DMA triggers.

Table 426: ADC Flags register (FLAGS, address offset 0x68) bit description

| Bit | Symbol | Description | Reset value |
|-------|-----------|--|-------------|
| 0 | THCMP0 | Threshold comparison event on Channel 0. Set to 1 upon either an out-of-range result or a threshold-crossing result if enabled to do so in the INTEN register. This bit is cleared by writing a 1. | 0 |
| 1 | THCMP1 | Threshold comparison event on Channel 1. See description for channel 0. | 0 |
| 2 | THCMP2 | Threshold comparison event on Channel 2. See description for channel 0. | 0 |
| 3 | THCMP3 | Threshold comparison event on Channel 3. See description for channel 0. | 0 |
| 4 | THCMP4 | Threshold comparison event on Channel 4. See description for channel 0. | 0 |
| 5 | THCMP5 | Threshold comparison event on Channel 5. See description for channel 0. | 0 |
| 6 | THCMP6 | Threshold comparison event on Channel 6. See description for channel 0. | 0 |
| 7 | THCMP7 | Threshold comparison event on Channel 7. See description for channel 0. | 0 |
| 8 | THCMP8 | Threshold comparison event on Channel 8. See description for channel 0. | 0 |
| 9 | THCMP9 | Threshold comparison event on Channel 9. See description for channel 0. | 0 |
| 10 | THCMP10 | Threshold comparison event on Channel 10. See description for channel 0. | 0 |
| 11 | THCMP11 | Threshold comparison event on Channel 11. See description for channel 0. | 0 |
| 12 | OVERRUN0 | Mirrors the OVERRRUN status flag from the result register for ADC channel 0 | 0 |
| 13 | OVERRUN1 | Mirrors the OVERRRUN status flag from the result register for ADC channel 1 | 0 |
| 14 | OVERRUN2 | Mirrors the OVERRRUN status flag from the result register for ADC channel 2 | 0 |
| 15 | OVERRUN3 | Mirrors the OVERRRUN status flag from the result register for ADC channel 3 | 0 |
| 16 | OVERRUN4 | Mirrors the OVERRRUN status flag from the result register for ADC channel 4 | 0 |
| 17 | OVERRUN5 | Mirrors the OVERRRUN status flag from the result register for ADC channel 5 | 0 |
| 18 | OVERRUN6 | Mirrors the OVERRRUN status flag from the result register for ADC channel 6 | 0 |
| 19 | OVERRUN7 | Mirrors the OVERRRUN status flag from the result register for ADC channel 7 | 0 |
| 20 | OVERRUN8 | Mirrors the OVERRRUN status flag from the result register for ADC channel 8 | 0 |
| 21 | OVERRUN9 | Mirrors the OVERRRUN status flag from the result register for ADC channel 9 | 0 |
| 22 | OVERRUN10 | Mirrors the OVERRRUN status flag from the result register for ADC channel 10 | 0 |
| 23 | OVERRUN11 | Mirrors the OVERRRUN status flag from the result register for ADC channel 11 | 0 |
| 24 | SEQA_OVR | Mirrors the global OVERRUN status flag in the SEQA_GDAT register | 0 |
| 25 | SEQB_OVR | Mirrors the global OVERRUN status flag in the SEQB_GDAT register | 0 |
| 27:26 | - | Reserved. | NA |

Table 426: ADC Flags register (FLAGS, address offset 0x68) bit description

| Bit | Symbol | Description | Reset value |
|-----|-----------|--|-------------|
| 28 | SEQA_INT | <p>Sequence A interrupt/DMA trigger.</p> <p>If the MODE bit in the SEQA_CTRL register is 0, this flag will mirror the DATAVALID bit in the sequence A global data register (SEQA_GDAT), which is set at the end of every ADC conversion performed as part of sequence A. It will be cleared automatically when the SEQA_GDAT register is read.</p> <p>If the MODE bit in the SEQA_CTRL register is 1, this flag will be set upon completion of an entire A sequence. In this case it must be cleared by writing a 1 to this SEQA_INT bit.</p> <p>This interrupt must be enabled in the INTEN register.</p> | 0 |
| 29 | SEQB_INT | <p>Sequence B interrupt/DMA trigger.</p> <p>If the MODE bit in the SEQB_CTRL register is 0, this flag will mirror the DATAVALID bit in the sequence B global data register (SEQB_GDAT), which is set at the end of every ADC conversion performed as part of sequence B. It will be cleared automatically when the SEQB_GDAT register is read.</p> <p>If the MODE bit in the SEQB_CTRL register is 1, this flag will be set upon completion of an entire B sequence. In this case it must be cleared by writing a 1 to this SEQB_INT bit.</p> <p>This interrupt must be enabled in the INTEN register.</p> | 0 |
| 30 | THCMP_INT | <p>Threshold Comparison Interrupt.</p> <p>This bit will be set if any of the THCMP flags in the lower bits of this register are set to 1 (due to an enabled out-of-range or threshold-crossing event on any channel).</p> <p>Each type of threshold comparison interrupt on each channel must be individually enabled in the INTEN register to cause this interrupt.</p> <p>This bit will be cleared when all of the individual threshold flags are cleared via writing 1s to those bits.</p> | 0 |
| 31 | OVR_INT | <p>Overrun Interrupt flag.</p> <p>Any overrun bit in any of the individual channel data registers will cause this interrupt. In addition, if the MODE bit in either of the SEQn_CTRL registers is 0 then the OVERRUN bit in the corresponding SEQn_GDAT register will also cause this interrupt.</p> <p>This interrupt must be enabled in the INTEN register.</p> <p>This bit will be cleared when all of the individual overrun bits have been cleared via reading the corresponding data registers.</p> | 0 |

26.6.11 ADC Startup register

This register is used exclusively when enabling the ADC, and typically only by the ADC API. This register should never be accessed during normal ADC operation. The ADC clock should be selected and running at full frequency prior to writing to this register.

Table 427: ADC Startup register (STARTUP, address offset 0x6C) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|---|-------------|
| 0 | ADC_ENA | ADC Enable bit. This bit can only be set to a 1 by software. It is cleared automatically whenever the ADC is powered down. This bit must not be set until at least 10 microseconds after the ADC is powered up (typically by altering a system-level ADC power control bit). | 0 |
| 1 | ADC_INIT | ADC Initialization. After enabling the ADC (setting the ADC_ENA bit), the API routine will EITHER set this bit or the CALIB bit in the CALIB register, depending on whether or not calibration is required. Setting this bit will launch the “dummy” conversion cycle that is required if a calibration is not performed. It will also reload the stored calibration value from a previous calibration unless the BYPASSCAL bit is set. This bit should only be set AFTER the ADC_ENA bit is set and after the CALIREQD bit is tested to determine whether a calibration or an ADC dummy conversion cycle is required. It should not be set during the same write that sets the ADC_ENA bit. This bit can only be set to a ‘1’ by software. It is cleared automatically when the “dummy” conversion cycle completes. | 0 |
| 31:2 | - | Reserved. Read value is undefined, only zero should be written. | 0 |

26.6.12 ADC Calibration register

This register is used to perform ADC offset calibration. It will be used by the ADC API. It may also be written-to subsequently by user software to initiate re-calibration. **The maximum ADC clock frequency during calibration is 30 MHz.** If the operating ADC frequency exceeds this, a slower clock should be selected for calibration (eg. increasing the synchronous divided clock value).

Table 428: ADC Calibration register (CALIB, address offset 0x70) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|--|-------------|
| 0 | CALIB | Calibration request. Setting this bit will launch an ADC calibration cycle. This bit can only be set to a ‘1’ by software. It is cleared automatically when the calibration cycle completes. | 0 |
| 1 | CALREQD | Calibration required. This read-only bit indicates if calibration is required when enabling the ADC. CALREQD will be ‘1’ if no calibration has been run since the chip was powered-up and if the BYPASSCAL bit in the CTRL register is low. The ADC API will test this bit to determine whether to initiate a calibration cycle or whether to set the ADC_INIT bit (in the STARTUP register) to launch the ADC initialization process which includes a “dummy” conversion cycle. Note: A “dummy” conversion cycle requires approximately 6 ADC clocks as opposed to 81 clocks required for calibration. | 1 |
| 8:2 | CALVALUE | Calibration Value. This read-only field displays the calibration value established during last calibration cycle. This value is not typically of any use to the user. | 0 |
| 31:9 | - | Reserved. Read value is undefined, only zero should be written. | 0 |

26.7 Functional description

26.7.1 Conversion Sequences

A conversion sequence is a single pass through a series of ADC conversions performed on a selected set of ADC channels. Software can configure up to two independent conversion sequences, either of which can be triggered by software or by a transition on one of the hardware triggers. Each sequence can be triggered by a different hardware trigger. One of these conversion sequences is referred to as the A sequence and the other as the B sequence.

An optional single-step mode allows advancing through the channels of a sequence one at a time on each successive occurrence of a trigger.

The user can select whether a trigger on the B sequence can interrupt an already in-progress A sequence. The B sequence, however, can never be interrupted by an A trigger.

26.7.2 Hardware-triggered conversion

Software can select among hardware triggers will launch each conversion sequence and it can specify the active edge for the selected trigger independently for each conversion sequence.

For each conversion sequence, if a designated trigger event occurs, one single cycle through that conversion sequence will be launched unless:

- The BURST bit in the SEQn_CTRL register for this sequence is set to 1.
- The requested conversion sequence is already in progress.
- A set of conversions for the alternate conversion sequence is already in progress except in the case of a B trigger interrupting an A sequence if the A sequence is set to LOWPRIO.

If any of these conditions is true, the new trigger event will be ignored and will have no effect.

In addition, if the single-step bit for a sequence is set, each new trigger will cause a single conversion to be performed on the next channel in the sequence rather than launching a pass through the entire sequence.

If the A sequence is enabled to be interrupted (i.e. the LOWPRIO bit in the SEQA_CTRL register is set) and a B trigger occurs while an A sequence is in progress, then the following will occur:

- The ADC conversion which is currently in-progress will be aborted.
- The A sequence will be paused, and the B sequence will immediately commence.
- The interrupted A sequence will resume after the B sequence completes, beginning with the conversion that was aborted when the interruption occurred. The channel for that conversion will be re-sampled.

26.7.2.1 Avoiding spurious hardware triggers

Care should be taken to avoid generating a spurious trigger when writing to the SEQn_CTRL register to change the trigger selected for the sequence, switch the polarity of the selected trigger, or to enable the sequence for operation.

In general, the TRIGGER and TRIGPOL bits in the SEQn_ENA bit is should only be written when the sequence is disabled (while the SEQn_ENA bit = 0). The SEQn_ENA bit itself should only be set when the selected trigger input is in its INACTIVE state (as designated by the TRIGPOL bit). If this condition is not met, a trigger will be generated immediately upon enabling the sequence - even though no actual transition has occurred on the trigger input.

26.7.3 Software-triggered conversion

There are two ways that software can trigger a conversion sequence:

1. **Start Bit:** Setting the START bit in the corresponding SEQn_CTRL register. The response to this is identical to occurrence of a hardware trigger on that sequence. Specifically, one cycle of conversions through that conversion sequence will be immediately triggered except as indicated above.
2. **Burst Mode:** Set the BURST bit in the SEQn_CTRL register. As long as this bit is 1 the designated conversion sequence will be continuously and repetitively cycled through. Any new software or hardware trigger on this sequence will be ignored.

If a bursting A sequence is allowed to be interrupted (i.e. the LOWPRIO bit in its SEQn_CTRL register is set to 1) and a software or hardware trigger for the B sequence occurs, then the burst will be immediately interrupted and a B sequence will be initiated. The interrupted A sequence will resume continuous cycling, starting with the aborted conversion, after the alternate sequence has completed.

26.7.4 Interrupts

There are four interrupts that can be generated by the ADC:

- Conversion-Complete or Sequence-Complete interrupt for sequence A
- Conversion-Complete or Sequence-Complete interrupt for sequence B
- Threshold-Compare Out-of-Range Interrupt
- Data Overrun Interrupt

Any of these interrupt requests may be individually enabled or disabled in the INTEN register. Note that the threshold and overrun interrupts share a slot in the NVIC.

26.7.4.1 Conversion-Complete or Sequence-Complete interrupts

For each of the two sequences, an interrupt/DMA trigger can either be asserted at the end of each ADC conversion performed as part of that sequence or when the entire sequence of conversions is completed. The MODE bits in the SEQn_CTRL registers select between these alternative behaviors.

If the MODE bit for a sequence is 0 (conversion-complete mode), then the interrupt flag/DMA request for that sequence will reflect the state of the DATAVALID bit in the global data register (SEQn_GDAT) for that sequence. In this case, reading the SEQn_GDAT register will automatically clear the interrupt/DMA trigger.

If the MODE bit for the sequence is 1 (sequence-complete mode) then the interrupt flag/DMA request must be written-to by software to clear it (except when used as a DMA trigger, in which case it will be cleared in hardware by the DMA engine).

26.7.4.2 Threshold-Compare Out-of-Range Interrupt

Every conversion performed on any channel is automatically compared against a designated set of low and high threshold levels specified in the THRN_HIGH and THRN_LOW registers. The results of this comparison on any individual channel(s) can be enabled to cause a threshold-compare interrupt if that result was above or below the range specified by the two thresholds or, alternatively, if the result represented a crossing of the low threshold in either direction.

This flag must be cleared by a software write to clear the individual THCMP flags in the FLAGS register.

26.7.4.3 Data Overrun Interrupt

This interrupt/DMA trigger will be asserted if any of the OVERRUN bits in the individual channel data registers are set. In addition, the OVERRUN bits in the two sequence global data (SEQn_GDAT) registers will cause this interrupt/DMA trigger IF the MODE bit for that sequence is set to 0 (conversion-complete mode).

This flag will be cleared when the OVERRUN bit that caused it is cleared via reading the register containing it.

Note that the OVERRUN bits in the individual data registers are cleared when data related to that channel is read from either of the global data registers as well as when the individual data registers themselves are read.

26.7.5 Optional Operating Modes

There are three optional modes of ADC operation which may be selected in the CTRL register.

Four alternative ADC accuracy settings are available ranging from 12 bits down to 6 bits of resolution. Lowering the ADC resolution results in faster conversion times. A single ADC conversion (including one conversion in a burst or sequence) requires (resolution+3) ADC clocks when the minimum sampling period is selected. When reduced accuracy is selected, the unused LSBs of result data will automatically be forced to zero.

Two clocking modes are available, synchronous mode and asynchronous mode. The synchronous clocking mode uses the system clock in conjunction with an internal programmable divider. The main advantage of this mode is determinism. The start of ADC sampling is always a fixed number of system clocks following any ADC trigger. The alternative asynchronous mode (on chips where this mode is supported) uses an independent clock source. In this mode the user has greater flexibility in selecting the ADC clock frequency to better achieve the maximum ADC conversion rate without restricting the clock rate for other peripherals. The penalty for using this mode may be longer latency and greater uncertainty in response to a hardware trigger.

26.7.6 Offset calibration and enabling the ADC

The A/D converter includes a built-in, self-calibration mode which can be used to minimize offset error. For applications where offset error is not a concern, calibration may be disabled by setting the BYPASSCAL bit in the CTRL register. If this bit is not set, a calibration cycle must be performed following chip power-up (including exit from Deep Sleep, Power-down, or Deep Power-down mode) prior to using the ADC. The provided ADC API will automatically perform this required calibration.

Additional calibrations may be performed at any time by setting the CALIB bit in the CALIB register. Re-calibration is recommended if the temperature or voltage operating conditions have changed (including if the chip has been in a low-power sleep mode for a considerable period of time). Re-calibration should also be performed if the ADC clock rate is changed.

A calibration cycle requires approximately 81 ADC clocks to complete. Normal ADC conversions cannot be launched, and the ADC Control Register must not be written while calibration is in progress.

Remark: Enabling the ADC (following chip power-up, exit from any low-power mode, or when the ADC has been manually disabled) requires a specific start-up procedure. It is strongly recommended that the provided "Enable-ADC" API routine be used to enable or re-enable the ADC. The API routine will perform the necessary steps - including executing a calibration cycle if required.

Important: The ADC clock must be running at its full operating frequency prior to calling the API routine to enable the ADC. This means that the desired clocking mode and clock divide value (for sync mode) must be programmed into the CTRL register. If offset calibration is not desired, the BYPASSCAL bit in the CTRL register should also be set prior to calling the API routine to avoid wasting time on an unnecessary calibration cycle.

The ADC cannot be utilized until the startup routine has completed.

26.7.7 ADC vs. digital receiver

The analog ADC input must be selected via IOCON registers in order to get accurate voltage readings on the monitored pin. In the IOCON, the pull-up and pull-down resistors should be both disabled using the MODE bits. For a pin hosting an ADC input, it is not possible to have a have the digital function enabled and yet get valid ADC readings. Software must write a 0 to the DIGIMODE bit in the related IOCON register.

26.7.8 DMA control

The sequence A or sequence B conversion sequence complete interrupts may also be used to generate a DMA transfer trigger. To generate a DMA transfer the same conditions must be met as the conditions for generating an interrupt (see [Section 26.7.4](#) and [Section 26.6.9](#)).

Remark: If DMA is used for a sequence, the corresponding sequence interrupt must be disabled in the INTEN register.

For DMA transfers, only burst requests are supported. The burst size can be set to one in the DMA channel control register. If the number of ADC channels is not equal to one of the other DMA-supported burst sizes (applicable DMA burst sizes are 1, 4, 8), set the burst size to one.

The DMA transfer size determines when a DMA interrupt is generated. The transfer size can be set to the number of ADC channels being converted. Non-contiguous channels can be transferred by the DMA using the scatter/gather linked lists.

26.7.9 ADC hardware trigger inputs

An analog-to-digital conversion can be initiated by a hardware trigger. The trigger can be selected independently for each of the two conversion sequences in the ADC SEQA_CTRL or SEQB_CTRL registers by programming the hardware trigger input # into the TRIGGER bits.

Related registers:

- [Table 414 “ADC Conversion Sequence A Control Register \(SEQA_CTRL, address offset 0x08\) bit description”](#)
- [Table 415 “ADC Conversion Sequence B Control Register \(SEQB_CTRL, address offset 0x0C\) bit description”](#)

Table 429. ADC0 hardware trigger inputs

| Input # | Source | Description |
|---------|----------|---|
| 0 | PINT0 | See Chapter 26 Group GPIO input interrupt (GINT0/1) |
| 1 | PINT1 | See Chapter 26 Group GPIO input interrupt (GINT0/1) |
| 2 | SCT_OUT7 | See Chapter 14 SCTimer/PWM |
| 4:3 | Reserved | - |
| 5 | ARM_TXEV | Transmit Event output from either CPU |

26.7.10 Sample time

The analog input from the selected channel is sampled at the start of each new A/D conversion. The default (and shortest) duration of this sample period is 2.5 ADC clock cycles. Under some conditions, longer sample times may be required. A variety of factors including operating conditions, the ADC clock frequency, the selected ADC resolution, and the impedance of the analog source will influence the required sample period.

ADC channels 6 to 11 are somewhat slower than channels 0 to 5. Lower analog-source impedances may be required for these slow channels for a given sample period and set of operating conditions.

The following table provides guidelines for the required sample times. The “TSAMP” values displayed in the tables refer to the TSAMP field in the ADCTRL Register. This value represents the number of additional clock cycles that must be added to the minimum 2.5-clock sample period in order to meet or exceed the required minimum sample time for the maximum ADC clock rate of 80 MHz under worst-case operating conditions. At slower clock frequencies fewer sample clocks will be needed to achieve the required sample times.

Table 430. Minimum required sample times

| Selected ADC Resolution | Analog signal source impedance | Fast channels (ADC5:0) | | Slow channels (ADC11:6) | |
|-------------------------|--------------------------------|------------------------|-------------|-------------------------|-------------|
| | | Min. sample time | TSAMP field | Min. sample time | TSAMP field |
| 12 bits | under 0.05k ohms | 20 ns | 0 | 43 ns | 1 |
| | 0.05 to 0.1k ohms | 23 ns | 0 | 46 ns | 1 |
| | 0.1K to 0.2k ohms | 26 ns | 0 | 50 ns | 2 |
| | 0.2k to 0.5k ohms | 31 ns | 0 | 56 ns | 2 |
| | 0.5k to 1.0k ohms | 47 ns | 1 | 74 ns | 3 |
| | 1k to 5k ohms | 75 ns | 3 | 105 ns | 6 |
| 10 bits | under 0.05k ohms | 15 ns | 0 | 35 ns | 1 |
| | 0.05 to 0.1k ohms | 18 ns | 0 | 38 ns | 1 |
| | 0.1K to 0.2k ohms | 20 ns | 0 | 40 ns | 1 |
| | 0.2k to 0.5k ohms | 24 ns | 0 | 46 ns | 1 |
| | 0.5k to 1.0k ohms | 38 ns | 1 | 61 ns | 2 |
| | 1k to 5k ohms | 62 ns | 2 | 86 ns | 4 |
| 8 bits | under 0.05k ohms | 12 ns | 0 | 27 ns | 0 |
| | 0.05 to 0.1k ohms | 13 ns | 0 | 29 ns | 0 |
| | 0.1K to 0.2k ohms | 15 ns | 0 | 32 ns | 0 |
| | 0.2k to 0.5k ohms | 19 ns | 0 | 36 ns | 1 |
| | 0.5k to 1.0k ohms | 30 ns | 0 | 48 ns | 1 |
| | 1k to 5k ohms | 48 ns | 1 | 69 ns | 3 |
| 6 bits | under 0.05k ohms | 9 ns | 0 | 20 ns | 0 |
| | 0.05 to 0.1k ohms | 10 ns | 0 | 22 ns | 0 |
| | 0.1K to 0.2k ohms | 11 ns | 0 | 23 ns | 0 |
| | 0.2k to 0.5k ohms | 13 ns | 0 | 26 ns | 0 |
| | 0.5k to 1.0k ohms | 22 ns | 0 | 36 ns | 1 |
| | 1k to 5k ohms | 36 ns | 1 | 51 ns | 2 |

26.8 Examples

26.8.1 Perform a single ADC conversion triggered by software

Remark: When ADC conversions are triggered by software only and hardware triggers are not used in the conversion sequence, follow these steps to avoid spurious conversions:

1. Before changing the trigger set-up, disable the conversion sequence by setting the SEQ_ENA bit to 0 in the SEQA_CTRL register.
2. Set the trigger source to an unused setting using the TRIGGER bits in the SEQA_CTRL register.
3. Set the TRIGPOL bit to 1 in the in the SEQA_CTRL register.

Once the sequence is enabled again, the ADC converts a sample whenever the START bit is written to.

The ADC converts an analog input signal VIN on the ADC0_[11:0] pins. The VREFP and VREFN pins provide a positive and negative reference voltage input. The result of the conversion is $(4095 \times VIN)/(VREFP - VREFN)$. The result of an input voltage below VREFN is 0, and the result of an input voltage above VREFP is 4095 (0xFFF).

To perform a single ADC conversion for ADC0 channel 1 using the analog signal on pin ADC0_1, follow these steps:

1. Enable the analog function on pin ADC0_1 via IOCON See [Table 116](#) and [Table 119](#).
2. Configure the system clock to be 50 MHz and select a CLKDIV value of 0 for a sampling rate of 50 MHz using the ADC CTRL register.
3. Select the synchronous mode in the CTRL register.
4. Select ADC channel 1 to perform the conversion by setting the CHANNELS bits to 0x2 in the SEQA_CTL register.
5. Set the TRIGPOL bit to 1 and the SEQA_ENA bit to 1 in the SEQA_CTRL register.
6. Set the START bit to 1 in the SEQA_CTRL register.
7. Read the RESULT bits in the DAT1 register for the conversion result.

26.8.2 Perform a sequence of conversions triggered by an external pin

The ADC can perform conversions on a sequence of selected channels. Each individual conversion of the sequence (single-step) or the entire sequence can be triggered by hardware. Hardware triggers are either a signal from an external pin or an internal signal. See [Section 26.7.9](#).

To perform a single-step conversion on the first four channels of ADC0 triggered by rising edges on pin PIO1_0, follow these steps:

1. Enable the analog function on pin ADC0_0 to ADC0_3 via IOCON. See [Table 116](#) and [Table 119](#).
2. Configure PINT1 to respond to PIO1_0, see [Chapter 11](#) for details.
3. Configure the system clock to be 80 MHz and select a CLKDIV value of 0 for a sampling rate of 80 MHz using the ADC CTRL register.
4. Select the synchronous mode in the CTRL register.
5. Select ADC channels 0 to 3 to perform the conversion by setting the CHANNELS bits to 0xF in the SEQA_CTRL register.
6. Select trigger PINT1 by writing 0x1 the TRIGGER bits in the SEQA_CTRL register.
7. To generate one interrupt at the end of the entire sequence, set the MODE bit to 1 in the SEQA_CTRL register.
8. Select single-step mode by setting the SINGLESTEP bit in the SEQA_CTRL register to 1.
9. Enable the Sequence A by setting the SEQA_ENA bit.

A conversion on ADC0 channel 0 will be triggered whenever the pin PIO1_0 goes from LOW to HIGH. The conversion on the next channel (initially channel 1) is triggered on the next 0 to 1 transition of PINT1. The ADC0 interrupt is generated when the sequence has finished after four 0 to 1 transitions of PINT1.

10. Read the RESULT bits in the DAT0 to DAT3 registers for the conversion result.

27.1 How to read this chapter

The CRC engine is available on all LPC5410x parts.

27.2 Features

- Supports three common polynomials CRC-CCITT, CRC-16, and CRC-32.
 - CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$
 - CRC-16: $x^{16} + x^{15} + x^2 + 1$
 - CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- Bit order reverse and 1's complement programmable setting for input data and CRC sum.
- Programmable seed number setting.
- Supports CPU PIO back-to-back transfer.
- Accept any size of data width per write: 8, 16 or 32-bit.
 - 8-bit write: 1-cycle operation
 - 16-bit write: 2-cycle operation (8-bit x 2-cycle)
 - 32-bit write: 4-cycle operation (8-bit x 4-cycle)

27.3 Basic configuration

Set the CRC bit in the AHBCLKCTRL0 register ([Table 51](#)) to enable the clock to the CRC engine.

27.4 Pin description

The CRC engine has no configurable pins.

27.5 General description

The Cyclic Redundancy Check (CRC) generator with programmable polynomial settings supports several CRC standards commonly used.

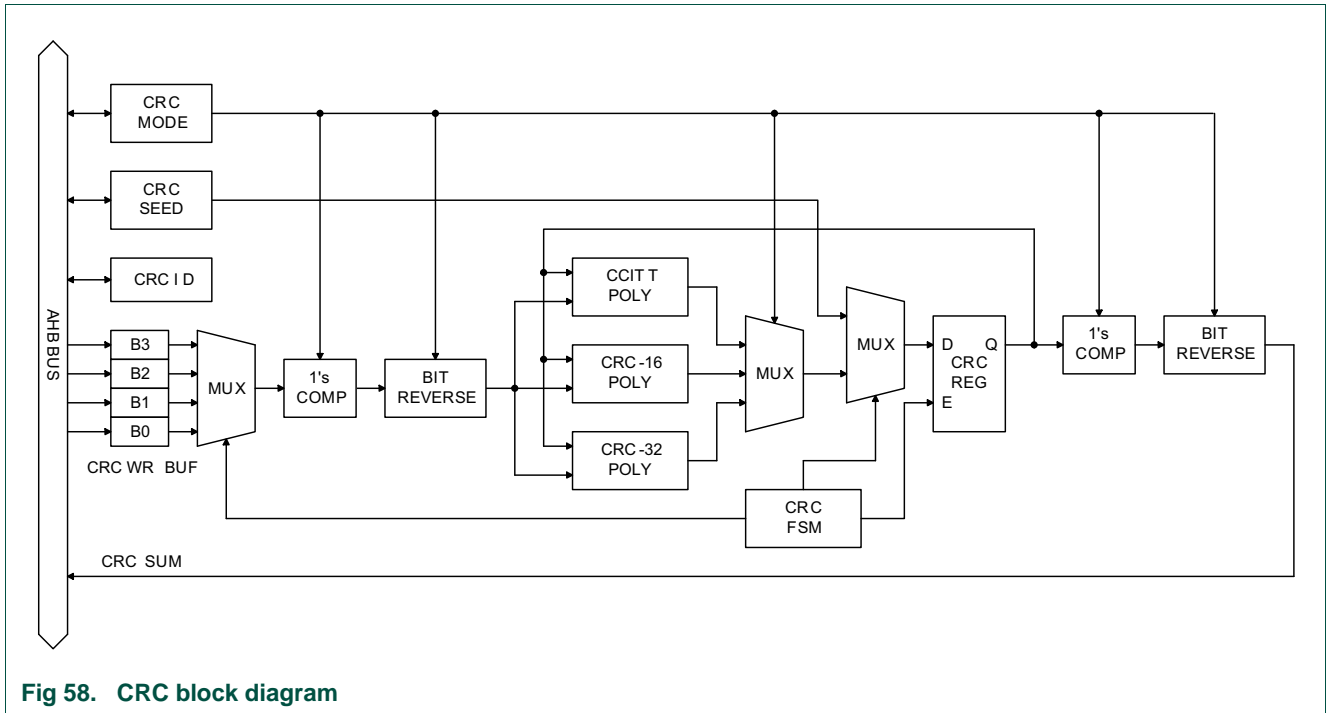


Fig 58. CRC block diagram

27.6 Register description

Table 431. Register overview: CRC engine (base address 0x1C01 0000)

| Name | Access | Address offset | Description | Reset value | Reference |
|---------|--------|----------------|-----------------------|-------------|---------------------------|
| MODE | R/W | 0x000 | CRC mode register | 0x0000 0000 | Table 432 |
| SEED | R/W | 0x004 | CRC seed register | 0x0000 FFFF | Table 433 |
| SUM | RO | 0x008 | CRC checksum register | 0x0000 FFFF | Table 434 |
| WR_DATA | WO | 0x008 | CRC data register | - | Table 435 |

27.6.1 CRC mode register

Table 432. CRC mode register (MODE, address 0x1C01 0000) bit description

| Bit | Symbol | Description | Reset value |
|------|-------------|--|-------------|
| 1:0 | CRC_POLY | CRC polynom: 1X= CRC-32 polynomial 01= CRC-16 polynomial 00= CRC-CCITT polynomial | 00 |
| 2 | BIT_RVS_WR | Data bit order: 1= Bit order reverse for CRC_WR_DATA (per byte) 0= No bit order reverse for CRC_WR_DATA (per byte) | 0 |
| 3 | CMPL_WR | Data complement: 1= 1's complement for CRC_WR_DATA 0= No 1's complement for CRC_WR_DATA | 0 |
| 4 | BIT_RVS_SUM | CRC sum bit order: 1= Bit order reverse for CRC_SUM 0= No bit order reverse for CRC_SUM | 0 |
| 5 | CMPL_SUM | CRC sum complement: 1= 1's complement for CRC_SUM 0=No 1's complement for CRC_SUM | 0 |
| 31:6 | Reserved | Always 0 when read | 0x0000000 |

27.6.2 CRC seed register

Table 433. CRC seed register (SEED, address 0x1C01 0004) bit description

| Bit | Symbol | Description | Reset value |
|------|----------|---|-------------|
| 31:0 | CRC_SEED | A write access to this register will load CRC seed value to CRC_SUM register with selected bit order and 1's complement pre-processes. Remark: A write access to this register will overrule the CRC calculation in progresses. | 0x0000 FFFF |

27.6.3 CRC checksum register

This register is a Read-only register containing the most recent checksum. The read request to this register is automatically delayed by a finite number of wait states until the results are valid and the checksum computation is complete.

Table 434. CRC checksum register (SUM, address 0x1C01 0008) bit description

| Bit | Symbol | Description | Reset value |
|------|---------|--|-------------|
| 31:0 | CRC_SUM | The most recent CRC sum can be read through this register with selected bit order and 1's complement post-processes. | 0x0000 FFFF |

27.6.4 CRC data register

This register is a Write-only register containing the data block for which the CRC sum will be calculated.

Table 435. CRC data register (WR_DATA, address 0x1C01 0008) bit description

| Bit | Symbol | Description | Reset value |
|------|-------------|---|-------------|
| 31:0 | CRC_WR_DATA | Data written to this register will be taken to perform CRC calculation with selected bit order and 1's complement pre-process. Any write size 8, 16 or 32-bit are allowed and accept back-to-back transactions. | - |

27.7 Functional description

The following sections describe the register settings for each supported CRC standard:

27.7.1 CRC-CCITT set-up

Polynomial = $x^{16} + x^{12} + x^5 + 1$
Seed Value = 0xFFFF
Bit order reverse for data input: NO
1's complement for data input: NO
Bit order reverse for CRC sum: NO
1's complement for CRC sum: NO
CRC_MODE = 0x0000 0000
CRC_SEED = 0x0000 FFFF

27.7.2 CRC-16 set-up

Polynomial = $x^{16} + x^{15} + x^2 + 1$
Seed Value = 0x0000
Bit order reverse for data input: YES
1's complement for data input: NO
Bit order reverse for CRC sum: YES
1's complement for CRC sum: NO
CRC_MODE = 0x0000 0015
CRC_SEED = 0x0000 0000

27.7.3 CRC-32 set-up

Polynomial = $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
Seed Value = 0xFFFF FFFF
Bit order reverse for data input: YES
1's complement for data input: NO
Bit order reverse for CRC sum: YES
1's complement for CRC sum: YES
CRC_MODE = 0x0000 0036
CRC_SEED = 0xFFFF FFFF

28.1 How to read this chapter

The Mailbox is available on all LPC5410x parts.

28.2 Features

- Provides a means Inter-Processor Communication, allowing multiple CPUs to share resources and communicate with each other in a simple manner.
- Each CPU can cause up to 32 user defined interrupts to its partner.
- Each CPU can claim a shared resource if it is available.

28.3 Basic configuration

Set the MAILBOX bit in the AHBCLKCTRL0 register ([Table 51](#)) to enable the clock to the Mailbox.

28.4 Pin description

The Mailbox has no configurable pins.

28.5 General description

The Mailbox provides a means for simple communication between CPUs.

28.6 Register description

Table 436. Register overview: Mailbox (base address 0x1C02 C000)

| Name | Access | Address offset | Description | Reset value | Reference |
|---------|--------|----------------|--|-------------|---------------------------|
| IRQ0 | R/W | 0x000 | Interrupt request register for the Cortex-M0+ CPU. | 0 | Table 437 |
| IRQ0SET | WO | 0x004 | Set bits in IRQ0 | - | Table 438 |
| IRQ0CLR | WO | 0x008 | Clear bits in IRQ0 | - | Table 439 |
| IRQ1 | R/W | 0x010 | Interrupt request register for the Cortex M4 CPU. | 0 | Table 440 |
| IRQ1SET | WO | 0x014 | Set bits in IRQ1 | - | Table 441 |
| IRQ1CLR | WO | 0x018 | Clear bits in IRQ1 | - | Table 442 |
| MUTEX | R/W | 0x0F8 | Mutual exclusion register ^[1] | 0x1 | Table 443 |

[1] Reading and writing have specific side effects see detailed register description.

28.6.1 M0+ interrupt register

The IRQ0 register allows other CPUs to send interrupt requests to the Cortex-M0+ CPU. This is intended to allow communication between CPUs. For example, one CPU could be handling certain peripherals and signalling another CPU when data is available. Each bit can represent a different situation. The use of this feature is entirely up to the user.

Table 437. M0+ interrupt register (IRQ0, address 0x1C02 C000) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 31:0 | INTREQ | If any bit is set, an interrupt request is sent to the Cortex-M0+ interrupt controller. | 0 |

28.6.2 Cortex M0+ interrupt set register

The IRQ0SET register is used to set bits in the IRQ0 register.

Table 438. M0+ interrupt set register (IRQ0SET, address 0x1C02 C004) bit description

| Bit | Symbol | Description | Reset Value |
|------|------------|--|-------------|
| 31:0 | INTREQ SET | Writing 1 sets the corresponding bit in the IRQ0 register. | - |

28.6.3 M0+ interrupt clear register

The IRQ0CLR register is used to clear bits in the IRQ0 register.

Table 439. M0+ interrupt clear register (IRQ0CLR, address 0x1C02 C008) bit description

| Bit | Symbol | Description | Reset Value |
|------|------------|--|-------------|
| 31:0 | INTREQ CLR | Writing 1 clears the corresponding bit in the IRQ0 register. | - |

28.6.4 M4 interrupt register

The IRQ1 register allows other CPUs to send interrupt requests to the Cortex-M4 CPU. This is intended to allow communication between CPUs. For example, one CPU could be handling certain peripherals and signalling another CPU when data is available. Each bit can represent a different situation. The use of this feature is entirely up to the user.

Table 440. M4 interrupt (IRQ1, address 0x1C02 C010) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 31:0 | INTREQ | If any bit is set, an interrupt request is sent to the Cortex-M0+ interrupt controller. | 0 |

28.6.5 M4 interrupt set register

The IRQ0SET register is used to set bits in the IRQ0 register.

Table 441. M4 interrupt set register (IRQ1SET, address 0x1C02 C014) bit description

| Bit | Symbol | Description | Reset Value |
|------|---------------|--|-------------|
| 31:0 | INTREQ SET | Writing 1 sets the corresponding bit in the IRQ1 register. | - |

28.6.6 M4 interrupt clear register

The IRQ0SET register is used to clear bits in the IRQ0 register.

Table 442. M4 interrupt clear register (IRQ1CLR, address 0x1C02 C018) bit description

| Bit | Symbol | Description | Reset Value |
|------|---------------|--|-------------|
| 31:0 | INTREQ CLR | Writing 1 clears the corresponding bit in the IRQ1 register. | - |

28.6.7 Mutual Exclusion register

This register provides an Inter-Processor Communication handshake. When read for any reason, the current value will be returned and the bit will be cleared. The bit will be set again following any write.

This can be used as a resource allocation handshake between 2 CPUs. Whenever a CPU wishes to access a shared resource (possibly a resource allocation table in memory), it reads the MUTEX register. If it sees a 1, it has control over the shared resource allocation. When it has made any needed changes, it writes to the register, causing it to become set again, and making control of shared resource allocation available to another CPU. If a CPU reads a 0, it must wait for the bit to read as a 1 before accessing the shared resource allocation information.

Table 443. Mutual Exclusion register (MUTEX, address 0x1C02 C0F8) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 0 | EX | Cleared when read, set when written. See usage description above. | 1 |
| 31:1 | - | Reserved | - |

29.1 Functional description

The flash controller can generate a flash signature value on demand using an IAP function call. See [Section 32.6](#) for details.

29.1.1 Flash signature generation

The flash module contains a built-in signature generator. This generator can produce a 32-bit signature from a range of flash memory. A typical usage is to verify the flashed contents against a calculated signature (e.g. during programming).

The address range for generating a signature must be aligned on flash-word boundaries, i.e. 32-bit boundaries. Once started, signature generation completes independently. While signature generation is in progress, the flash memory cannot be accessed for other purposes, and an attempted read will cause a wait state to be asserted until signature generation is complete. Code outside of the flash (e.g. internal RAM) can be executed during signature generation. This can include interrupt services, if the interrupt vector table is re-mapped to memory other than the flash memory. The code that initiates signature generation should also be placed outside of the flash memory.

29.1.1.1 Signature generation

A signature can be generated for any part of the flash contents. The address range to be used for signature generation is defined by writing the start address to the FMSSTART register, and the stop address to the FMSSTOP register.

The signature generation is started by writing a 1 to the SIG_START bit in the FMSSTOP register. Starting the signature generation is typically combined with defining the stop address, which is done in the STOP bits of the same register.

The time that the signature generation takes is proportional to the address range for which the signature is generated. Reading of the flash memory for signature generation uses a self-timed read mechanism and does not depend on any configurable timing settings for the flash. A safe estimation for the duration of the signature generation is:

$$\text{Duration} = \text{int}((60 / \text{tcy}) + 3) \times (\text{FMSSTOP} - \text{FMSSTART} + 1)$$

When signature generation is triggered via software, the duration is in AHB clock cycles, and tcy is the time in ns for one AHB clock. The SIG_DONE bit in FMSTAT can be polled by software to determine when signature generation is complete.

After signature generation, a 32-bit signature can be read from the FMSW0 register. The 32-bit signature reflects the corrected data read from the flash and the flash parity bits and check bit values.

29.1.1.2 Content verification

The signature as it is read from the FMSW0 register must be equal to the reference signature. The following pseudo-code shows the algorithm to derive the reference signature:

```
sign = 0
FOR address = FMSSTART.START to FMSSTOP.STOPA
{
  FOR i = 0 TO 30
  {
    nextSign[i] = f_Q[address][i] XOR sign[i + 1]
  }
  nextSign[31] = f_Q[address][31] XOR sign[0] XOR sign[10] XOR sign[30] XOR sign[31]
  sign = nextSign
}
signature32 = sign
```

30.1 How to read this chapter

Debug functionality is available on all parts. Details depend on whether the Cortex-M0+ is present on the device.

30.2 Features

- Supports ARM Serial Wire Debug mode for the Cortex-M4 and, if present, the Cortex-M0+.
- Trace port provides Cortex-M4 CPU instruction trace capability. Output via a Serial Wire Viewer.
- Direct debug access to all memories, registers, and peripherals.
- No target resources are required for the debugging session.
- Breakpoints: the Cortex-M4 includes 8 instruction breakpoints that can also be used to remap instruction addresses for code patches. Two data comparators that can be used to remap addresses for patches to literal values. The Cortex-M0+, if present, includes 4 breakpoints.
- Watchpoints: the Cortex-M4 includes 4 data watchpoints that can also be used as triggers. The Cortex-M0+, if present, includes 2 watchpoints.
- Supports JTAG boundary scan.
- Instrumentation Trace Macrocell allows additional software controlled trace for the Cortex-M4.

30.3 Basic configuration

The serial wire debug pins are enabled by default. The JTAG pins for boundary scan are selected by hardware after a reset.

30.4 Pin description

The tables below indicate the various pin functions related to debug. Some of these functions share pins with other functions which therefore may not be used at the same time. Trace using the Serial Wire Output has limited bandwidth.

Table 444. Serial Wire Debug pin description

| Function | Direction | Connect to | Description |
|----------|----------------|-------------------|---|
| SWCLK | Input | PIO0_16 | Serial Wire Clock. This pin is the clock for SWD debug logic when in the Serial Wire Debug mode (SWD). This pin is pulled up internally. |
| SWDIO | Input / Output | PIO0_17 | Serial wire debug data input/output. The SWDIO pin is used by an external debug tool to communicate with and control the part. This pin is pulled up internally. |
| SWO | Output | PIO0_15 or PIO1_1 | Serial Wire Output. The SWO pin optionally provides data from the ITM for an external debug tool to evaluate. |

The JTAG boundary pin functions are selected by hardware at reset. See [Section 30.6.3](#).

The following setup is required to enable SWO output on GPIO PIO0-15 (FUNC2) or PIO1_1 (FUNC2):

1. Write 0x00000001 to TRACECLKDIV (0x400000E4). Enables the Trace divider.
2. If the clock to the IOCON block is not already enabled, write 0x00002000 to AHBCLKCTRLSET[0] (0x400000C8). The clock must be enabled in order to access any IOCON registers.

Table 445. JTAG boundary scan pin description

| Function | Direction | Description |
|--------------------------|-----------|--|
| TCK | Input | JTAG Test Clock. This pin is the clock for JTAG boundary scan when the $\overline{\text{RESET}}$ pin is LOW. |
| TMS | Input | JTAG Test Mode Select. The TMS pin selects the next state in the TAP state machine. This pin includes an internal pull-up and is used for JTAG boundary scan when the $\overline{\text{RESET}}$ pin is LOW. |
| TDI | Input | JTAG Test Data In. This is the serial data input for the shift register. This pin includes an internal pull-up and is used for JTAG boundary scan when the $\overline{\text{RESET}}$ pin is LOW. |
| TDO | Output | JTAG Test Data Output. This is the serial data output from the shift register. Data is shifted out of the device on the negative edge of the TCK signal. This pin is used for JTAG boundary scan when the $\overline{\text{RESET}}$ pin is LOW. |
| $\overline{\text{TRST}}$ | Input | JTAG Test Reset. The $\overline{\text{TRST}}$ pin can be used to reset the test logic within the debug logic. This pin includes an internal pull-up and is used for JTAG boundary scan when the $\overline{\text{RESET}}$ pin is LOW. |

30.5 General description

Serial wire debug functions are integrated into each CPU, with up to four breakpoints and two watchpoints. Boundary scan is also available.

Trace on the Cortex-M4 is supported via the Serial Wire Output.

30.6 Functional description

30.6.1 Debug limitations

Important: Due to limitations of the CPU, the part cannot wake up in the usual manner from Deep-sleep mode during debugging.

The debug mode changes the way in which reduced power modes work internal to the CPU. Therefore power measurements should not be made while debugging, power consumption is higher than during normal operation.

During a debugging session, the System Tick Timer is automatically stopped whenever the CPU is stopped. Other peripherals are not affected.

30.6.2 Debug connections for SWD

For debugging purposes, it is useful to provide access to the ISP entry pins (see [Chapter 6 “LPC5410x Boot process”](#)). The ISP entry pins can be used to recover the part from configurations which would disable the SWD port such as improper PLL configuration, assigning another function to the SWD pins via IOCON, entry into Deep power-down mode out of reset, etc. The ISP entry pins can be used for other functions such as GPIO but should not be held LOW on power-up or reset.

Internal and external SWD connections are shown in [Figure 59](#) and [Figure 60](#).

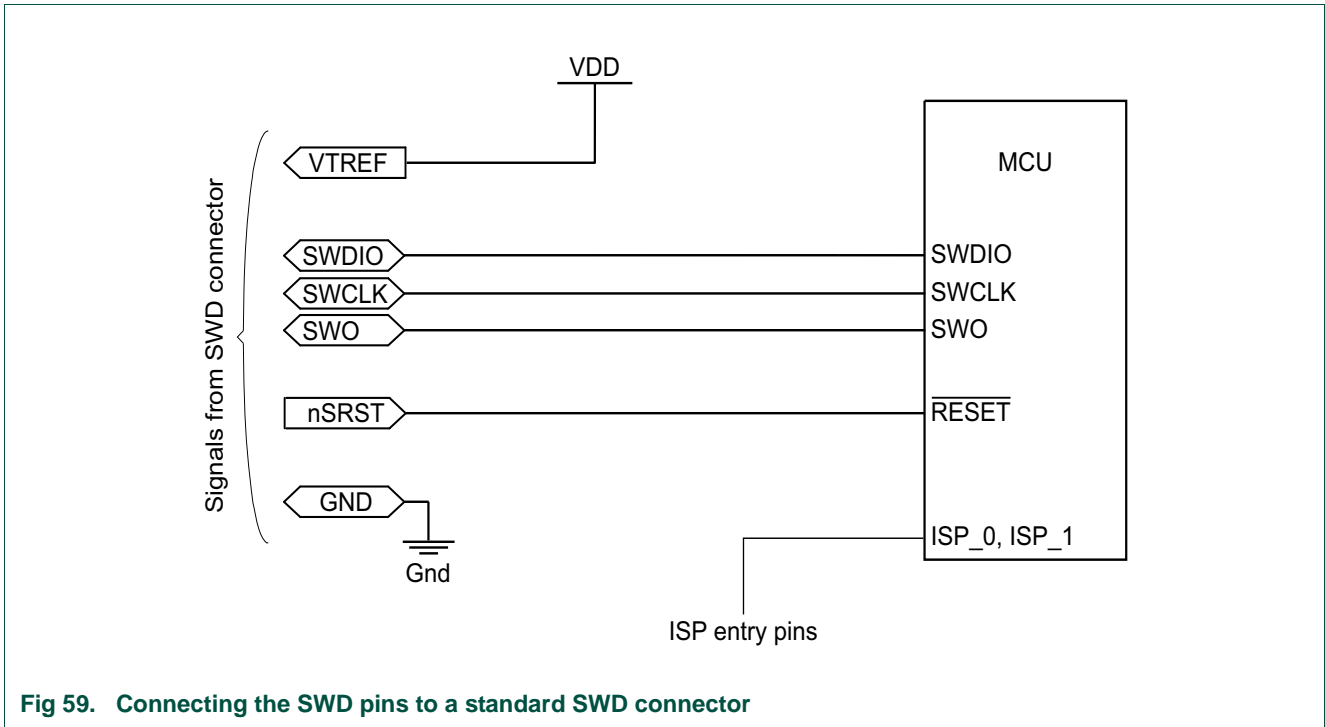


Fig 59. Connecting the SWD pins to a standard SWD connector

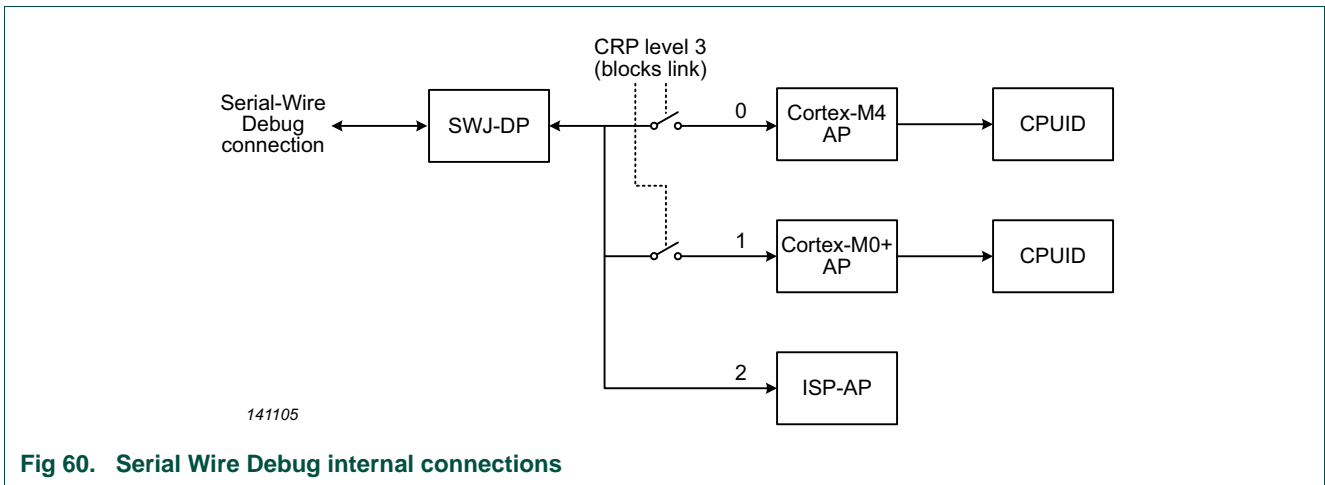


Fig 60. Serial Wire Debug internal connections

30.6.3 Boundary scan

The $\overline{\text{RESET}}$ pin selects between the test TAP controller for JTAG boundary scan ($\overline{\text{RESET}} = \text{LOW}$) and the ARM SWD debug port TAP controller ($\overline{\text{RESET}} = \text{HIGH}$). The ARM SWD debug port is disabled while the part is in reset. A LOW on the TRST pin resets the test TAP controller.

Remark: Boundary scan operations should not be started until 250 μs after POR. The test TAP must be reset after the boundary scan and left in either TLR or RTO state. Boundary scan is not affected by Code Read Protection.

Remark: POR, BOD reset, or a LOW on the TRST pin puts the test TAP controller in the Test-Logic Reset state. The first TCK clock while $\overline{\text{RESET}} = \text{HIGH}$ places the test TAP in Run-Test Idle mode.

30.6.4 In-System Programming Access Port (ISP-AP)

The ISP-AP is essentially a register-based communication port that may be accessed by both the CPU and the device debug port.

This port is used to implement certain commands that can operate even when the device has been programmed to the highest Code Read Protection level (CRP level 3). The ISP-AP is active whenever the device is attached to a debugger.

30.6.4.1 Resynchronization request

Communication with the ISP-AP is initiated by the debugger. The debugger first sets the RESYNCH_REQ bit in the CSW register. The debugger must then reset the device by either writing a 1 to the CHIP_RESET_REQ bit in the CSW, or by driving the actual reset pin of the device if it is able to do so.

30.6.4.2 Acknowledgement of resynchronization request

After requesting a resynchronization and resetting the device, the debugger reads the CSW register. This stalls the debugger if the device has not yet completed the resynchronization process. The debugger can repeat this process until it is able to read the CSW and find a 0 there.

30.6.4.3 Return phase

Following the initial resynchronization, communication by the debugger to the device is in the form of 32-bit writes to the REQUEST register. The debugger can read the result in the RETURN register. The debugger polls the RETURN register in the same manner as it polled the CSW following a resynchronization request.

30.6.4.4 Error handling

If an overrun occurs from either side of the communication, the appropriate error flag is set in the CSW. Once such an error occurs, the debugger will need to start with a new resynchronization request in order to clear the error flag.

30.6.4.5 Register description

The registers in the ISP-AP are shown below. These registers are readable by the CPU and are intended primarily to allow on-chip ROM routines to implement requests from an external debugger.

Table 446. Register overview: ISP-AP (base address 0x1C04 0000)

| Name | Access | Address offset | Description | Reset value | Reference |
|---------|--------|----------------|---|-------------|---------------------------|
| CSW | R/W | 0x00 | Command and status word. | 0 | Table 447 |
| REQUEST | R/W | 0x04 | Request from the debugger to the device. | 0 | Table 448 |
| RETURN | R/W | 0x08 | Return value from the device to the debugger. | 0 | Table 449 |
| ID | RO | 0xFC | Identification register. | 0x002A 0000 | Table 450 |

30.6.4.5.1 Command and Status Word register (CSW, offset 0x00) bit description

The CSW register contains command and status bits to facilitate communication between the debugger and the device.

Table 447. Command and Status Word register (CSW, offset 0x00) bit description

| Bit | Symbol | Description | Reset value |
|------|----------------|--|-------------|
| 0 | RESYNCH_REQ | The debugger sets this bit to requests a re-synchronization. | 0 |
| 1 | REQ_PENDING | A request is pending for the debugger: a value is waiting to be read from the REQUEST register. | 0 |
| 2 | DBG_OR_ERR | When 1, a debug overrun has occurred: a REQUEST value has been overwritten by the debugger before it was read by the device. | 0 |
| 3 | AHB_OR_ERR | When 1, an AHB overrun has occurred: a RETURN value has been overwritten by the device before it was read by the debugger. | 0 |
| 4 | SOFT_RESET | This bit is write-only by the device and resets the ISP-AP. | 0 |
| 5 | CHIP_RESET_REQ | This write -only bit causes the device (but not the ISP-AP) to be reset. | 0 |
| 31:6 | - | Reserved | - |

30.6.4.5.2 Request value register (REQUEST, offset 0x04) bit description

The REQUEST register is used by a debugger to send action requests to the device.

Table 448. Request value register (REQUEST, offset 0x04) bit description

| Bit | Symbol | Description | Reset value |
|------|---------|--|-------------|
| 31:0 | REQUEST | Request value. Reads as 0 when no new request is present. Cleared by the device. Can be read back by the debugger in order to confirm communication. | 0 |

30.6.4.5.3 Return value register (RETURN, offset 0x08) bit description

The RETURN register provides any response from the device to the debugger.

Table 449. Return value register (RETURN, offset 0x08) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|---|-------------|
| 31:0 | RETURN | Return value. This is any response from the device to the debugger. If no new data is present, a debugger read will be stalled until new data is available. | 0 |

30.6.4.5.4 Identification register (ID, offset 0xFC) bit description

The ID register provides an identification of the ISP-AP interface.

Table 450. Identification register (ID, offset 0xFC) bit description

| Bit | Symbol | Description | Reset value |
|------|--------|-----------------------|-------------|
| 31:0 | ID | Identification value. | 0x002A 0000 |

30.6.4.6 ISP-AP commands

Commands for the ISP-AP are listed below.

Table 451. Register overview: ISP-AP (base address 0x1C04 0000)

| Name | Command code | Description |
|-----------------|--------------|---|
| Enter ISP-AP | 1 | Cause the device to enter ISP-AP command mode. This must be done prior to sending other commands. |
| Bulk Erase | 2 | Erase the entire on-chip flash memory. |
| Query CRP Level | 3 | Queries the Code Read Protection (CRP) level currently in force on the device. |
| Exit ISP-AP | 4 | Cause the device to exit ISP-AP command mode. The Device returns to normal mode. |

30.6.4.7 ISP-AP return codes

Return codes for ISP-AP commands are listed below.

Table 452. Register overview: ISP-AP (base address 0x1C04 0000)

| Return code | Description |
|-------------|--|
| 0x0000 0000 | Command succeeded. Applies to commands other than “Query CRP level”. |
| 0x0010 0001 | Debug mode not entered. This is returned if other commands are sent prior to the “Enter ISP-AP” command. |
| 0x0010 0002 | Command not recognized. A command was received other than the ones defined above. |
| 0x0010 0003 | Command failed. |

30.7 Debug configuration

30.7.1 Cortex-M4

- Eight breakpoints. Six instruction breakpoints that can also be used to remap instruction addresses for code patches. Two data comparators that can be used to remap addresses for patches to literal values.
- Four data Watchpoints.
- Instrumentation Trace Macrocell allows additional software controlled trace capability.

30.7.2 Cortex-M0+ (present on LPC54102 devices)

- Four breakpoints.
- Two data Watchpoints.

31.1 How to read this chapter

The power profiles are available for all LPC5410x parts.

31.2 Features

- Simple APIs to control power consumption and wake-up in all power modes.
- Manage power consumption for sleep and active modes
- Prepare the part to enter low power modes (sleep, deep-sleep, power-down, and deep power-down).
- Configure wake-up from Deep-sleep and power-down via functions enabled by bits in the PDRUNCFG register.

31.3 General description

PLL setup and control of device power consumption or entry to low power modes can be configured through simple calls to the power profile. APIs exist to:

- Set up the System PLL.
- Set up on-chip power based on the expected operating frequency.
- Set up reduced power modes.
- Set up special low-frequency low power operation.

Remark: Disable all interrupts before making calls to the power profile API. The interrupts can be re-enabled after the power profile API calls have completed.

The API calls to the ROM are performed by executing functions which are pointed to by a pointer within the ROM Driver Table. [Figure 61](#) shows the pointer structure used to call the Power Profiles API. Other APIs must be included in user application code.

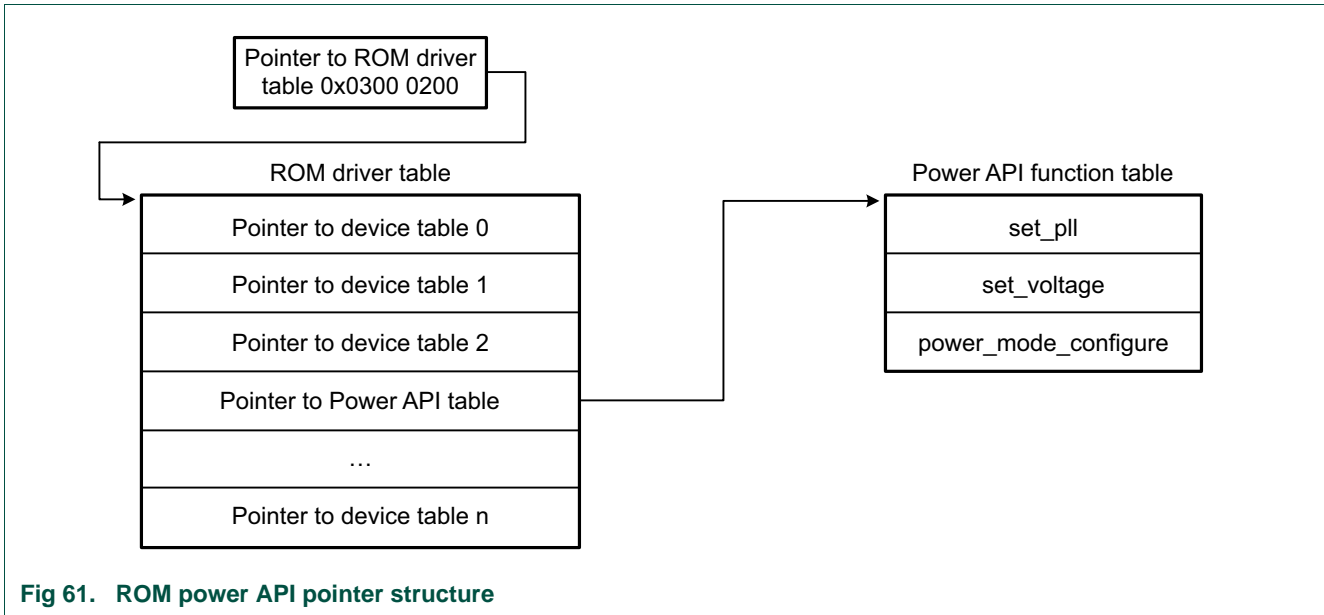


Fig 61. ROM power API pointer structure

31.4 API description

Power APIs provide functions to configure the system clock and set up the system for expected performance requirements.

Table 453. Power API ROM calls

| Function prototype | API description | Reference |
|--|---|--------------------------------|
| <pre>unsigned int set_pll (unsigned int multiplier, unsigned int input_freq)</pre> | Power API PLL configuration routine. This API sets up basic PLL operation. | Section 31.4.1 |
| <pre>unsigned int set_voltage (unsigned int mode, unsigned int desired_freq)</pre> | Power API internal voltage configuration routine. This API configures the internal regulator for the desired active operating mode and frequency. | Section 31.4.2 |
| <pre>void power_mode_configure (unsigned int mode, unsigned int peripheral);</pre> | Power API power mode configuration routine. This API prepares the chip for sleep, deep-sleep, power-down, or deep power-down mode and selects which peripherals can wake up the part from deep-sleep or power-down modes. | Section 31.4.3 |

The following elements have to be defined in an application that uses the ROM power profiles:

```

typedef struct _PWRD {
    unsigned int (*set_pll)(unsigned int multiplier, unsigned int input_freq);
    unsigned int (*set_voltage)(unsigned int mode, unsigned int desired_freq);
    void (*power_mode_configure)(unsigned int mode, unsigned int peripheral);
} PWRD;

#define rom_driver_ptr (*(ROM) **) 0x0300 0200
pPWRD = (PWRD *) (rom_driver_ptr->pPWRD);
    
```

31.4.1 set_pll

This routine sets up the System PLL given the PLL input frequency and feedback multiplier. Note that this API does not support the pre-divider and post-divider of the PLL, or support special PLL operating modes. A downloadable library is available that supports these features.

Table 454. set_pll routine

| Routine | set_pll |
|-----------------|---|
| Prototype | <code>void set_pll(multiplier, input_freq);</code> |
| Input parameter | Param0: multiplier (1 to 16) Param1: input_freq |
| Result | Error code. 0 = no error. |
| Description | Sets up the PLL for the requested multiplier and input frequency. This API does not support setting up the PLL pre-divider or post-divider. |

31.4.1.1 Param0: multiplier

The input parameter multiplier (Param0) specifies the feedback multiplier for the PLL. The range supported is from 1 to 16.

31.4.1.2 Param1: input_freq

The input frequency is the clock rate of the PLL input. The input frequency times the multiplier must not be greater than 100 MHz or less than 75 MHz.

31.4.1.3 Error or return codes

A return code of zero indicates that the operation was successful.

Table 455. Error codes

| Return code | Error code | Description |
|-------------|-----------------------|--|
| 0x000B 0002 | ERR_CLK_INVALID_PARAM | Multiplier = 0 or (multiplier * input_freq) > 100MHz |

31.4.2 set_voltage

This routine configures the device's internal power control settings according to the calling arguments. The goal is to prepare on-chip regulators to deliver the amount of power needed for the requested performance level, as defined by the CPU operating frequency.

Remark: The set_voltage API should only be used when the system clock divider is 1 (AHBCLKDIV = 1, see [Table 58](#)).

Table 456. set_voltage routine

| Routine | set_voltage |
|-----------------|---|
| Prototype | <code>void set_voltage(mode, desired_freq);</code> |
| Input parameter | Param0: mode (0 = low power mode) Param1: desired frequency (in MHz) |
| Result | Error code. 0 = no error. |
| Description | Configures the internal device voltage in active mode. |

31.4.2.1 Param0: mode

The input parameter mode (Param0) specifies one of two available power settings. If an illegal selection is provided, set_power returns PWR_INVALID_MODE and does not change the power control system.

Low power Mode

Optimize system power for the requested frequency.

31.4.2.2 Param1: frequency

The frequency is the clock rate the CPU will be using during the selected mode. The microcontroller uses to source the system and peripheral clocks. This operand must be an integer between 1 to 100 MHz inclusive.

31.4.2.3 Error or return codes

A return code of zero indicates that the operation was successful.

Table 457. Error codes

| Return code | Error code | Description |
|-------------|-----------------------|--|
| 0x000C 0004 | PWR_ERROR_INVALID_CFG | Mode is invalid |
| 0x000B 0002 | ERR_CLK_INVALID_PARAM | Frequency is outside the supported range |

31.4.3 power_mode_configure

The power_mode_configure API prepares the part, then enters any of the low power modes. Specifically for the deep-sleep and power-down modes, the API function configures which analog components remain running in those two modes, so that an interrupt from one of the analog peripherals can wake up the part.

Table 458. power_mode_configure routine

| Routine | power_mode_configure |
|-----------------|--|
| Prototype | <code>void power_mode_configure(mode, peripheral);</code> |
| Input parameter | Param0: mode Param1: peripheral |
| Result | None |
| Description | Defines the low power mode (either sleep, deep-sleep, power-down, or deep power-down modes). |

Remark: Aside from the analog peripherals listed with this parameter, the serial peripherals can also wake up the chip from deep-sleep or power-down modes on an interrupt triggered by an incoming signal. This wake-up scenario is not configured using the power_mode_configure API. For details, see [Section 22.3.2 “Configure the USART for wake-up”](#), [Section 23.3.1 “Configure the SPI for wake-up”](#), or [Section 24.4.3 “Configure the I²C for wake-up”](#).

31.4.3.1 Param0: mode

The mode parameter defines the low power mode and prepares the chip to enter the selected mode.

The following modes are valid:


```

#define SLEEP                0
#define DEEP_SLEEP           1
#define POWER_DOWN           2
#define DEEP_POWER_DOWN      3

```

31.4.3.2 Param1: peripheral

If sleep mode is selected with the mode parameter, the peripheral parameter is ignored.

The peripheral parameter defines which analog peripherals can wake up the chip from deep-sleep, power-down, or deep power-down mode. The selected peripherals remain running in deep-sleep or power-down. For example, the watchdog oscillator must be running if the WWDT is to remain active in deep-sleep or power-down mode or the RTC must be running if it is to remain active in deep power-down mode.

The peripheral parameter is a 32-bit value that corresponds to the PDRUNCFG register (see [Table 72](#)).

31.5 Functional description

31.5.1 Example low power mode control

31.5.1.1 Enter sleep mode

```

/* get Power config API table pointer */
pPWRD = (PWRD *) (rom_drivers_ptr->pPWRD);
/* parameter peripheral is don't care*/
pPWRD->power_mode_configure( SLEEP, 0x0 );
/* going to sleep mode. */

```

31.5.1.2 Enter power-down mode and set up WWDT and BOD for wake-up

```

/* configure wwdt and bod event to wake up the chip from power-down*/
LPC_SYSCON->STARTER0 = 0x3;
/* get Power config API table pointer */
pPWRD = (PWRD *) (rom_drivers_ptr->pPWRD);
/* WDT_OSC and BOD are turned on */
pPWRD->power_mode_configure( POWER_DOWN, PDRUNCFG_PD_WDT_OSC |
                             PDRUNCFG_PD_BOD_RESET |
                             PDRUNCFG_PD_BOD_INTR);
/* going to power-down mode. */

```

32.1 How to read this chapter

See [Table 459](#) for different flash configurations.

Table 459. LPC5410x flash configurations

| Type number | Flash | ISP via USART |
|-------------------------------|--------|---------------|
| LPC54101J256 and LPC54102J256 | 256 KB | yes |
| LPC54101J512 and LPC54102J512 | 512 KB | yes |

Remark: In addition to the ISP and IAP commands, a register can be accessed in the SYSCON block to configure flash memory access times, see [Section 4.5.33](#).

32.2 Features

- In-System Programming: In-System programming (ISP) is programming or reprogramming the on-chip flash memory, using the boot loader software and the USART serial port. This can be done when the part resides in the end-user board.
- In Application Programming: In-Application (IAP) programming is performing erase and write operation on the on-chip flash memory, as directed by the end-user application code.
- Small size (256 Byte) page erase programming.
- Flash access times can be configured through a register in the flash controller block.

32.3 General description

32.3.1 Boot loader

For the boot loader operation and boot pin, see [Chapter 6 “LPC5410x Boot process”](#).

The boot loader version can be read by ISP/IAP calls (see [Section 32.5.13](#) or [Section 32.6.6](#)).

32.3.2 Memory map after any reset

The boot ROM is 64 KB in size and is located in the memory region starting from the address 0x0300 0000. The boot loader is designed to run from this memory area, but both the ISP and IAP software use parts of the on-chip RAM. The RAM usage is described later in [Section 32.3.7](#).

32.3.3 Flash content protection mechanism

The LPC5410x is equipped with the Error Correction Code (ECC) capable Flash memory. The purpose of an error correction module is twofold. Firstly, it decodes data words read from the memory into output data words. Secondly, it encodes data words to be written to the memory. The error correction capability consists of single bit error correction with Hamming code.

The operation of ECC is transparent to the running application. The ECC content itself is stored in a flash memory not accessible by user's code to either read from it or write into it on its own. A byte of ECC corresponds to every consecutive 128 bits of the user accessible Flash. Consequently, Flash bytes from 0x0000 0000 to 0x0000 000F are protected by the first ECC byte, Flash bytes from 0x0000 0010 to 0x0000 001F are protected by the second ECC byte, etc.

Whenever the CPU requests a read from user's Flash, both 128 bits of raw data containing the specified memory location and the matching ECC byte are evaluated. If the ECC mechanism detects a single error in the fetched data, a correction will be applied before data are provided to the CPU. When a write request into the user's Flash is made, write of user specified content is accompanied by a matching ECC value calculated and stored in the ECC memory.

When a sector of Flash memory is erased, the corresponding ECC bytes are also erased. Once an ECC byte is written, it can not be updated unless it is erased first. Therefore, for the implemented ECC mechanism to perform properly, data must be written into the flash memory in groups of 16 bytes (or multiples of 16), aligned as described above.

32.3.4 Criteria for Valid User Code

The reserved CPU exception vector location 7 (offset 0x0000 001C in the vector table) should contain the 2's complement of the check-sum of table entries 0 through 6. This causes the checksum of the first 8 table entries to be 0. The boot loader code checksums the first 8 locations in sector 0 of the flash. If the result is 0, then execution control is transferred to the user code.

If the signature is not valid, the auto-baud routine synchronizes with the host via the serial port (USART).

If the USART is selected, the host should send a '?' (0x3F) as a synchronization character and wait for a response. The host side serial port settings should be 8 data bits, 1 stop bit and no parity. The auto-baud routine measures the bit time of the received synchronization character in terms of its own frequency and programs the baud rate generator of the serial port. It also sends an ASCII string ("Synchronized<CR><LF>") to the host. In response to this, the host should send back the same string ("Synchronized<CR><LF>").

The auto-baud routine looks at the received characters to verify synchronization. If synchronization is verified then "OK<CR><LF>" string is sent to the host. The host should respond by sending the crystal frequency (in kHz) at which the part is running. The response is required for backward compatibility of the boot loader code

and is ignored. "OK<CR><LF>" string is sent to the host after receiving the crystal frequency. If synchronization is not verified then the auto-baud routine waits again for a synchronization character. For auto-baud to work correctly in case of user invoked ISP, the clock frequency should be greater than or equal to 10 MHz. In USART ISP mode, the part is clocked by the IRC and the crystal frequency is ignored.

Once the crystal frequency is received the part is initialized and the ISP command handler is invoked. For safety reasons an "Unlock" command is required before executing the commands resulting in flash erase/write operations and the "Go" command. The rest of the commands can be executed without the unlock command. The Unlock command is required to be executed once per ISP session. The Unlock command is explained in

[Section 32.5 “USART ISP commands” on page 424.](#)

32.3.5 Flash partitions

Some IAP and ISP commands operate on sectors and specify sector numbers. In addition, a page erase command is available. The size of a sector is 32 KB and the size of a page is 256 Byte. One sector contains 128 pages. Sector 0 and page 0 are located at address 0x0000 0000.

Table 460. Flash sectors and pages

| Sector number | Sector size [KB] | Page numbers | Address range |
|---------------|------------------|--------------|---------------------------|
| 0 | 32 | 0 - 127 | 0x0000 0000 - 0x0000 7FFF |
| 1 | 32 | 128 - 255 | 0x0000 8000 - 0x0000 FFFF |
| 2 | 32 | 256 - 383 | 0x0001 0000 - 0x0001 7FFF |
| 3 | 32 | 384 - 511 | 0x0001 8000 - 0x0001 FFFF |
| 4 | 32 | 512 - 639 | 0x0002 0000 - 0x0002 7FFF |
| 5 | 32 | 640 - 767 | 0x0002 8000 - 0x0002 FFFF |
| 6 | 32 | 768 - 895 | 0x0003 0000 - 0x0003 7FFF |
| 7 | 32 | 896 - 1023 | 0x0003 8000 - 0x0003 FFFF |
| 8 | 32 | 1024 - 1151 | 0x0004 0000 - 0x0004 7FFF |
| 9 | 32 | 1152 - 1279 | 0x0004 8000 - 0x0004 FFFF |
| 10 | 32 | 1280 - 1407 | 0x0005 0000 - 0x0005 7FFF |
| 11 | 32 | 1408 - 1535 | 0x0005 8000 - 0x0005 FFFF |
| 12 | 32 | 1536 - 1663 | 0x0006 0000 - 0x0006 7FFF |
| 13 | 32 | 1664 - 1791 | 0x0006 8000 - 0x0006 FFFF |
| 14 | 32 | 1792 - 1919 | 0x0007 0000 - 0x0007 7FFF |
| 15 | 32 | 1920 - 2047 | 0x0007 8000 - 0x0007 FFFF |

32.3.6 Code Read Protection (CRP)

Code Read Protection is a mechanism that allows the user to enable different levels of security in the system so that access to the on-chip flash and use of the ISP can be restricted. When needed, CRP is invoked by programming a specific pattern in flash location at 0x0000 02FC. IAP commands are not affected by the code read protection.

Important: any CRP change becomes effective only after the device has gone through a power cycle.

Table 461. Code Read Protection (CRP) options

| Name | Pattern programmed in 0x0000 02FC | Description |
|--------|-----------------------------------|--|
| NO_ISP | 0x4E69 7370 | Prevents sampling of the pin for entering ISP mode. ISP sampling pin is available for other applications. |
| CRP1 | 0x1234 5678 | <p>Access to chip via the SWD pins is disabled. This mode allows partial flash update using the following ISP commands and restrictions:</p> <ul style="list-style-type: none"> • Write to RAM command cannot access RAM below 0x0200 0300. • Copy RAM to flash command can not write to Sector 0. • Erase command can erase Sector 0 only when all sectors are selected for erase. • Compare command is disabled. • Read Memory command is disabled. <p>This mode is useful when CRP is required and flash field updates are needed but all sectors can not be erased. Since compare command is disabled in case of partial updates the secondary loader should implement checksum mechanism to verify the integrity of the flash.</p> |
| CRP2 | 0x8765 4321 | <p>Access to chip via the SWD pins is disabled. The following ISP commands are disabled:</p> <ul style="list-style-type: none"> • Read Memory • Write to RAM • Go • Copy RAM to flash • Compare <p>When CRP2 is enabled the ISP erase command only allows erasure of all user sectors.</p> |
| CRP3 | 0x4321 8765 | <p>Access to chip via the SWD pins is disabled. ISP entry selected via the ISP entry pin is disabled if a valid user code is present in flash sector 0.</p> <p>This mode effectively disables ISP override using the entry pin. It is up to the user's application to provide a flash update mechanism using IAP calls or call reinvoke ISP command to enable flash update via USART0.</p> <p>Caution: If CRP3 is selected, no future factory testing can be performed on the device.</p> |

Table 462. ISP commands allowed for different CRP levels

| ISP command | CRP1 | CRP2 | CRP3 (no entry in ISP mode allowed) |
|-------------------------------------|---|-----------------------|-------------------------------------|
| Unlock | yes | yes | n/a |
| Set Baud Rate | yes | yes | n/a |
| Echo | yes | yes | n/a |
| Write to RAM | yes; above 0x0200 0300 only | no | n/a |
| Read Memory | no | no | n/a |
| Prepare sectors for write operation | yes | yes | n/a |
| Copy RAM to flash | yes; not to sector 0 | no | n/a |
| Go | no | no | n/a |
| Erase sector(s) | yes; sector 0 can only be erased when all sectors are erased. | yes; all sectors only | n/a |
| Erase page(s) | yes; page 0 can only be erased when all pages are erased (not recommended, use Erase Sector). | yes; all pages only | n/a |
| Blank check sectors | no | no | n/a |

Table 462. ISP commands allowed for different CRP levels

| ISP command | CRP1 | CRP2 | CRP3 (no entry in ISP mode allowed) |
|------------------------|------|------|-------------------------------------|
| Read Part ID | yes | yes | n/a |
| Read Boot code version | yes | yes | n/a |
| Compare | no | no | n/a |
| ReadUID | yes | yes | n/a |

In case a CRP mode is enabled and access to the chip is allowed via the ISP, an unsupported or restricted ISP command will be terminated with return code `CODE_READ_PROTECTION_ENABLED`.

32.3.6.1 ISP entry protection

In addition to the three CRP modes, the user can prevent the sampling of the pin for entering ISP mode and thereby release the pin for other applications. This is called the `NO_ISP` mode. The `NO_ISP` mode can be entered by programming the pattern `0x4E69 7370` at location `0x0000 02FC`.

The `NO_ISP` mode is identical to the CRP3 mode except for SWD access, which is allowed in `NO_ISP` mode but disabled in CRP3 mode. The `NO_ISP` mode does not offer any code protection.

32.3.7 ISP interrupt and SRAM use

32.3.7.1 Interrupts during IAP

The on-chip flash memory is not accessible during erase/write operations. When the user application code starts executing, the interrupt vectors from the user flash area are active. Before making any IAP call, either disable the interrupts or ensure that the user interrupt vectors are active in RAM and that the interrupt handlers reside in RAM. The IAP code does not use or disable interrupts.

32.3.7.2 RAM used by ISP command handler

Memory for the USART ISP commands is allocated dynamically.

32.3.7.3 RAM used by IAP command handler

Flash programming commands use the top 32 bytes of on-chip SRAM0 (see [Section 2.1.1](#) for details of the SRAM configuration). The maximum stack usage in the user allocated stack space is 128 bytes and grows downwards.

32.4 USART communication protocol

All USART ISP commands should be sent as single ASCII strings. Strings should be terminated with Carriage Return (CR) and/or Line Feed (LF) control characters. Extra <CR> and <LF> characters are ignored. All ISP responses are sent as <CR><LF> terminated ASCII strings. Data is sent and received in plain binary format.

32.4.1 USART ISP command format

"Command Parameter_0 Parameter_1 ... Parameter_n<CR><LF>" "Data" (Data only for Write commands).

32.4.2 USART ISP response format

"Return_Code<CR><LF>Response_0<CR><LF>Response_1<CR><LF> ... Response_n<CR><LF>" "Data" (Data only for Read commands).

32.4.3 USART ISP data format

The data stream is in plain binary format.

32.5 USART ISP commands

The following commands are accepted by the ISP command handler. Detailed status codes are supported for each command. The command handler sends the return code `INVALID_COMMAND` when an undefined command is received. Commands and return codes are in ASCII format.

`CMD_SUCCESS` is sent by ISP command handler only when received ISP command has been completely executed and the new ISP command can be given by the host. Exceptions from this rule are "Set Baud Rate", "Write to RAM", "Read Memory", and "Go" commands.

Table 463. USART ISP command summary

| ISP Command | Usage | Described in |
|-------------------------------------|---|---------------------------|
| Unlock | U <Unlock Code> | Table 464 |
| Set Baud Rate | B <Baud Rate> <stop bit> | Table 465 |
| Echo | A <setting> | Table 466 |
| Write to RAM | W <start address> <number of bytes> | Table 467 |
| Read Memory | R <address> <number of bytes> | Table 468 |
| Prepare sectors for write operation | P <start sector number> <end sector number> | Table 469 |
| Copy RAM to flash | C <Flash address> <RAM address> <number of bytes> | Table 470 |
| Go | G <address> <Mode> | Table 471 |
| Erase sector(s) | E <start sector number> <end sector number> | Table 472 |
| Erase page(s) | X <start page number> <end page number> | Table 473 |
| Blank check sector(s) | I <start sector number> <end sector number> | Table 474 |
| Read Part ID | J | Table 475 |
| Read Boot code version | K | Table 477 |
| Compare | M <address1> <address2> <number of bytes> | Table 478 |
| ReadUID | N | Table 479 |
| Read CRC checksum | S <address> <number of bytes> | Table 480 |
| Read flash signature | Z | Table 481 |

32.5.1 Unlock

Table 464. USART ISP Unlock command

| Command | U |
|-------------|---|
| Input | Unlock code: 23130 ₁₀ |
| Return Code | <code>CMD_SUCCESS</code> <code>INVALID_CODE</code> <code>PARAM_ERROR</code> |
| Description | This command is used to unlock Flash Write, Erase, and Go commands. |
| Example | "U 23130<CR><LF>" unlocks the Flash Write/Erase & Go commands. |

32.5.2 Set Baud Rate

Table 465. USART ISP Set Baud Rate command

| Command | B |
|-------------|---|
| Input | Baud Rate: 9600 19200 38400 57600 115200 Stop bit: 1 2 |
| Return Code | CMD_SUCCESS INVALID_BAUD_RATE INVALID_STOP_BIT PARAM_ERROR |
| Description | This command is used to change the baud rate. The new baud rate is effective after the command handler sends the CMD_SUCCESS return code. |
| Example | "B 57600 1<CR><LF>" sets the serial port to baud rate 57600 bps and 1 stop bit. |

32.5.3 Echo

Table 466. USART ISP Echo command

| Command | A |
|-------------|--|
| Input | Setting: ON = 1 OFF = 0 |
| Return Code | CMD_SUCCESS PARAM_ERROR |
| Description | The default setting for echo command is ON. When ON the ISP command handler sends the received serial data back to the host. |
| Example | "A 0<CR><LF>" turns echo off. |

32.5.4 Write to RAM

The host should send the plain binary code after receiving the CMD_SUCCESS return code. This ISP command handler responds with "OK<CR><LF>" when the transfer has finished.

Table 467. USART ISP Write to RAM command

| Command | W |
|-------------|---|
| Input | Start Address: RAM address where data bytes are to be written. This address should be a word boundary. Number of Bytes: Number of bytes to be written. Count should be a multiple of 4 |
| Return Code | CMD_SUCCESS ADDR_ERROR (Address not on word boundary) ADDR_NOT_MAPPED COUNT_ERROR (Byte count is not multiple of 4) PARAM_ERROR CODE_READ_PROTECTION_ENABLED |
| Description | This command is used to download data to RAM. This command is blocked when code read protection levels 2 or 3 are enabled. Writing |
| Example | "W 33555200 4<CR><LF>" writes 4 bytes of data to address 0x0200 0300. |

32.5.5 Read Memory

Reads the plain binary code of the data stream, followed by the CMD_SUCCESS return code.

Table 468. USART ISP Read Memory command

| Command | R |
|-------------|--|
| Input | Start Address: Address from where data bytes are to be read. This address should be a word boundary. Number of Bytes: Number of bytes to be read. Count should be a multiple of 4. |
| Return Code | CMD_SUCCESS followed by <actual data (plain binary)> ADDR_ERROR (Address not on word boundary) ADDR_NOT_MAPPED COUNT_ERROR (Byte count is not a multiple of 4) PARAM_ERROR CODE_READ_PROTECTION_ENABLED |
| Description | This command is used to read data from RAM or flash memory. This command is blocked when code read protection is enabled. |
| Example | "R 33554432 4<CR><LF>" reads 4 bytes of data from address 0x0200 0000. |

32.5.6 Prepare sectors for write operation

This command makes flash write/erase operation a two step process.

Table 469. USART ISP Prepare sectors for write operation command

| Command | P |
|-------------|--|
| Input | Start Sector Number End Sector Number: Should be greater than or equal to start sector number. |
| Return Code | CMD_SUCCESS BUSY INVALID_SECTOR PARAM_ERROR |
| Description | This command must be executed before executing "Copy RAM to flash" or "Erase Sector(s)" command. Successful execution of the "Copy RAM to flash" or "Erase Sector(s)" command causes relevant sectors to be protected again. To prepare a single sector use the same "Start" and "End" sector numbers. |
| Example | "P 0 0<CR><LF>" prepares the flash sector 0. |

32.5.7 Copy RAM to flash

When writing to the flash, the following limitations apply:

1. The smallest amount of data that can be written to flash by the copy RAM to flash command is 256 byte (equal to one page).
2. One page consists of 16 flash words (lines), and the smallest amount that can be modified per flash write is one flash word (one line). This limitation exists because ECC is applied during the flash write operation, see [Section 32.3.3](#).
3. To avoid write disturbance (a mechanism intrinsic to flash memories), an erase should be performed after 16 consecutive writes inside the same page. Note that the erase operation then erases the entire sector.

Remark: Once a page has been written to 16 times, it is still possible to write to other pages within the same sector without performing a sector erase (assuming that those pages have been erased previously).

Table 470. USART ISP Copy command

| Command | C |
|-------------|--|
| Input | <p>Flash Address(DST): Destination flash address where data bytes are to be written. The destination address should be a 256 byte boundary.</p> <p>RAM Address(SRC): Source RAM address from where data bytes are to be read.</p> <p>Number of Bytes: Number of bytes to be written. Should be 256 512 1024 4096.</p> |
| Return Code | <p>CMD_SUCCESS </p> <p>SRC_ADDR_ERROR (Address not on word boundary) </p> <p>DST_ADDR_ERROR (Address not on correct boundary) </p> <p>SRC_ADDR_NOT_MAPPED </p> <p>DST_ADDR_NOT_MAPPED </p> <p>COUNT_ERROR (Byte count is not 256 512 1024 4096) </p> <p>SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION </p> <p>BUSY </p> <p>CMD_LOCKED </p> <p>PARAM_ERROR </p> <p>CODE_READ_PROTECTION_ENABLED</p> |
| Description | <p>This command is used to program the flash memory. The "Prepare Sector(s) for Write Operation" command should precede this command. The affected sectors are automatically protected again once the copy command is successfully executed. This command is blocked when code read protection is enabled. Also see Section 32.3.3 for the number of bytes that can be written.</p> |
| Example | <p>"C 0 33556480 512<CR><LF>" copies 512 bytes from the RAM address 0x0200 0800 to the flash address 0.</p> |

32.5.8 Go

Table 471. USART ISP Go command

| Command | G |
|-------------|---|
| Input | <p>Address: Flash or RAM address from which the code execution is to be started. This address should be on a word boundary.</p> <p>Mode: T (Execute program in Thumb Mode) A (Execute program in ARM mode).</p> |
| Return Code | <p>CMD_SUCCESS </p> <p>ADDR_ERROR </p> <p>ADDR_NOT_MAPPED </p> <p>CMD_LOCKED </p> <p>PARAM_ERROR </p> <p>CODE_READ_PROTECTION_ENABLED</p> |
| Description | <p>This command is used to execute a program residing in RAM or flash memory. It may not be possible to return to the ISP command handler once this command is successfully executed. This command is blocked when code read protection is enabled.</p> |
| Example | <p>"G 0 A<CR><LF>" branches to address 0x0000 0000 in ARM mode.</p> |

32.5.9 Erase sectors

Table 472. USART ISP Erase sector command

| Command | E |
|-------------|---|
| Input | Start Sector Number End Sector Number: Should be greater than or equal to start sector number. |
| Return Code | CMD_SUCCESS BUSY INVALID_SECTOR SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION CMD_LOCKED PARAM_ERROR CODE_READ_PROTECTION_ENABLED |
| Description | This command is used to erase one or more sector(s) of on-chip flash memory. This command only allows erasure of all user sectors when the code read protection is enabled. |
| Example | "E 2 3<CR><LF>" erases the flash sectors 2 and 3. |

32.5.10 Erase pages

Table 473. USART ISP Erase page command

| Command | X |
|-------------|---|
| Input | Start Page Number End Page Number: Should be greater than or equal to start page number. |
| Return Code | CMD_SUCCESS BUSY INVALID_PAGE SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION CMD_LOCKED PARAM_ERROR CODE_READ_PROTECTION_ENABLED |
| Description | This command is used to erase one or more page(s) of on-chip flash memory. |
| Example | "X 2 3<CR><LF>" erases the flash pages 2 and 3. |

32.5.11 Blank check sectors

Table 474. USART ISP Blank check sector command

| Command | I |
|-------------|---|
| Input | Start Sector Number: End Sector Number: Should be greater than or equal to start sector number. |
| Return Code | CMD_SUCCESS SECTOR_NOT_BLANK (followed by <Offset of the first non blank word location> <Contents of non blank word location>) INVALID_SECTOR PARAM_ERROR |
| Description | This command is used to blank check one or more sectors of on-chip flash memory. When CRP is enabled, the blank check command returns 0 for the offset and value of sectors which are not blank. Blank sectors are correctly reported irrespective of the CRP setting. |
| Example | "I 2 3<CR><LF>" blank checks the flash sectors 2 and 3. |

32.5.12 Read Part Identification number

Table 475. USART ISP Read Part Identification command

| Command | J |
|-------------|--|
| Input | None. |
| Return Code | CMD_SUCCESS followed by part identification number (see Table 476 "LPCA5410x device identification numbers"). |
| Description | This command is used to read the part identification number. |

Table 476. LPCA5410x device identification numbers

| Device | Hex coding |
|--------------|-------------|
| LPC54101J256 | 0x8845 4101 |
| LPC54101J512 | 0x8885 4101 |
| LPC54102J256 | 0x8845 4102 |
| LPC54102J512 | 0x8885 4102 |

32.5.13 Read Boot code version number

Table 477. USART ISP Read Boot Code version number command

| Command | K |
|-------------|--|
| Input | None |
| Return Code | CMD_SUCCESS followed by 2 bytes of boot code version number in ASCII format. It is to be interpreted as <byte1(Major)>.<byte0(Minor)>. |
| Description | This command is used to read the boot code version number. |

32.5.14 Compare

Table 478. USART ISP Compare command

| Command | M |
|-------------|---|
| Input | <p>Address1 (DST): Starting flash or RAM address of data bytes to be compared. This address should be a word boundary.</p> <p>Address2 (SRC): Starting flash or RAM address of data bytes to be compared. This address should be a word boundary.</p> <p>Number of Bytes: Number of bytes to be compared; should be a multiple of 4.</p> |
| Return Code | <p>CMD_SUCCESS (Source and destination data are equal)</p> <p>COMPARE_ERROR (Followed by the offset of first mismatch)</p> <p>COUNT_ERROR (Byte count is not a multiple of 4) </p> <p>ADDR_ERROR </p> <p>ADDR_NOT_MAPPED </p> <p>PARAM_ERROR </p> |
| Description | <p>This command is used to compare the memory contents at two locations.</p> <p>Compare result may not be correct when source or destination address contains any of the first 512 bytes starting from address zero. First 512 bytes are re-mapped to boot ROM</p> |
| Example | <p>"M 8192 33587200 4<CR><LF>" compares 4 bytes from the RAM address 0x0200 8000 to the 4 bytes from the flash address 0x2000.</p> |

32.5.15 ReadUID

Table 479. USART ReadUID command

| Command | N |
|-------------|---|
| Input | None |
| Return Code | CMD_SUCCESS followed by four 32-bit words of a unique serial number in ASCII format. The word sent at the lowest address is sent first. |
| Description | This command is used to read the unique ID. |

32.5.16 Read CRC checksum

Get the CRC checksum of a block of RAM or flash. CMD_SUCCESS followed by 8 bytes of CRC checksum in decimal format.

The checksum is calculated as follows:

CRC-32 polynomial: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$

Seed Value: 0xFFFF FFFF

Table 480. USART ISP Read CRC checksum command

| Command | S |
|-------------|--|
| Input | <p>Address: The data are read from this address for CRC checksum calculation. This address must be on a word boundary.</p> <p>Number of Bytes: Number of bytes to be calculated for the CRC checksum; must be a multiple of 4.</p> |
| Return Code | CMD_SUCCESS followed by data in decimal format ADDR_ERROR (address not on word boundary) ADDR_NOT_MAPPED COUNT_ERROR (byte count is not a multiple of 4) PARAM_ERROR CODE_READ_PROTECTION_ENABLED |
| Description | This command is used to read the CRC checksum of a block of RAM or flash memory. This command is blocked when code read protection is enabled. |
| Example | <p>"S 33587200 4<CR><LF>" reads the CRC checksum for 4 bytes of data from address 0x0200 8000.</p> <p>If checksum value is 0xCB43926, then the host will receive:</p> <p>"3421780262 <CR><LF>"</p> |

32.5.17 Read flash signature

Get the signature for the entire flash memory using the flash signature generator described in [Chapter 29](#). CMD_SUCCESS followed by the 32-bit flash signature represented in decimal format.

Table 481. USART ISP Read flash signature command

| Command | Z |
|-------------|--|
| Input | none |
| Return Code | CMD_SUCCESS followed by data in decimal format CODE_READ_PROTECTION_ENABLED |
| Description | This command is used to read the signature of the entire flash memory. This command is blocked when code read protection is enabled. |
| Example | <p>"Z<CR><LF>" returns the signature for the entire flash memory.</p> <p>If signature value is 0x3BD7, then the host will receive:</p> <p>"15319 <CR><LF>"</p> |

32.5.18 ISP Error codes

Table 482. USART ISP Error codes

| Return Code | Error code | Description |
|-------------|-----------------------------|--|
| 0x0 | ERR_ISP_CMD_SUCCESS | Command is executed successfully. Sent by ISP handler only when command given by the host has been completely and successfully executed. |
| 0x1 | ERR_ISP_INVALID_COMMAND | Invalid command. |
| 0x2 | ERR_ISP_SRC_ADDR_ERROR | Source address is not on word boundary. |
| 0x3 | ERR_ISP_DST_ADDR_ERROR | Destination address is not on a correct boundary. |
| 0x4 | ERR_ISP_SRC_ADDR_NOT_MAPPED | Source address is not mapped in the memory map. Count value is taken in to consideration where applicable. |

Table 482. USART ISP Error codes

| Return Code | Error code | Description |
|-------------|---|---|
| 0x5 | ERR_ISP_DST_ADDR_NOT_MAPPED | Destination address is not mapped in the memory map. Count value is taken in to consideration where applicable. |
| 0x6 | ERR_ISP_COUNT_ERROR | Byte count is not multiple of 4 or is not a permitted value. |
| 0x7 | ERR_ISP_INVALID_SECTOR | Sector number is invalid or end sector number is greater than start sector number. |
| 0x8 | ERR_ISP_SECTOR_NOT_BLANK | Sector is not blank. |
| 0x9 | ERR_ISP_SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | Command to prepare sector for write operation was not executed. |
| 0xA | ERR_ISP_COMPARE_ERROR | Source and destination data not equal. |
| 0xB | ERR_ISP_BUSY | Flash programming hardware interface is busy. |
| 0xC | ERR_ISP_PARAM_ERROR | Insufficient number of parameters or invalid parameter. |
| 0xD | ERR_ISP_ADDR_ERROR | Address is not on word boundary. |
| 0xE | ERR_ISP_ADDR_NOT_MAPPED | Address is not mapped in the memory map. Count value is taken in to consideration where applicable. |
| 0xF | ERR_ISP_CMD_LOCKED | Command is locked. |
| 0x10 | ERR_ISP_INVALID_CODE | Unlock code is invalid. |
| 0x11 | ERR_ISP_INVALID_BAUD_RATE | Invalid baud rate setting. |
| 0x12 | ERR_ISP_INVALID_STOP_BIT | Invalid stop bit setting. |
| 0x13 | ERR_ISP_CODE_READ_PROTECTION_ENABLED | Code read protection enabled. |
| 0x14 | - | Reserved. |
| 0x15 | - | Reserved. |
| 0x16 | - | Reserved. |
| 0x17 | ERR_ISP_IRC_NO_POWER | IRC not turned on in the PDRUNCFG register. |
| 0x18 | ERR_ISP_FLASH_NO_POWER | Flash not turned on in the PDRUNCFG register. |
| 0x19 | - | Reserved. |
| 0x1A | - | Reserved. |
| 0x1B | ERR_ISP_FLASH_NO_CLOCK | Flash clock disabled in the AHBCLKCTRL register. |
| 0x1C | ERR_ISP_REINVOKE_ISP_CONFIG | Reinvoke ISP not successful. |

```

typedef enum
{
    ERR_ISP_BASE = 0x00000000,
    /*0x00000001*/ ERR_ISP_INVALID_COMMAND = ERR_ISP_BASE + 1,
    /*0x00000002*/ ERR_ISP_SRC_ADDR_ERROR,
    /*0x00000003*/ ERR_ISP_DST_ADDR_ERROR,
    /*0x00000004*/ ERR_ISP_SRC_ADDR_NOT_MAPPED,
    /*0x00000005*/ ERR_ISP_DST_ADDR_NOT_MAPPED,
    /*0x00000006*/ ERR_ISP_COUNT_ERROR,
    /*0x00000007*/ ERR_ISP_INVALID_SECTOR,
    /*0x00000008*/ ERR_ISP_SECTOR_NOT_BLANK,
    /*0x00000009*/ ERR_ISP_SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION,
    /*0x0000000A*/ ERR_ISP_COMPARE_ERROR,
    /*0x0000000B*/ ERR_ISP_BUSY, /* Flash programming hardware interface is busy */

```



```
/*0x0000000C*/ ERR_ISP_PARAM_ERROR, /* Insufficient number of parameters */
/*0x0000000D*/ ERR_ISP_ADDR_ERROR, /* Address not on word boundary */
/*0x0000000E*/ ERR_ISP_ADDR_NOT_MAPPED,
/*0x0000000F*/ ERR_ISP_CMD_LOCKED, /* Command is locked */
/*0x00000010*/ ERR_ISP_INVALID_CODE, /* Unlock code is invalid */
/*0x00000011*/ ERR_ISP_INVALID_BAUD_RATE,
/*0x00000012*/ ERR_ISP_INVALID_STOP_BIT,
/*0x00000013*/ ERR_ISP_CODE_READ_PROTECTION_ENABLED,
/*0x00000014*/ ERR_ISP_INVALID_FLASH_UNIT, /* reserved */
/*0x00000015*/ ERR_ISP_USER_CODE_CHECKSUM, /* reserved */
/*0x00000016*/ ERR_ISP_SETTING_ACTIVE_PARTITION, /* reserved */
/*0x00000017*/ ERR_ISP_IRC_NO_POWER,
/*0x00000018*/ ERR_ISP_FLASH_NO_POWER,
/*0x0000001B*/ ERR_ISP_FLASH_NO_CLOCK,
/*0x0000001C*/ ERR_ISP_REINVOKE_ISP_CONFIG
} ErrorCode_t;
```

32.6 IAP commands

For in application programming the IAP routine should be called with a word pointer in register r0 pointing to memory (RAM) containing command code and parameters. The result of the IAP command is returned in the result table pointed to by register r1. The user can reuse the command table for result by passing the same pointer in registers r0 and r1. The parameter table should be big enough to hold all the results in case the number of results are more than number of parameters. Parameter passing is illustrated in the [Figure 62](#).

The number of parameters and results vary according to the IAP command. The maximum number of parameters is 5, passed to the "Copy RAM to FLASH" command. The maximum number of results is 5, returned by the "ReadUID" command. The command handler sends the status code INVALID_COMMAND when an undefined command is received. The IAP routine resides at 0x0300 0204 location and it is thumb code, therefore called as 0x03000205 by the Cortex-M4 to insure Thumb operation.

The IAP function could be called in the following way using C:

Define the IAP location entry point. Since the least significant bit of the IAP location is set there will be a change to Thumb instruction set if called by the Cortex-M4.

```
#define IAP_LOCATION 0x03000205
```

Define data structure or pointers to pass IAP command table and result table to the IAP function:

```
unsigned int command_param[5];  
unsigned int status_result[5];
```

or

```
unsigned int * command_param;  
unsigned int * status_result;  
command_param = (unsigned int *) 0x...  
status_result =(unsigned int *) 0x...
```

Define pointer to function type, which takes two parameters and returns void. Note the IAP returns the result with the base address of the table residing in R1.

```
typedef void (*IAP)(unsigned int [],unsigned int[]);  
IAP iap_entry;
```

Setting the function pointer:

```
#define IAP_LOCATION 0x0300 0204  
  
iap_entry=(IAP) IAP_LOCATION;
```

To call the IAP use the following statement.

```
iap_entry (command_param,status_result);
```

Up to 4 parameters can be passed in the r0, r1, r2 and r3 registers respectively (see the *ARM Thumb Procedure Call Standard SWS ESPC 0002 A-05*). Additional parameters are passed on the stack. Up to 4 parameters can be returned in the r0, r1, r2 and r3 registers respectively. Additional parameters are returned indirectly via memory. Some of the IAP calls require more than 4 parameters. If the ARM suggested scheme is used for the parameter passing/returning then it might create problems due to difference in the C compiler implementation from different vendors. The suggested parameter passing scheme reduces such risk.

The flash memory is not accessible during a write or erase operation. IAP commands, which results in a flash write/erase operation, use 32 bytes of space in the top portion of the on-chip RAM for execution. The user program should not use this space if IAP flash programming is permitted in the application.

Table 483. IAP Command Summary

| IAP Command | Command code | Reference |
|---------------------------------------|--------------|---------------------------|
| Prepare sector(s) for write operation | 50 (decimal) | Table 484 |
| Copy RAM to flash | 51 (decimal) | Table 485 |
| Erase sector(s) | 52 (decimal) | Table 486 |
| Blank check sector(s) | 53 (decimal) | Table 487 |
| Read Part ID | 54 (decimal) | Table 488 |
| Read Boot code version | 55 (decimal) | Table 489 |
| Compare | 56 (decimal) | Table 490 |
| Reinvoke ISP | 57 (decimal) | Table 491 |
| Read UID | 58 (decimal) | Table 492 |
| Erase page(s) | 59 (decimal) | Table 493 |
| Read Signature | 70 (decimal) | Table 494 |

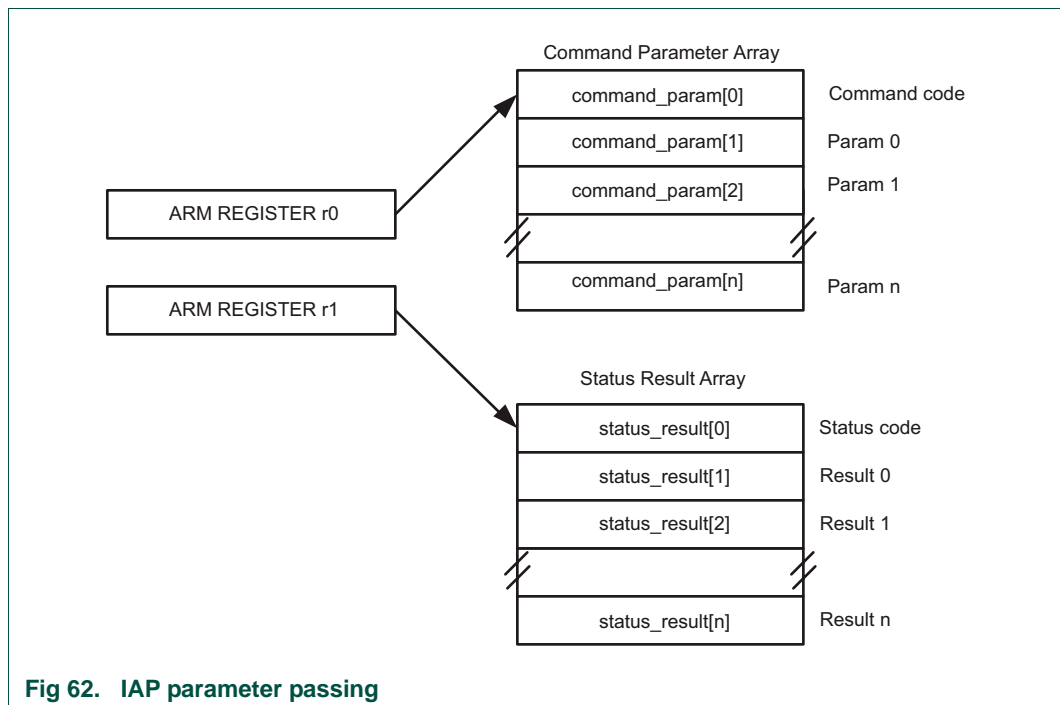


Fig 62. IAP parameter passing

32.6.1 Prepare sector(s) for write operation

This command makes flash write/erase operation a two step process.

Table 484. IAP Prepare sector(s) for write operation command

| Command | Prepare sector(s) for write operation |
|-------------|--|
| Input | <p>Command code: 50 (decimal)</p> <p>Param0: Start Sector Number</p> <p>Param1: End Sector Number (should be greater than or equal to start sector number).</p> |
| Status code | <p>CMD_SUCCESS </p> <p>BUSY </p> <p>INVALID_SECTOR</p> |
| Result | None |
| Description | <p>This command must be executed before executing "Copy RAM to flash" or "Erase Sector(s)" command. Successful execution of the "Copy RAM to flash" or "Erase Sector(s)" command causes relevant sectors to be protected again. The boot sector can not be prepared by this command. To prepare a single sector use the same "Start" and "End" sector numbers.</p> |

32.6.2 Copy RAM to flash

See [Section 32.5.7](#) for limitations on the write-to-flash process.

Table 485. IAP Copy RAM to flash command

| Command | Copy RAM to flash |
|-------------|---|
| Input | <p>Command code: 51 (decimal)</p> <p>Param0(DST): Destination flash address where data bytes are to be written. This address should be a 256 byte boundary.</p> <p>Param1(SRC): Source RAM address from which data bytes are to be read. This address should be a word boundary.</p> <p>Param2: Number of bytes to be written. Should be 256 512 1024 4096.</p> <p>Param3: System Clock Frequency (CCLK) in kHz.</p> |
| Status code | <p>CMD_SUCCESS </p> <p>SRC_ADDR_ERROR (Address not a word boundary) </p> <p>DST_ADDR_ERROR (Address not on correct boundary) </p> <p>SRC_ADDR_NOT_MAPPED </p> <p>DST_ADDR_NOT_MAPPED </p> <p>COUNT_ERROR (Byte count is not 256 512 1024 4096) </p> <p>SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION </p> <p>BUSY</p> |
| Result | None |
| Description | <p>This command is used to program the flash memory. The affected sectors should be prepared first by calling "Prepare Sector for Write Operation" command. The affected sectors are automatically protected again once the copy command is successfully executed. The boot sector can not be written by this command. Also see Section 32.3.3 for the number of bytes that can be written.</p> <p>Remark: All user code must be written in such a way that no master accesses the flash while this command is executed and the flash is programmed.</p> |

32.6.3 Erase Sector(s)

Table 486. IAP Erase Sector(s) command

| Command | Erase Sector(s) |
|-------------|--|
| Input | Command code: 52 (decimal) Param0: Start Sector Number Param1: End Sector Number (should be greater than or equal to start sector number). Param2: System Clock Frequency (CCLK) in kHz. |
| Status code | CMD_SUCCESS BUSY SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION INVALID_SECTOR |
| Result | None |
| Description | This command is used to erase a sector or multiple sectors of on-chip flash memory. The boot sector can not be erased by this command. To erase a single sector use the same "Start" and "End" sector numbers. Remark: All user code must be written in such a way that no master accesses the flash while this command is executed and the flash is erased. |

32.6.4 Blank check sector(s)

Table 487. IAP Blank check sector(s) command

| Command | Blank check sector(s) |
|-------------|---|
| Input | Command code: 53 (decimal) Param0: Start Sector Number Param1: End Sector Number (should be greater than or equal to start sector number). |
| Status code | CMD_SUCCESS BUSY SECTOR_NOT_BLANK INVALID_SECTOR |
| Result | Result0: Offset of the first non blank word location if the status code is SECTOR_NOT_BLANK. Result1: Contents of non blank word location. |
| Description | This command is used to blank check a sector or multiple sectors of on-chip flash memory. To blank check a single sector use the same "Start" and "End" sector numbers. |

32.6.5 Read Part Identification number

Table 488. IAP Read Part Identification command

| Command | Read part identification number |
|-------------|--|
| Input | Command code: 54 (decimal) Parameters: None |
| Status code | CMD_SUCCESS |
| Result | Result0: Part Identification Number. |
| Description | This command is used to read the part identification number. |

32.6.6 Read Boot code version number

Table 489. IAP Read Boot Code version number command

| Command | Read boot code version number |
|-------------|--|
| Input | Command code: 55 (decimal) Parameters: None |
| Status code | CMD_SUCCESS |
| Result | Result0: 2 bytes of boot code version number. Read as <byte1(Major)>.<byte0(Minor)> |
| Description | This command is used to read the boot code version number. |

32.6.7 Compare <address1> <address2> <no of bytes>

Table 490. IAP Compare command

| Command | Compare |
|-------------|--|
| Input | Command code: 56 (decimal) Param0(DST): Starting flash or RAM address of data bytes to be compared; should be a word boundary. Param1(SRC): Starting flash or RAM address of data bytes to be compared; should be a word boundary. Param2: Number of bytes to be compared; should be a multiple of 4. |
| Status code | CMD_SUCCESS COMPARE_ERROR COUNT_ERROR (Byte count is not a multiple of 4) ADDR_ERROR ADDR_NOT_MAPPED |
| Result | Result0: Offset of the first mismatch if the status code is COMPARE_ERROR. |
| Description | This command is used to compare the memory contents at two locations. The result may not be correct when the source or destination includes any of the first 512 bytes starting from address zero. The first 512 bytes can be re-mapped to RAM. |

32.6.8 Reinvoke ISP

Table 491. Reinvoke ISP

| Command | Compare |
|-------------|--|
| Input | Command code: 57 (decimal) |
| Status code | None |
| Result | None. |
| Description | This command is used to invoke the boot loader in ISP mode. It maps boot vectors and configures the peripherals for ISP. This command may be used when a valid user program is present in the internal flash memory and the ISP entry pin are not accessible to force the ISP mode. Before calling this command, enable the clocks to the default USART0 Rx/D and Tx/D pins. |

32.6.9 ReadUID

Table 492. IAP ReadUID command

| Command | Compare |
|-------------|--|
| Input | Command code: 58 (decimal) |
| Status code | CMD_SUCCESS |
| Result | Result0: The first 32-bit word (at the lowest address). Result1: The second 32-bit word. Result2: The third 32-bit word. Result3: The fourth 32-bit word. |
| Description | This command is used to read the unique ID. |

32.6.10 Erase page

Table 493. IAP Erase page command

| Command | Erase page |
|-------------|---|
| Input | Command code: 59 (decimal) Param0: Start page number. Param1: End page number (should be greater than or equal to start page) Param2: System Clock Frequency (CCLK) in kHz. |
| Status code | CMD_SUCCESS BUSY SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION INVALID_PAGE |
| Result | None |
| Description | This command is used to erase a page or multiple pages of on-chip flash memory. To erase a single page use the same "start" and "end" page numbers. Remark: All user code must be written in such a way that no master accesses the flash while this command is executed and the flash is erased. |

32.6.11 Read Signature

Table 494. IAP Read Signature command

| Command | Read Signature |
|-------------|--|
| Input | Command code: 70 (decimal) |
| Status code | CMD_SUCCESS |
| Result | Result0: The 32-bit generated signature. |
| Description | This command is used to obtain a 32-bit signature value of the entire flash memory. See Chapter 29 "LPC5410x flash controller" for details of the signature algorithm. |

32.6.12 IAP Status Codes

Table 495. IAP Status codes Summary

| Status code | Mnemonic | Description |
|-------------|---|---|
| 0 | CMD_SUCCESS | Command is executed successfully. |
| 1 | INVALID_COMMAND | Invalid command. |
| 2 | SRC_ADDR_ERROR | Source address is not on a word boundary. |
| 3 | DST_ADDR_ERROR | Destination address is not on a correct boundary. |
| 4 | SRC_ADDR_NOT_MAPPED | Source address is not mapped in the memory map. Count value is taken in to consideration where applicable. |
| 5 | DST_ADDR_NOT_MAPPED | Destination address is not mapped in the memory map. Count value is taken in to consideration where applicable. |
| 6 | COUNT_ERROR | Byte count is not multiple of 4 or is not a permitted value. |
| 7 | INVALID_SECTOR | Sector number is invalid. |
| 8 | SECTOR_NOT_BLANK | Sector is not blank. |
| 9 | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | Command to prepare sector for write operation was not executed. |
| 10 | COMPARE_ERROR | Source and destination data is not same. |
| 11 | BUSY | flash programming hardware interface is busy. |

33.1 ARM Cortex-M4 Details

ARM Limited publishes the document “Cortex™-M4 Devices Generic User Guide”, which is available on their website at:

- For the online manual, go to “infocenter.arm.com”, then search for “cortex-m4 user guide”. This will bring up links to chapters of the user guide.
- There are links at the bottom of user guide chapters to download a PDF file of the user guide.

This section of this manual describes the Cortex-M4 implementation options and other distinctions that apply for the LPC5410x devices.

33.1.1 Cortex-M4 implementation options

The Cortex™-M4 CPU provides a number of implementation options. These are given below for the LPC5410x.

- The MPU is included for the Cortex-M4. The MPU provides fine grain memory control, enabling applications to implement security privilege levels, separating code, data and stack on a task-by-task basis.
- The FPU is included for the Cortex-M4. The FPU supports single-precision floating-point computation functionality in compliance with the ANSI/IEEE Standard 754-2008. The FPU provides add, subtract, multiply, divide, multiply and accumulate, and square root operations. It also performs a variety of conversions between fixed-point, floating-point, and integer data formats.
- 46 interrupts are implemented for the Cortex-M4. Not all interrupts are available on all part numbers.
- 3 interrupt priority bits are implemented on the Cortex-M4.
- Sleep mode power-saving: NXP microcontrollers extend the number of reduced power modes beyond what is directly supported by the Cortex-M4. Details of reduced power modes and wake-up possibilities can be found in [Chapter 31](#).
- Reset of the Cortex-M4 resets the CPU register bank.
- Memory features: The memory map for LPC5410x devices is shown in [Section 2.1.2](#).
- Bit banding is included on the Cortex-M4. APB peripherals are located in bit-band space.

In addition, there are debug and trace options, see [Chapter 30](#).

33.2 ARM Cortex-M0+ Details (present on LPC54102 devices)

ARM Limited publishes the document “Cortex™-M0+ Devices Generic User Guide”, which is available on their website at:

- For the online manual, go to “infocenter.arm.com”, then search for “cortex-mo+ user guide”. This will bring up links to chapters of the user guide.

- There are links at the bottom of user guide chapters to download a PDF file of the user guide.

This section of this manual describes the Cortex-M0+ implementation options and any other distinctions that apply for the LPC5410x devices.

33.2.1 Cortex-M0+ implementation options

The Cortex™-M0+ provides a number of implementation options. These are given below for the LPC5410x.

- An MPU is not included for the Cortex-M0+.
- 32 interrupts are implemented for the Cortex-M0+. Not all interrupts are available on all part numbers.
- The vector table offset register is included.
- The multiplier configuration is the low power, 32-clock version.
- Sleep mode power-saving: NXP microcontrollers extend the number of reduced power modes beyond what is directly supported by the Cortex-M0+. Details of reduced power modes and wake-up possibilities on the LPC5410x can be found in [Chapter 31](#).
- Reset of the Cortex-M0+ resets the CPU register bank.
- Memory features: The memory map for LPC5410x devices is shown in [Section 2.1.2](#).
- SysTick timer: The SysTick timer is included for the Cortex-M0+, for details see [Section 20.5](#).

In addition, there are debug and trace options, see [Chapter 30](#).

34.1 Abbreviations

Table 496. Abbreviations

| Acronym | Description |
|----------------|---|
| ADC | Analog-to-Digital Converter |
| AHB | Advanced High-performance Bus |
| AMBA | Advanced Microcontroller Bus Architecture |
| APB | Advanced Peripheral Bus |
| API | Application Programming Interface |
| BOD | BrownOut Detection |
| BSDL | Boundary-Scan Description Language |
| CRC | Cyclic Redundancy Check |
| DCC | Debug Communication Channel |
| DMA | Direct Memory Access |
| FIFO | First-In-First-Out |
| FMC | Flash Memory Controller |
| GPIO | General Purpose Input/Output |
| I2C | Inter-IC Control bus |
| I2C or IIC | Inter-Integrated Circuit bus |
| IAP | In-Application Programming |
| IRC oscillator | Internal Resistor-Capacitor oscillator |
| IrDA | Infrared Data Association |
| ISP | In-System Programming |
| ISR | Interrupt Service Routine |
| JTAG | Joint Test Action Group |
| LIN | Local Interconnect Network |
| NVIC | Nested Vectored Interrupt Controller |
| PLL | Phase-Locked Loop |
| PLL | Phase-Locked Loop |
| POR | Power-On Reset |
| PWM | Pulse Width Modulator |
| RAM | Random Access Memory |
| SPI | Serial Peripheral Interface |
| SRAM | Static Random Access Memory |
| SWD | Serial-Wire Debug |
| TAP | Test Access Port |
| USART | Universal Synchronous/Asynchronous Receiver/Transmitter |

34.2 References

- [1] **Cortex-M4 TRM** — ARM Cortex-M4 Processor Technical Reference Manual
- [2] **Cortex-M0+ TRM** — ARM Cortex-M0+ Processor Technical Reference Manual
- [3] **AN11538** — [AN11538 application note and code bundle](#) (SCT cookbook)

34.3 Legal information

34.3.1 Definitions

Draft — The document is a draft version only. The content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included herein and shall have no liability for the consequences of use of such information.

34.3.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors accepts no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Evaluation products — This product is provided on an "as is" and "with all faults" basis for evaluation purposes only. NXP Semiconductors, its affiliates and their suppliers expressly disclaim all warranties, whether express, implied or statutory, including but not limited to the implied warranties of non-infringement, merchantability and fitness for a particular purpose. The entire risk as to the quality, or arising out of the use or performance, of this product remains with customer.

In no event shall NXP Semiconductors, its affiliates or their suppliers be liable to customer for any special, indirect, consequential, punitive or incidental damages (including without limitation damages for loss of business, business interruption, loss of use, loss of data or information, and the like) arising out of the use of or inability to use the product, whether or not based on tort (including negligence), strict liability, breach of contract, breach of warranty or any other theory, even if advised of the possibility of such damages.

Notwithstanding any damages that customer might incur for any reason whatsoever (including without limitation, all damages referenced above and all direct or general damages), the entire liability of NXP Semiconductors, its affiliates and their suppliers and customer's exclusive remedy for all of the foregoing shall be limited to actual damages incurred by customer based on reasonable reliance up to the greater of the amount actually paid by customer for the product or five dollars (US\$5.00). The foregoing limitations, exclusions and disclaimers shall apply to the maximum extent permitted by applicable law, even if any remedy fails of its essential purpose.

34.3.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

IP-C-bus — logo is a trademark of NXP B.V.

34.4 Tables

| | | | |
|---|----|---|----|
| Table 1. Main SRAM configuration | 8 | | |
| Table 2. Connection of interrupt sources to the NVIC | 12 | Table 39. Peripheral reset control clear register 0 (PRESETCTRLCLR0, address 0x4000 0054) bit description | 32 |
| Table 3. Register overview: NVIC (base address 0xE000 E000) | 14 | Table 40. Peripheral reset control clear register 1 (PRESETCTRLCLR1, address 0x4000 0058) bit description | 33 |
| Table 4. Interrupt Set-Enable Register 0 register | 15 | Table 41. POR captured PIO status register 0 (PIOPORCAP0, address 0x4000 005C) bit description | 33 |
| Table 5. Interrupt Set-Enable Register 1 register | 16 | Table 42. POR captured PIO status register 1 (PIOPORCAP1, address 0x4000 0060) bit description | 33 |
| Table 6. Interrupt Clear-Enable Register 0 | 16 | Table 43. Reset captured PIO status register 0 (PIORESCAP0, address 0x4000 0068) bit description | 33 |
| Table 7. Interrupt Clear-Enable Register 1 register | 16 | Table 44. Reset captured PIO status register 1 (PIORESCAP1, address 0x4000 006C) bit description | 33 |
| Table 8. Interrupt Set-Pending Register 0 register | 17 | Table 45. Main clock source select register A (MAINCLKSELA, address 0x4000 0080) bit description | 34 |
| Table 9. Interrupt Set-Pending Register 1 register | 17 | Table 46. Main clock source select register B (MAINCLKSELB, address 0x4000 0084) bit description | 34 |
| Table 10. Interrupt Clear-Pending Register 0 register | 17 | Table 47. ADC clock source select (ADCCLKSEL, address 0x4000 008C) bit description | 35 |
| Table 11. Interrupt Clear-Pending Register 1 register | 17 | Table 48. CLKOUT clock source select register (CLKOUTSELA, address 0x4000 0094) bit description | 35 |
| Table 12. Interrupt Active Bit Register 0 | 18 | Table 49. CLKOUT clock source select register (CLKOUTSELB, address 0x4000 0098) bit description | 36 |
| Table 13. Interrupt Active Bit Register 1 | 18 | Table 50. System PLL clock source select register (SYSPLLCLKSEL, address 0x4000 00A0) bit description | 36 |
| Table 14. Interrupt Priority Register 0 | 18 | Table 51. AHB Clock Control register 0 (AHBCLKCTRL0, address 0x4000 00C0) bit description | 37 |
| Table 15. Interrupt Priority Register 1 | 18 | Table 52. AHB Clock Control register 1 (AHBCLKCTRL1, address 0x4000 00C4) bit description | 38 |
| Table 16. Interrupt Priority Register 2 | 19 | Table 53. Clock control set register 0 (AHBCLKCTRLSET0, address 0x4000 00C8) bit description | 38 |
| Table 17. Interrupt Priority Register 3 | 19 | Table 54. Clock control set register 1 (AHBCLKCTRLSET1, address 0x4000 00CC) bit description | 38 |
| Table 18. Interrupt Priority Register 4 | 20 | Table 55. Clock control clear register 0 (AHBCLKCTRLCLR0, address 0x4000 00D0) bit description | 39 |
| Table 19. Interrupt Priority Register 5 | 20 | Table 56. Clock control clear register 1 (AHBCLKCTRLCLR1, address 0x4000 00D4) bit description | 39 |
| Table 20. Interrupt Priority Register 6 | 20 | Table 57. SYSTICK clock divider (SYSTICKCLKDIV, address 0x4000 00E0) bit description | 39 |
| Table 21. Interrupt Priority Register 7 | 21 | Table 58. System clock divider register (AHBCLKDIV, address 0x4000 0100) bit description | 39 |
| Table 22. Interrupt Priority Register 8 | 21 | Table 59. ADC clock source divider (ADCCLKDIV, address 0x4000 0108) bit description | 40 |
| Table 23. Interrupt Priority Register 9 | 21 | | |
| Table 24. Interrupt Priority Register 10 | 22 | | |
| Table 25. Software Trigger Interrupt Register (STIR) | 22 | | |
| Table 26. SYSCON pin description | 24 | | |
| Table 27. Register overview: Main system configuration (base address 0x4000 0000) | 26 | | |
| Table 28. Register overview: Asynchronous system configuration (base address 0x4008 0000) | 27 | | |
| Table 29. Register overview: Other system configuration (base address 0x4002 C000) | 28 | | |
| Table 30. AHB matrix priority register 0 (AHBMATPRIO, address 0x4000 0004) bit description | 28 | | |
| Table 31. System tick timer calibration register (SYSTCKCAL, address 0x4000 0014) bit description | 29 | | |
| Table 32. NMI source selection register (NMISRC, address 0x4000 001C) bit description | 29 | | |
| Table 33. Asynchronous APB Control register (ASYNCAPBCTRL, address 0x4000 0020) bit description | 29 | | |
| Table 34. System reset status register (SYSRSTSTAT, address 0x4000 0040) bit description | 30 | | |
| Table 35. Peripheral reset control register 0 (PRESETCTRL0, address 0x4000 0044) bit description | 30 | | |
| Table 36. Peripheral reset control register 1 (PRESETCTRL1, address 0x4000 0048) bit description | 31 | | |
| Table 37. Peripheral reset control set register 0 (PRESETCTRLSET0, address 0x4000 004C) bit description | 32 | | |
| Table 38. Peripheral reset control set register 1 (PRESETCTRLSET1, address 0x4000 0050) bit | | | |

| | | | |
|--|----|--|-----|
| Table 60. CLKOUT clock divider register (CLKOUTDIV, address 0x4000 010C) bit description | 40 | 0x4000 03F8) bit description | 58 |
| Table 61. Frequency measure function control register (FREQMCTRL, address 0x4000 0120) bit description | 40 | Table 87. Device ID0 register values | 58 |
| Table 62. Flash configuration register (FLASHCFG, main syscon: offset 0x400) bit description | 41 | Table 88. Device ID1 register (DEVICE_ID1, address 0x4000 03FC) bit description | 58 |
| Table 63. FIFO control register (FIFOCTRL, address 0x4000 0148) bit description | 42 | Table 89. Asynchronous peripheral reset control register (ASYNCPRESETCTRL, address 0x4008 0000) bit description | 59 |
| Table 64. IRC control register (IRCCTRL, address 0x4000 0184) bit description | 43 | Table 90. Asynchronous peripheral reset control set register (ASYNCPRESETCTRLSET, address 0x4008 0004) bit description | 59 |
| Table 65. RTC oscillator control register (RTCOSCCTRL, address 0x4000 0190) bit description | 43 | Table 91. Asynchronous peripheral reset control clear register (ASYNCPRESETCTRLCLR, address 0x4008 0008) bit description | 60 |
| Table 66. System PLL control register (SYSPLLCTRL, address 0x4000 01B0) bit description | 44 | Table 92. Asynchronous APB clock control register (ASYNCAPBCLKCTRL, address 0x4008 0010) bit description | 60 |
| Table 67. System PLL status register (SYSPLLSTAT, address 0x4000 01B4) bit description | 45 | Table 93. Asynchronous APB clock control set register (ASYNCAPBCLKCTRLSET, address 0x4008 0014) bit description | 60 |
| Table 68. System PLL N-divider register (SYSPLLNDEC, address 0x4000 01B8) bit description | 45 | Table 94. Asynchronous APB clock control clear register (ASYNCAPBCLKCTRLCLR, address 0x4008 0018) bit description | 61 |
| Table 69. System PLL P-divider register (SYSPLLPDEC, address 0x4000 01BC) bit description | 46 | Table 95. Asynchronous clock source select register A (ASYNCAPBCLKSELA, address 0x4008 0020) bit description | 61 |
| Table 70. System PLL spread spectrum control register 0 (SYSPLLSSCTRL0, address 0x4000 01C0) bit description | 47 | Table 96. Asynchronous clock source select register B (ASYNCAPBCLKSELB, address 0x4008 0024) bit description | 62 |
| Table 71. System PLL spread spectrum control register 1 (SYSPLLSSCTRL1, address 0x4000 01C4) bit description | 48 | Table 97. Asynchronous APB clock divider register (ASYNCCLKDIV, address 0x4008 0028) bit description | 62 |
| Table 72. Power Configuration register (PDRUNCFG, address 0x4000 0210) bit description | 50 | Table 98. USART fractional baud rate generator register (FRGCTRL, address 0x4008 0030) bit description | 62 |
| Table 73. Power configuration set register (PDRUNCFGSET, address 0x4000 0214) bit description | 51 | Table 99. BOD control register (BODCTRL, address 0x4002 C044) bit description | 63 |
| Table 74. Power configuration clear register (PDRUNCFGCLR, address 0x4000 0218) bit description | 51 | Table 100. PLL operating mode summary | 67 |
| Table 75. Start enable register 0 (STARTER0, address 0x4000 0240) bit description | 52 | Table 101. System PLL status register (SYSPLLSTAT, address 0x4000 01B4) bit description | 69 |
| Table 76. Start enable register 1 (STARTER1, address 0x4000 0244) bit description | 54 | Table 102. Peripheral configuration in reduced power modes | 74 |
| Table 77. Start enable set register 0 (STARTERSET0, address 0x4000 0248) bit description | 54 | Table 103. Wake-up sources for reduced power modes | 75 |
| Table 78. Start enable set register 1 (STARTERSET1, address 0x4000 024C) bit description | 54 | Table 104. ISP modes | 81 |
| Table 79. Start enable clear register 0 (STARTERCLR0, address 0x4000 0250) bit description | 55 | Table 105. Pin description | 83 |
| Table 80. Start enable clear register 1 (STARTERCLR1, address 0x4000 0254) bit description | 55 | Table 106. Available pins and configuration registers | 93 |
| Table 81. CPU Control register (CPUCTRL, address 0x4000 0300) bit description | 56 | Table 107. Register overview: I/O configuration (base address 0x4001 C000) | 97 |
| Table 82. Coprocessor Boot register (CPBOOT, address 0x4000 0304) bit description | 57 | Table 108. Address map PIO0_[0:22] registers | 98 |
| Table 83. Coprocessor Stack register (CPSTACK, address 0x4000 0308) bit description | 57 | Table 109. Type D IOCON registers (PIO0_[0:22], address offsets [0x000:0x058]) bit description | 98 |
| Table 84. Coprocessor Status register (CPSTAT, address 0x4000 030C) bit description | 57 | Table 110. Type D I/O Control registers: FUNC values and pin functions | 99 |
| Table 85. JTAG ID code register (JTAGIDCODE, address 0x4000 03F4) bit description | 58 | Table 111. Address map PIO0_[23:28] registers | 100 |
| Table 86. Device ID0 register (DEVICE_ID0, address | | Table 112. Type I IOCON registers (PIO0_[23:28], address offsets [0x05C:0x070]) bit description | 100 |
| | | Table 113. Suggested IOCON settings for I2C functions | 100 |
| | | Table 114. Type I I/O Control registers: FUNC values and pin functions | 101 |

| | | | |
|---|-----|--|-----|
| Table 115. Address map PIO0_[29:31] registers | 101 | Table 147. Address map SET[0:1] registers | 117 |
| Table 116. Type A IOCON registers(PIO0_[29:31], address offsets [0x074:0x07C]) bit description. | 101 | Table 148. GPIO set port register (SET[0:1], address offset [0x2200:0x2204]) bit description | 117 |
| Table 117. Address map PIO1_[0:8] registers | 102 | Table 149. Address map CLR[0:1] registers | 117 |
| Table 118. Type A IOCON registers(PIO1_[0:8], address offsets [0x080:0x0A0]) bit description. | 102 | Table 150. GPIO clear port register (CLR[0:1], address offset [0x2280:0x2284]) bit description | 118 |
| Table 119. Type A I/O Control registers: FUNC values and pin functions | 103 | Table 151. Address map NOT[0:1] registers. | 118 |
| Table 120. Address map PIO1_[9:17] registers. | 104 | Table 152. GPIO toggle port register (NOT[0:1], address offset [0x2300:0x2304]) bit description | 118 |
| Table 121. Type D IOCON registers (PIO1_[9:17], address offsets [0x0A4:0x0C4]) bit description | 104 | Table 153. Register overview: Pin interrupts/pattern match engine (base address: 0x4001 8000) | 127 |
| Table 122. Type D I/O Control registers: FUNC values and pin functions | 105 | Table 154. Pin interrupt mode register (ISEL, address 0x4001 8000) bit description | 128 |
| Table 123. INPUT MUX pin description. | 106 | Table 155. Pin interrupt level or rising edge interrupt enable register (IENR, address 0x4001 8004) bit description | 128 |
| Table 124. Register overview: Input multiplexing (base address 0x4005 0000) | 108 | Table 156. Pin interrupt level or rising edge interrupt set register (SIENR, address 0x4001 8008) bit description | 128 |
| Table 125. Address map PINTSEL[0:7] registers | 109 | Table 157. Pin interrupt level or rising edge interrupt clear register (CIENR, address 0x4001 800C) bit description | 129 |
| Table 126. Pin interrupt select registers (PINTSEL[0:7], address offsets [0x0C0:0x0DC]) bit description | 109 | Table 158. Pin interrupt active level or falling edge interrupt enable register (IENF, address 0x4001 8010) bit description | 129 |
| Table 127. Address map DMA_ITRIG_INMUX[0:21] registers | 110 | Table 159. Pin interrupt active level or falling edge interrupt set register (SIENF, address 0x4001 8014) bit description | 130 |
| Table 128. DMA trigger Input mux registers (DMA_ITRIG_INMUX[0:21], address offsets [0x0E0:0x134]) bit description | 110 | Table 160. Pin interrupt active level or falling edge interrupt clear register (CIENF, address 0x4001 8018) bit description | 130 |
| Table 129. Address map DMA_OTRIG_INMUX[0:3] registers | 110 | Table 161. Pin interrupt rising edge register (RISE, address 0x4001 801C) bit description | 130 |
| Table 130. DMA output trigger feedback mux registers (DMA_OTRIG_INMUX[0:3], address offset [0x140:0x14C]) bit description | 111 | Table 162. Pin interrupt falling edge register (FALL, address 0x4001 8020) bit description | 131 |
| Table 131. Frequency measure function frequency clock select register (FREQMEAS_REF, address 0x4005 0160) bit description | 112 | Table 163. Pin interrupt status register (IST, address 0x4001 8024) bit description | 131 |
| Table 132. Frequency measure function target clock select register (FREQMEAS_TARGET, address 0x4005 0164) bit description | 112 | Table 164. Pattern match interrupt control register (PMCTRL, address 0x4001 8028) bit description. | 132 |
| Table 133. GPIO pins available | 113 | Table 165. Pattern match bit-slice source register (PMSRC, address 0x4001 802C) bit description | 132 |
| Table 134. Register overview: GPIO port (base address 0x1C00 0000) | 114 | Table 166. Pattern match bit slice configuration register (PMCFG, address 0x4001 8030) bit description | 135 |
| Table 135. Address map B[0:49] registers. | 115 | Table 167. Pin interrupt registers for edge- and level-sensitive pins | 140 |
| Table 136. GPIO port byte pin registers (B[0:49], address offset [0x0000:0x0031]) bit description. | 115 | Table 168. Register overview: GROUP0 interrupt (base address 0x4001 0000 (GINT0) and 0x4001 4000 (GINT1)) | 145 |
| Table 137. Address map W[0:49] registers | 115 | Table 169. GPIO grouped interrupt control register (CTRL, addresses 0x4001 0000 (GINT0) and 0x4001 4000 (GINT1)) bit description | 146 |
| Table 138. GPIO port word pin registers (W[0:49], address offsets [0x1000:0x10C4]) bit description. | 115 | Table 170. GPIO grouped interrupt port polarity registers (PORT_POL[0:1], addresses 0x4001 0020 (PORT_POL0) to 0x4001 0024 (PORT_POL1) (GINT0) and 0x4001 4020 (PORT_POL0) to 0x4001 4024 (PORT_POL1) (GINT1)) bit | |
| Table 139. Address map DIR[0:1] registers. | 116 | | |
| Table 140. GPIO direction port register (DIR[0:1], address offset [0x2000:0x2004]) bit description | 116 | | |
| Table 141. Address map MASK[0:1] registers. | 116 | | |
| Table 142. GPIO mask port register (MASK[0:1], address offset [0x2080:0x2084]) bit description | 116 | | |
| Table 143. Address map PIN[0:1] registers. | 116 | | |
| Table 144. GPIO port pin register (PIN[0:1], address offset [0x2100:0x2104]) bit description. | 116 | | |
| Table 145. Address map MPIN[0:1] registers | 117 | | |
| Table 146. GPIO masked port pin register (MPIN[0:1], address offset [0x2180:0x2184]) bit description | 117 | | |

| | | | |
|--|-----|--|-----|
| description | 146 | Table 202. SCT0 pin description (inputs) | 172 |
| Table 171. GPIO grouped interrupt port enable registers (PORT_ENA[0:1], addresses 0x4001 0040 (PORT_ENA0) to 0x4001 0044 (PORT_ENA1) (GINT0) and 0x4001 4040 (PORT_ENA0) to 0x4001 4044 (PORT_ENA1) (GINT1)) bit description | 147 | Table 203. SCT0 pin description (outputs) | 172 |
| Table 172. DMA trigger sources | 149 | Table 204. Suggested SCT input pin settings | 172 |
| Table 173. DMA requests | 150 | Table 205. Suggested SCT output pin settings | 173 |
| Table 174. Channel descriptor | 152 | Table 206. Register overview: State Configurable Timer SCT/PWM (base address 0x5000 4000) | 176 |
| Table 175. Reload descriptors | 152 | Table 207. SCT configuration register (CONFIG, address 0x5000 4000) bit description | 182 |
| Table 176. Channel descriptor for a single transfer | 153 | Table 208. SCT control register (CTRL, address 0x5000 4004) bit description | 184 |
| Table 177. Example descriptors for ping-pong operation: peripheral to buffer | 154 | Table 209. SCT limit event select register (LIMIT, address 0x5000 4008) bit description | 185 |
| Table 178. Register overview: DMA controller (base address 0x1C00 4000) | 155 | Table 210. SCT halt event select register (HALT, address 0x5000 400C) bit description | 186 |
| Table 179. Control register (CTRL, address 0x1C00 4000) bit description | 158 | Table 211. SCT stop event select register (STOP, address 0x5000 4010) bit description | 186 |
| Table 180. Interrupt Status register (INTSTAT, address 0x1C00 4004) bit description | 158 | Table 212. SCT start event select register (START, address 0x5000 4014) bit description | 187 |
| Table 181. SRAM Base address register (SRAMBASE, address 0x1C00 4008) bit description | 158 | Table 213. SCT counter register (COUNT, address 0x5000 4040) bit description | 187 |
| Table 182. Channel descriptor map | 159 | Table 214. SCT state register (STATE, address 0x5000 4044) bit description | 188 |
| Table 183. Enable read and Set register 0 (ENABLESET0, address 0x1C00 4020) bit description | 159 | Table 215. SCT input register (INPUT, address 0x5000 4048) bit description | 189 |
| Table 184. Enable Clear register 0 (ENABLECLR0, address 0x1C00 4028) bit description | 160 | Table 216. SCT match/capture mode register (REGMODE, address 0x5000 404C) bit description | 189 |
| Table 185. Active status register 0 (ACTIVE0, address 0x1C00 4030) bit description | 160 | Table 217. SCT output register (OUTPUT, address 0x5000 4050) bit description | 190 |
| Table 186. Busy status register 0 (BUSY0, address 0x1C00 4038) bit description | 160 | Table 218. SCT bidirectional output control register (OUTPUTDIRCTRL, address 0x5000 4054) bit description | 190 |
| Table 187. Error Interrupt register 0 (ERRINT0, address 0x1C00 4040) bit description | 161 | Table 219. SCT conflict resolution register (RES, address 0x5000 4058) bit description | 191 |
| Table 188. Interrupt Enable read and Set register 0 (INTENSET0, address 0x1C00 4048) bit description | 161 | Table 220. SCT DMA 0 request register (DMAREQ0, address 0x5000 405C) bit description | 192 |
| Table 189. Interrupt Enable Clear register 0 (INTENCLR0, address 0x1C00 4050) bit description | 161 | Table 221. SCT DMA 1 request register (DMAREQ1, address 0x5000 4060) bit description | 192 |
| Table 190. Interrupt A register 0 (INTA0, address 0x1C00 4058) bit description | 162 | Table 222. SCT event interrupt enable register (EVEN, address 0x5000 40F0) bit description | 192 |
| Table 191. Interrupt B register 0 (INTB0, address 0x1C00 4060) bit description | 162 | Table 223. SCT event flag register (EVFLAG, address 0x5000 40F4) bit description | 192 |
| Table 192. Set Valid 0 register (SETVALID0, address 0x1C00 4068) bit description | 162 | Table 224. SCT conflict interrupt enable register (CONEN, address 0x5000 40F8) bit description | 193 |
| Table 193. Set Trigger 0 register (SETTRIG0, address 0x1C00 4070) bit description | 163 | Table 225. SCT conflict flag register (CONFLAG, address 0x5000 40FC) bit description | 193 |
| Table 194. Abort 0 register (ABORT0, address 0x1C00 4078) bit description | 163 | Table 226. SCT match registers 0 to 12 (MATCH[0:12], address 0x5000 4100 (MATCH0) to 0x5000 4130 (MATCH12)) bit description (REGMODEn bit = 0) | 194 |
| Table 195. Address map CFG[0:21] registers | 164 | Table 227. SCT capture registers 0 to 12 (CAP[0:12], address 0x5000 4100 (CAP0) to 0x5000 4130 (CAP12)) bit description (REGMODEn bit = 1) | 194 |
| Table 196. Channel configuration registers bit description | 164 | Table 228. SCT match reload registers 0 to 12 (MATCHREL[0:12], address 0x5000 4200 (MATCHRELO) to 0x5000 4230 (MATCHREL12)) bit description (REGMODEn bit = 0) | 194 |
| Table 197. Trigger setting summary | 165 | Table 229. SCT capture control registers 0 to 12 | |
| Table 198. Address map CTLSTAT[0:21] registers | 166 | | |
| Table 199. Channel control and status registers bit description | 166 | | |
| Table 200. Address map XFERCFG[0:21] registers | 167 | | |
| Table 201. Channel transfer configuration registers bit description | 167 | | |

| | | | |
|---|-----|---|-----|
| (CAPCTRL[0:12], address 0x5000 4200 (CAPCTRL0) to 0x5000 4230 (CAPCTRL12)) bit description (REGMODEn bit = 1) | 195 | Table 261. Address map PWMC register | 221 |
| Table 230. SCT event state mask registers 0 to 12 (EV[0:12]_STATE, addresses 0x5000 4300 (EV0_STATE) to 0x5000 4360 (EV12_STATE)) bit description | 195 | Table 262. PWM Control Register (PWMC, address offset 0x074) bit description | 222 |
| Table 231. SCT event control register 0 to 12 (EV[0:12]_CTRL, address 0x5000 4304 (EV0_CTRL) to 0x5000 4364 (EV12_CTRL)) bit description | 196 | Table 263. Register overview: Watchdog timer (base address 0x4003 8000) | 229 |
| Table 232. SCT output set register (OUT[0:7]_SET, address 0x5000 4500 (OUT0_SET) to 0x5000 4538 (OUT7_SET) bit description | 197 | Table 264. Watchdog mode register (MOD, 0x4003 8000) bit description | 229 |
| Table 233. SCT output clear register (OUT[0:7]_CLR, address 0x5000 4504 (OUT0_CLR) to 0x5000 453C (OUT7_CLR)) bit description. | 198 | Table 265. Watchdog operating modes selection | 230 |
| Table 234. Event conditions | 202 | Table 266. Watchdog Timer Constant register (TC, 0x4003 8004) bit description | 231 |
| Table 235. SCT configuration example | 206 | Table 267. Watchdog Feed register (FEED, 0x4003 8008) bit description | 231 |
| Table 236. Timer/Counter pin description | 211 | Table 268. Watchdog Timer Value register (TV, 0x4003 800C) bit description | 231 |
| Table 237. Suggested CT32B timer pin settings | 211 | Table 269. Watchdog Timer Warning Interrupt register (WARNINT, 0x4003 8014) bit description | 232 |
| Table 238. Register overview: CT32B0/1/2/3 (register base addresses 0x400B 4000 (CT32B0), 0x400B 8000 (CT32B1), 0x4000 4000 (CT32B2), 0x4000 8000 (CT32B3), 0x4000 C000 (CT32B4)) | 212 | Table 270. Watchdog Timer Window register (WINDOW, 0x4003 8018) bit description | 232 |
| Table 239. Address map IR register | 213 | Table 271. RTC pin description. | 236 |
| Table 240. Interrupt Register (IR, address offset 0x000) bit description | 213 | Table 272. Register overview: RTC (base address 0x4003 C000) | 237 |
| Table 241. Address map TCR register | 213 | Table 273. RTC control register (CTRL, address 0x4003 C000) bit description | 237 |
| Table 242. Timer Control Register (TCR, address offset 0x004) bit description | 213 | Table 274. RTC match register (MATCH, address 0x4003 C004) bit description | 238 |
| Table 243. Address map TC register | 214 | Table 275. RTC counter register (COUNT, address 0x4003 C008) bit description | 238 |
| Table 244. Timer counter registers (TC, address offset 0x08) bit description | 214 | Table 276. RTC high-resolution/wake-up register (WAKE, address 0x4003 C00C) bit description | 239 |
| Table 245. Address map PR register | 215 | Table 277. Register overview: MRT (base address 0x4007 4000) | 243 |
| Table 246. Timer prescale registers (PR, address offset 0x00C) bit description | 215 | Table 278. Time interval register (INTVAL[0:3], address 0x4007 4000 (INTVAL0) to 0x4007 4030 (INTVAL3)) bit description | 244 |
| Table 247. Address map PC register | 215 | Table 279. Timer register (TIMER[0:3], address 0x4007 4004 (TIMER0) to 0x4007 4034 (TIMER3)) bit description | 244 |
| Table 248. Timer prescale counter registers (PC, address offset 0x010) bit description | 215 | Table 280. Control register (CTRL[0:3], address 0x4007 4008 (CTRL0) to 0x4007 4038 (CTRL3)) bit description | 245 |
| Table 249. Address map MCR register | 215 | Table 281. Status register (STAT[0:3], address 0x4007 400C (STAT0) to 0x4007 403C (STAT3)) bit description 245 | |
| Table 250. Match Control Register (MCR, address offset 0x014) bit description | 216 | Table 282. Module Configuration register (MODCFG, address 0x4007 40F0) bit description | 246 |
| Table 251. Address map MR[0:3] registers | 216 | Table 283. Idle channel register (IDLE_CH, address 0x4007 40F4) bit description | 246 |
| Table 252. Timer match registers (MR[0:3], address offset [0x018:0x024]) bit description. | 217 | Table 284. Global interrupt flag register (IRQ_FLAG, address 0x4007 40F8) bit description | 247 |
| Table 253. Address map CCR register | 217 | Table 285. Register overview: Repetitive Interrupt Timer (RIT) (base address 0x4007 0000) | 250 |
| Table 254. Capture Control Register (CCR, address offset 0x028) bit description | 217 | Table 286. RI Compare Value LSB register (COMPVAL, address 0x4007 0000) bit description | 250 |
| Table 255. Address map CR[0:3] registers | 218 | Table 287. RI Mask LSB register (MASK, address 0x4007 0004) bit description | 250 |
| Table 256. Timer capture registers (CR[0:3], address offsets [0x02C:0x038]) bit description | 218 | Table 288. RI Control register (CTRL, address 0x4007 0008) bit description | 250 |
| Table 257. Address map EMR register | 218 | Table 289. RI Counter register (COUNTER, address | |
| Table 258. Timer external match registers (EMR, address offset 0x03C) bit description | 219 | | |
| Table 259. Address map CTCR register | 220 | | |
| Table 260. Count Control Register (CTCR, address offset 0x070) bit description | 220 | | |

| | | | |
|---|-----|---|-----|
| 0x4007 000C) bit description | 251 | | |
| Table 290. RI Compare Value MSB register (COMPVAL_H, address 0x4007 0010) bit description. | 251 | Table 319. SPI Configuration register (CFG, offset 0x00) bit description | 288 |
| Table 291. RI Mask MSB register (MASK_H, address 0x4007 0014) bit description | 251 | Table 320. SPI Delay register (DLY, offset 0x04) bit description | 290 |
| Table 292. RI Counter MSB register (COUNTER_H, address 0x4007 001C) bit description | 252 | Table 321. SPI Status register (STAT, offset 0x08) bit description | 291 |
| Table 293. Register overview: SysTick timer (base address 0xE000 E000). | 255 | Table 322. SPI Interrupt Enable read and Set register (INTENSET, offset 0x0C) bit description | 292 |
| Table 294. SysTick Timer Control and status register (SYST_CSR - 0xE000 E010) bit description | 255 | Table 323. SPI Interrupt Enable clear register (INTENCLR, offset 0x10) bit description | 293 |
| Table 295. System Timer Reload value register (SYST_RVR - 0xE000 E014) bit description | 255 | Table 324. SPI Receiver Data register (RXDAT, offset 0x14) bit description | 294 |
| Table 296. System Timer Current value register (SYST_CVR - 0xE000 E018) bit description | 256 | Table 325. SPI Transmitter Data and Control register (TXDATCTL, offset 0x18) bit description | 295 |
| Table 297. System Timer Calibration value register (SYST_CALIB - 0xE000 E01C) bit description. | 256 | Table 326. SPI Transmitter Data Register (TXDAT, offset 0x1C) bit description | 297 |
| Table 298. Register overview: Micro-Tick Timer (base address 0x4002 0000) | 259 | Table 327. SPI Transmitter Control register (TXCTL, offset 0x20) bit description. | 297 |
| Table 299. Control register (CTRL, address offset 0x00) bit description | 259 | Table 328. SPI Divider register (DIV, offset 0x24) bit description | 298 |
| Table 300. Status register (STAT, address offset 0x04) bit description | 259 | Table 329. SPI Interrupt Status register (INTSTAT, offset 0x28) bit description. | 298 |
| Table 301. USART pin description. | 264 | Table 330. SPI mode summary. | 299 |
| Table 302. Suggested USART pin settings | 264 | Table 331. I ² C-bus pin description | 307 |
| Table 303. Register overview: USART (base address 0x4008 4000 (USART0), 0x4008 8000 (USART1), 0x4008 C000 (USART2), 0x0x4009 0000 (USART3)) | 267 | Table 332. Code example | 308 |
| Table 304. USART Configuration register (CFG, offset 0x00) bit description | 268 | Table 333. Code example | 309 |
| Table 305. USART Control register (CTL, offset 0x04) bit description | 271 | Table 334. Code example | 310 |
| Table 306. USART Status register (STAT, offset 0x08) bit description | 272 | Table 335. Code example | 311 |
| Table 307. USART Interrupt Enable read and set register (INTENSET, offset 0x0C) bit description | 273 | Table 336. I ² C base addresses. | 313 |
| Table 308. USART Interrupt Enable clear register (INTENCLR, offset 0x10) bit description | 274 | Table 337. Register overview: I2C0/1/2 (register base addresses 0x4009 4000 (I2C0), 0x4009 8000 (I2C1), 0x4009 C000 (I2C2)) | 314 |
| Table 309. USART Receiver Data register (RXDAT, offset 0x14) bit description | 274 | Table 338. Address map CFG register | 315 |
| Table 310. USART Receiver Data with Status register (RXDATSTAT, offset 0x18) bit description | 275 | Table 339. I2C Configuration register (CFG, address offset 0x000) bit description | 315 |
| Table 311. USART Transmitter Data Register (TXDAT, offset 0x1C) bit description. | 275 | Table 340. Address map STAT register | 317 |
| Table 312. USART Baud Rate Generator register (BRG, offset 0x20) bit description | 276 | Table 341. I ² C Status register (STAT, address offset 0x004) bit description | 317 |
| Table 313. USART Interrupt Status register (INTSTAT, offset 0x24) bit description | 277 | Table 342. Master function state codes (MSTSTATE) | 320 |
| Table 314. Oversample selection register (OSR, offset 0x28) bit description | 277 | Table 343. Slave function state codes (SLVSTATE) | 320 |
| Table 315. Address register (ADDR, offset 0x2C) bit description | 278 | Table 344. Address map INTENSET register | 321 |
| Table 316. SPI Pin Description | 286 | Table 345. Interrupt Enable Set and read register (INTENSET, address offset 0x008) bit description | 321 |
| Table 317. Suggested SPI pin settings | 286 | Table 346. Address map INTENCLR register. | 322 |
| Table 318. Register overview: SPI (base address 0x400A 4000 (SPI0) and 0x400A 8000 (SPI1)) | 288 | Table 347. Interrupt Enable Clear register (INTENCLR, address offset 0x00C) bit description | 322 |
| | | Table 348. Address map TIMEOUT register. | 323 |
| | | Table 349. Time-out value register (TIMEOUT, address offset 0x010) bit description | 323 |
| | | Table 350. Address map CLDIV register | 324 |
| | | Table 351. I ² C Clock Divider register (CLKDIV, offset 0x14) bit description | 324 |
| | | Table 352. Address map INTSTAT register. | 324 |
| | | Table 353. I ² C Interrupt Status register (INTSTAT, address offset 0x018) bit description | 324 |
| | | Table 354. Address map MSTCTL register. | 326 |

| | | | |
|---|-----|--|-----|
| Table 355. Master Control register (MSTCTL, address offset 0x020) bit description | 326 | (CTLSETUSART[0:3], address offset [0x100C:0x130C]) bit description | 349 |
| Table 356. Address map MSTTIME register | 327 | Table 387. Address map CTLCLRUSARTT[0:3] registers | 350 |
| Table 357. Master Time register (MSTTIME, address offset 0x024) bit description | 327 | Table 388. Control read and clear register for USARTn (CTLCLRUSART[0:3], address offset [0x1010:0x1310]) bit description | 350 |
| Table 358. Address map MSTDAT register | 328 | Table 389. Address map RXDATUSART[0:3] registers | 350 |
| Table 359. Master Data register (MSTDAT, address offset 0x028) bit description | 328 | Table 390. Received data register for USARTn (RXDATUSART[0:3], address offset [0x1014:0x1314]) bit description | 351 |
| Table 360. Address map SLVCTL register | 329 | Table 391. Address map RXDATSTATUSART[0:3] registers | 351 |
| Table 361. Slave Control register (SLVCTL, address offset 0x040) bit description | 329 | Table 392. Received data with status register for USARTn (RXDATSTATUSART[0:3], address offset [0x1018:0x1318]) bit description | 351 |
| Table 362. Address map SLVDAT register | 329 | Table 393. Address map TXDATUSART[0:3] registers | 351 |
| Table 363. Slave Data register (SLVDAT, address offset 0x044) bit description | 330 | Table 394. Transmit data register for USARTn (TXDATUSART[0:3], address offset [0x101C:0x131C]) bit description | 351 |
| Table 364. Address map SLVADR[0:3] registers | 330 | Table 395. Address map CFGSPI[0:1] registers | 352 |
| Table 365. Slave Address registers (SLVADR[0:3], address offset [0x048:0x054]) bit description | 330 | Table 396. Configuration register for SPIn (CFGSPI[0:1], address offset [0x2000:0x2100]) bit description | 352 |
| Table 366. Address map SLVQUAL0 register | 330 | Table 397. Address map STATSPI[0:1] registers | 353 |
| Table 367. Slave address Qualifier 0 register (SLVQUAL0, address offset 0x058) bit description | 331 | Table 398. Status register for SPIn (STATSPI[0:1], address offset [0x2004:0x2104]) bit description | 353 |
| Table 368. Address map MONRXDAT register | 332 | Table 399. Address map INTSTATSPI[0:1] registers | 354 |
| Table 369. Monitor data register (MONRXDAT, address offset 0x080) bit description | 332 | Table 400. Interrupt status register for SPIn (INTSTATSPI[0:1], address offset [0x2008:0x2108]) bit description | 354 |
| Table 370. Register overview: FIFO register map (base address 0x1C03 8000) | 340 | Table 401. Address map CTLSETSPI[0:1] registers | 354 |
| Table 371. USART FIFO global control register (FIFOCTUSART, address offset 0x0100) bit description | 343 | Table 402. Control read and set register for SPIn (CTLSETSPI[0:1], address offset [0x200C:0x210C]) bit description | 354 |
| Table 372. USART FIFO global reset register (FIFOUPDATEUSART, address offset 0x0104) bit description | 343 | Table 403. Address map CTLCLRSPi[0:1] registers | 355 |
| Table 373. Address map FIFOCFGUSART[0:3] registers | 344 | Table 404. Control read and clear register for SPIn (CTLCLRSPi[0:1], address offset [0x2010:0x2110]) bit description | 355 |
| Table 374. FIFO configuration register for USARTn (FIFOCFGUSART[0:3], address offset [0x0110:0x011C]) bit description | 344 | Table 405. Address map RXDATSPI[0:1] registers | 355 |
| Table 375. SPI FIFO global control register (FIFOCTLSPI, address offset 0x0200) bit description | 345 | Table 406. Received data register for SPIn (RXDATSPI[0:1], address offset [0x2014:0x2114]) bit description | 356 |
| Table 376. SPI FIFO global reset register (FIFOUPDATESPI, address offset 0x0204) bit description | 345 | Table 407. Address map TXDATCTLSPI[0:1] registers | 356 |
| Table 377. Address map FIFOCFGSPI[0:1] registers | 346 | Table 408. Transmit data register for SPIn (TXDATCTLSPI[0:1], address offset [0x2018:0x2118]) bit description | 357 |
| Table 378. FIFO configuration register for SPIn (FIFOCFGSPI[0:1], address offset [0x210:0x214]) bit description | 346 | Table 409. ADC common supply and reference pins | 364 |
| Table 379. Address map CFGUSART[0:3] registers | 347 | Table 410. ADC0 pin description | 364 |
| Table 380. Configuration register for USARTn (CFGUSART[0:3], address offset [0x1000:0x1300]) bit description | 347 | Table 411. Suggested ADC input pin settings | 365 |
| Table 381. Address map STATUSART[0:3] registers | 348 | Table 412. Register overview: ADC (base address 0x1C03 4000) | 367 |
| Table 382. Status register for USARTn (STATUSART[0:3], address offset [0x1004:0x1304]) bit description | 348 | Table 413. ADC Control Register (CTRL, address offset 0x0) bit description | 369 |
| Table 383. Address map INTSTATUSART[0:3] registers | 349 | Table 416. ADC Sequence A Global Data Register (SEQA_GDAT, address offset 0x10) bit description | 376 |
| Table 384. Interrupt status register for USARTn (INTSTATUSART[0:3], address offset [0x1008:0x1308]) bit description | 349 | Table 417. ADC Sequence B Global Data Register | |
| Table 385. Address map CTLSETUSART[0:3] registers | 349 | | |
| Table 386. Control read and set register for USARTn | | | |

| | | | |
|---|-----|---|-----|
| (SEQB_GDAT, address offset 0x14) bit description | 377 | 0x1C04 0000) | 411 |
| Table 418. Address map DAT[0:11] registers | 378 | Table 447. Command and Status Word register (CSW, offset 0x00) bit description. | 411 |
| Table 419. ADC Data Registers (DAT[0:11], address offset [0x020:0x04C]) bit description | 378 | Table 448. Request value register (REQUEST, offset 0x04) bit description. | 411 |
| Table 420. ADC Compare Low Threshold register 0 (THR0_LOW, address offset 0x50) bit description 380 | | Table 449. Return value register (RETURN, offset 0x08) bit description | 411 |
| Table 421. ADC Compare Low Threshold register 1 (THR1_LOW, address offset 0x54) bit description 380 | | Table 450. Identification register (ID, offset 0xFC) bit description | 412 |
| Table 422. Compare High Threshold register0 (THR0_HIGH, address offset 0x58) bit description | 381 | Table 451. Register overview: ISP-AP (base address 0x1C04 0000) | 412 |
| Table 423. Compare High Threshold register 1 (THR1_HIGH, address offset 0x5C) bit description | 381 | Table 452. Register overview: ISP-AP (base address 0x1C04 0000) | 412 |
| Table 424. ADC Channel Threshold Select register (CHAN_THRSEL, address offset 0x60) bit description | 382 | Table 453. Power API ROM calls | 414 |
| Table 425. ADC Interrupt Enable register (INTEN, address offset 0x64) bit description | 383 | Table 454. set_pll routine | 415 |
| Table 426. ADC Flags register (FLAGS, address offset 0x68) bit description | 385 | Table 455. Error codes | 415 |
| Table 427. ADC Startup register (STARTUP, address offset 0x6C) bit description | 387 | Table 456. set_voltage routine | 415 |
| Table 428. ADC Calibration register (CALIB, address offset 0x70) bit description | 387 | Table 457. Error codes | 416 |
| Table 429. ADC0 hardware trigger inputs | 392 | Table 458. power_mode_configure routine. | 416 |
| Table 430. Minimum required sample times | 393 | Table 459. LPC5410x flash configurations | 418 |
| Table 431. Register overview: CRC engine (base address 0x1C01 0000) | 398 | Table 460. Flash sectors and pages | 420 |
| Table 432. CRC mode register (MODE, address 0x1C01 0000) bit description | 398 | Table 461. Code Read Protection (CRP) options | 421 |
| Table 433. CRC seed register (SEED, address 0x1C01 0004) bit description | 398 | Table 462. ISP commands allowed for different CRP levels. 421 | |
| Table 434. CRC checksum register (SUM, address 0x1C01 0008) bit description | 399 | Table 463. USART ISP command summary. | 424 |
| Table 435. CRC data register (WR_DATA, address 0x1C01 0008) bit description | 399 | Table 464. USART ISP Unlock command. | 424 |
| Table 436. Register overview: Mailbox (base address 0x1C02 C000) | 402 | Table 465. USART ISP Set Baud Rate command | 425 |
| Table 437. M0+ interrupt register (IRQ0, address 0x1C02 C000) bit description | 402 | Table 466. USART ISP Echo command | 425 |
| Table 438. M0+ interrupt set register (IRQ0SET, address 0x1C02 C004) bit description | 402 | Table 467. USART ISP Write to RAM command | 425 |
| Table 439. M0+ interrupt clear register (IRQ0CLR, address 0x1C02 C008) bit description | 402 | Table 468. USART ISP Read Memory command | 426 |
| Table 440. M4 interrupt (IRQ1, address 0x1C02 C010) bit description | 403 | Table 469. USART ISP Prepare sectors for write operation command | 426 |
| Table 441. M4 interrupt set register (IRQ1SET, address 0x1C02 C014) bit description | 403 | Table 470. USART ISP Copy command | 427 |
| Table 442. M4 interrupt clear register (IRQ1CLR, address 0x1C02 C018) bit description | 403 | Table 471. USART ISP Go command. | 427 |
| Table 443. Mutual Exclusion register (MUTEX, address 0x1C02 C0F8) bit description | 403 | Table 472. USART ISP Erase sector command | 428 |
| Table 444. Serial Wire Debug pin description | 406 | Table 473. USART ISP Erase page command | 428 |
| Table 445. JTAG boundary scan pin description | 407 | Table 474. USART ISP Blank check sector command | 429 |
| Table 446. Register overview: ISP-AP (base address | | Table 475. USART ISP Read Part Identification command . 429 | |
| | | Table 476. LPCA5410x device identification numbers | 429 |
| | | Table 477. USART ISP Read Boot Code version number command | 429 |
| | | Table 478. USART ISP Compare command. | 430 |
| | | Table 479. USART ReadUID command | 430 |
| | | Table 480. USART ISP Read CRC checksum command. | 431 |
| | | Table 481. USART ISP Read flash signature command. | 431 |
| | | Table 482. USART ISP Error codes | 431 |
| | | Table 483. IAP Command Summary | 435 |
| | | Table 484. IAP Prepare sector(s) for write operation command | 436 |
| | | Table 485. IAP Copy RAM to flash command. | 436 |
| | | Table 486. IAP Erase Sector(s) command | 437 |
| | | Table 487. IAP Blank check sector(s) command | 437 |
| | | Table 488. IAP Read Part Identification command | 437 |
| | | Table 489. IAP Read Boot Code version number command 438 | |
| | | Table 490. IAP Compare command | 438 |
| | | Table 491. Reinvoke ISP | 438 |

Table 492. IAP ReadUID command.439
Table 493. IAP Erase page command.439
Table 494. IAP Read Signature command439
Table 495. IAP Status codes Summary.440
Table 496. Abbreviations443

34.5 Figures

| | | | |
|---|-----|---|-----|
| Fig 1. Block diagram | 6 | Fig 46. SPI block diagram | 287 |
| Fig 2. Memory mapping | 10 | Fig 47. Basic SPI operating modes | 299 |
| Fig 3. Clock generation | 25 | Fig 48. Pre_delay and Post_delay | 300 |
| Fig 4. Start-up timing | 64 | Fig 49. Frame_delay | 301 |
| Fig 5. PLL block diagram showing typical operation | 65 | Fig 50. Transfer_delay | 302 |
| Fig 6. PLL block diagram showing spread spectrum and fractional divide operation | 69 | Fig 51. Examples of data stalls | 305 |
| Fig 7. Boot process flowchart | 82 | Fig 52. I ² C block diagram | 312 |
| Fig 8. Pin configuration | 94 | Fig 53. System FIFO conceptual block diagram | 339 |
| Fig 9. Pin interrupt multiplexing | 107 | Fig 54. FIFO system | 339 |
| Fig 10. DMA trigger multiplexing | 107 | Fig 55. ADC clocking | 363 |
| Fig 11. Pin interrupt connections | 123 | Fig 56. ADC connections | 363 |
| Fig 12. Pattern match engine connections | 124 | Fig 57. ADC block diagram | 366 |
| Fig 13. Pattern match bit slice with detect logic | 125 | Fig 58. CRC block diagram | 397 |
| Fig 14. Pattern match engine examples: sticky edge detect 142 | | Fig 59. Connecting the SWD pins to a standard SWD connector | 409 |
| Fig 15. Pattern match engine examples: Windowed non-sticky edge detect evaluates as true | 142 | Fig 60. Serial Wire Debug internal connections | 409 |
| Fig 16. Pattern match engine examples: Windowed non-sticky edge detect evaluates as false | 143 | Fig 61. ROM power API pointer structure | 414 |
| Fig 17. DMA block diagram | 151 | Fig 62. IAP parameter passing | 435 |
| Fig 18. SCT clocking | 171 | | |
| Fig 19. SCT connections | 171 | | |
| Fig 20. SCTimer/PWM block diagram | 174 | | |
| Fig 21. SCTimer/PWM counter and select logic | 175 | | |
| Fig 22. SCT event configuration and selection registers | 179 | | |
| Fig 23. Match logic | 199 | | |
| Fig 24. Capture logic | 199 | | |
| Fig 25. Event selection | 200 | | |
| Fig 26. Output slice i | 200 | | |
| Fig 27. SCT interrupt generation | 201 | | |
| Fig 28. SCT configuration example | 206 | | |
| Fig 29. 32-bit counter/timer block diagram | 210 | | |
| Fig 30. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled | 222 | | |
| Fig 31. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled | 223 | | |
| Fig 32. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register | 223 | | |
| Fig 33. WWDT timing | 226 | | |
| Fig 34. Windowed Watchdog timer block diagram | 227 | | |
| Fig 35. Early watchdog feed with windowed mode enabled 233 | | | |
| Fig 36. Correct watchdog feed with windowed mode enabled | 233 | | |
| Fig 37. Watchdog warning interrupt | 233 | | |
| Fig 38. RTC clocking | 235 | | |
| Fig 39. MRT block diagram | 241 | | |
| Fig 40. Repetitive Interrupt Timer (RI Timer) block diagram 249 | | | |
| Fig 41. System tick timer block diagram | 254 | | |
| Fig 42. Micro-Tick Timer block diagram | 258 | | |
| Fig 43. USART clocking | 262 | | |
| Fig 44. USART block diagram | 266 | | |
| Fig 45. Hardware flow control using RTS and CTS | 281 | | |

34.6 Contents

Chapter 1: LPC5410x Introductory information

| | | | | | |
|-----|---------------|---|-----|--|---|
| 1.1 | Introduction | 3 | 1.4 | Architectural overview | 7 |
| 1.2 | Features | 3 | 1.5 | ARM Cortex-M4 processor | 7 |
| 1.3 | Block diagram | 6 | 1.6 | ARM Cortex-M0+ processor (present on LPC54102 devices) | 7 |

Chapter 2: LPC5410x Memory mapping

| | | | | | |
|---------|---------------------|---|-------|------------------------------|----|
| 2.1 | General description | 8 | 2.1.2 | Memory mapping | 10 |
| 2.1.1 | Main SRAM | 8 | 2.1.3 | AHB multilayer matrix | 11 |
| 2.1.1.1 | SRAM2 | 8 | 2.1.4 | Memory Protection Unit (MPU) | 11 |
| 2.1.1.2 | SRAM usage notes | 8 | | | |

Chapter 3: LPC5410x Nested Vectored Interrupt Controller (NVIC)

| | | | | | |
|-------|---|----|--------|-------------------------------------|----|
| 3.1 | How to read this chapter | 12 | 3.4.10 | Interrupt Active Bit Register 1 | 18 |
| 3.2 | Features | 12 | 3.4.11 | Interrupt Priority Register 0 | 18 |
| 3.3 | General description | 12 | 3.4.12 | Interrupt Priority Register 1 | 18 |
| 3.3.1 | Interrupt sources | 12 | 3.4.13 | Interrupt Priority Register 2 | 19 |
| 3.4 | Register description | 14 | 3.4.14 | Interrupt Priority Register 3 | 19 |
| 3.4.1 | Interrupt Set-Enable Register 0 register | 15 | 3.4.15 | Interrupt Priority Register 4 | 19 |
| 3.4.2 | Interrupt Set-Enable Register 1 register | 16 | 3.4.16 | Interrupt Priority Register 5 | 20 |
| 3.4.3 | Interrupt Clear-Enable Register 0 | 16 | 3.4.17 | Interrupt Priority Register 6 | 20 |
| 3.4.4 | Interrupt Clear-Enable Register 1 register | 16 | 3.4.18 | Interrupt Priority Register 7 | 20 |
| 3.4.5 | Interrupt Set-Pending Register 0 register | 16 | 3.4.19 | Interrupt Priority Register 8 | 21 |
| 3.4.6 | Interrupt Set-Pending Register 1 register | 17 | 3.4.20 | Interrupt Priority Register 9 | 21 |
| 3.4.7 | Interrupt Clear-Pending Register 0 register | 17 | 3.4.21 | Interrupt Priority Register 10 | 21 |
| 3.4.8 | Interrupt Clear-Pending Register 1 register | 17 | 3.4.22 | Software Trigger Interrupt Register | 22 |
| 3.4.9 | Interrupt Active Bit Register 0 | 18 | | | |

Chapter 4: LPC5410x System configuration (SYSCON)

| | | | | | |
|--------|---|----|--------|---|----|
| 4.1 | Features | 23 | 4.5.14 | Reset captured value of port 0 | 33 |
| 4.2 | Basic configuration | 23 | 4.5.15 | Reset captured value of port 1 | 33 |
| 4.2.1 | Set up the PLL | 23 | 4.5.16 | Main clock source select register A | 34 |
| 4.2.2 | Configure the main clock and system clock | 23 | 4.5.17 | Main clock source select register B | 34 |
| 4.2.3 | Measure the frequency of a clock signal | 24 | 4.5.18 | ADC clock source select register | 34 |
| 4.3 | Pin description | 24 | 4.5.19 | CLKOUT clock source select register A | 35 |
| 4.4 | General description | 24 | 4.5.20 | CLKOUT clock source select register B | 35 |
| 4.4.1 | Clock generation | 24 | 4.5.21 | System PLL clock source select register | 36 |
| 4.5 | Register description | 26 | 4.5.22 | AHB Clock Control register 0 | 37 |
| 4.5.1 | AHB matrix priority register | 28 | 4.5.23 | AHB Clock Control register 1 | 38 |
| 4.5.2 | System tick counter calibration register | 28 | 4.5.24 | AHB Clock Control Set register 0 | 38 |
| 4.5.3 | NMI source selection register | 29 | 4.5.25 | AHB Clock Control Set register 1 | 38 |
| 4.5.4 | Asynchronous APB Control register | 29 | 4.5.26 | AHB Clock Control Clear register 0 | 38 |
| 4.5.5 | System reset status register | 30 | 4.5.27 | AHB Clock Control Clear register 1 | 39 |
| 4.5.6 | Peripheral reset control register 0 | 30 | 4.5.28 | SYSTICK clock divider register | 39 |
| 4.5.7 | Peripheral reset control register 1 | 31 | 4.5.29 | System clock divider register | 39 |
| 4.5.8 | Peripheral reset control set register 0 | 32 | 4.5.30 | ADC clock source divider register | 39 |
| 4.5.9 | Peripheral reset control set register 1 | 32 | 4.5.31 | CLKOUT clock divider register | 40 |
| 4.5.10 | Peripheral reset control clear register 0 | 32 | 4.5.32 | Frequency measure function control register | 40 |
| 4.5.11 | Peripheral reset control clear register 1 | 33 | 4.5.33 | Flash configuration register | 41 |
| 4.5.12 | POR captured value of port 0 | 33 | 4.5.34 | FIFO control register | 42 |
| 4.5.13 | POR captured value of port 1 | 33 | 4.5.35 | IRC control register | 43 |
| | | | 4.5.36 | RTC oscillator control register | 43 |

| | | | | | |
|------------|--|----|------------|--|-----------|
| 4.5.37 | PLL registers | 44 | 4.5.54 | Asynchronous APB clock control register | 60 |
| 4.5.37.1 | System PLL control register | 44 | 4.5.55 | Asynchronous APB clock control set register | 60 |
| 4.5.37.2 | System PLL status register | 45 | 4.5.56 | Asynchronous APB clock control clear register | 61 |
| 4.5.37.3 | System PLL N-divider register | 45 | 4.5.57 | Asynchronous clock source select register A | 61 |
| 4.5.37.4 | System PLL P-divider register | 46 | 4.5.58 | Asynchronous clock source select register B | 61 |
| 4.5.37.5 | Spread spectrum control with PLL0 | 46 | 4.5.59 | Asynchronous APB clock divider register | 62 |
| 4.5.37.5.1 | System PLL spread spectrum control register 0 | 47 | 4.5.60 | USART fractional baud rate generator register | 62 |
| 4.5.37.5.2 | System PLL spread spectrum control register 1 | 48 | 4.5.61 | BOD control register | 63 |
| 4.5.38 | Power Configuration register | 50 | 4.6 | Functional description | 63 |
| 4.5.39 | Power configuration set register | 50 | 4.6.1 | Reset | 63 |
| 4.5.40 | Power configuration clear register | 51 | 4.6.2 | Start-up behavior | 64 |
| 4.5.41 | Start enable register 0 | 52 | 4.6.3 | Brown-out detection | 65 |
| 4.5.42 | Start enable register 1 | 53 | 4.6.4 | PLL functional description | 65 |
| 4.5.43 | Start enable set register 0 | 54 | 4.6.4.1 | PLL Features | 65 |
| 4.5.44 | Start enable set register 1 | 54 | 4.6.4.2 | PLL description | 66 |
| 4.5.45 | Start enable clear register 0 | 54 | 4.6.4.2.1 | Lock detector | 66 |
| 4.5.46 | Start enable clear register 1 | 55 | 4.6.4.2.2 | Power-down | 67 |
| 4.5.47 | Dual-CPU related registers | 56 | 4.6.4.3 | Operating modes | 67 |
| 4.5.47.1 | CPU Control register | 56 | 4.6.4.3.1 | Normal modes | 67 |
| 4.5.47.2 | Coprocessor Boot register | 57 | | Normal mode with optional pre-divide | 67 |
| 4.5.47.3 | Coprocessor Stack register | 57 | | Normal mode with post-divide and optional pre-divide | 68 |
| 4.5.47.4 | Coprocessor Status register | 57 | 4.6.4.3.2 | Fractional divider mode | 68 |
| 4.5.48 | JTAG ID code register | 58 | 4.6.4.3.3 | Spread Spectrum mode | 69 |
| 4.5.49 | Device ID0 register | 58 | 4.6.4.3.4 | Power-down mode | 69 |
| 4.5.50 | Device ID1 register | 58 | 4.6.4.4 | PLL Related registers | 69 |
| 4.5.51 | Asynchronous peripheral reset control register | 59 | 4.6.4.5 | PLL usage | 70 |
| 4.5.52 | Asynchronous peripheral reset control set register | 59 | 4.6.4.5.1 | Procedure for determining PLL settings | 70 |
| 4.5.53 | Asynchronous peripheral reset control clear register | 59 | 4.6.4.5.2 | PLL setup sequence | 70 |
| | | | 4.6.5 | Frequency measure function | 71 |
| | | | 4.6.5.1 | Accuracy | 71 |

Chapter 5: LPC5410x Power Management

| | | | | | |
|------------|--|-----------|---------|--|----|
| 5.1 | Introduction | 73 | 5.3.4.3 | Wake-up from Deep-sleep mode | 78 |
| 5.2 | General description | 73 | 5.3.5 | Power-down mode | 79 |
| 5.2.1 | Wake-up process | 74 | 5.3.5.1 | Power configuration in Power-down mode | 79 |
| 5.3 | Functional description | 76 | 5.3.5.2 | Programming Power-down mode | 79 |
| 5.3.1 | Power management | 76 | 5.3.5.3 | Wake-up from Power-down mode | 79 |
| 5.3.2 | Active mode | 76 | 5.3.6 | Deep power-down mode | 80 |
| 5.3.2.1 | Power configuration in Active mode | 76 | 5.3.6.1 | Power configuration in Deep power-down mode | 80 |
| 5.3.3 | Sleep mode | 77 | | | |
| 5.3.3.1 | Power configuration in Sleep mode | 77 | 5.3.6.2 | Wake-up from Deep power-down mode | 80 |
| 5.3.3.2 | Programming Sleep mode | 77 | 5.3.6.3 | Programming Deep power-down mode using the RTC for wake-up | 80 |
| 5.3.3.3 | Wake-up from Sleep mode | 77 | 5.3.6.4 | Wake-up from Deep power-down mode using the RTC | 80 |
| 5.3.4 | Deep-sleep mode | 77 | | | |
| 5.3.4.1 | Power configuration in Deep-sleep mode | 78 | | | |
| 5.3.4.2 | Programming Deep-sleep mode | 78 | | | |

Chapter 6: LPC5410x Boot process

| | | | | | |
|------------|------------------------|-----------|------------|----------------------------|-----------|
| 6.1 | Features | 81 | 6.3 | General description | 81 |
| 6.2 | Pin description | 81 | 6.3.1 | Boot process flowchart | 82 |

Chapter 7: LPC5410x Pin description

| | | |
|------------|------------------------|-----------|
| 7.1 | Pin description | 83 |
|------------|------------------------|-----------|

Chapter 8: LPC5410x I/O pin configuration (IOCON)

| | | | | | |
|------------|---|-----------|------------|---|-----------|
| 8.1 | How to read this chapter | 93 | 8.4.2.5 | Analog/digital mode | 95 |
| 8.2 | Features | 93 | 8.4.2.6 | Input filter | 96 |
| 8.3 | Basic configuration | 93 | 8.4.2.7 | Output slew rate | 96 |
| 8.4 | General description | 94 | 8.4.2.8 | I ² C modes | 96 |
| 8.4.1 | Pin configuration | 94 | 8.4.2.9 | Open-Drain Mode | 96 |
| 8.4.2 | IOCON registers | 94 | 8.5 | Register description | 97 |
| | Multiple connections | 94 | 8.5.1 | Type D IOCON registers (PIO0) | 98 |
| 8.4.2.1 | Pin function | 95 | 8.5.2 | Type I IOCON registers (PIO0) | 100 |
| 8.4.2.2 | Pin mode | 95 | 8.5.3 | Type A IOCON registers (PIO0, PIO1) | 101 |
| 8.4.2.3 | Hysteresis | 95 | 8.5.4 | Type D IOCON registers (PIO1) | 104 |
| 8.4.2.4 | Invert pin | 95 | | | |

Chapter 9: LPC5410x Input multiplexing (INPUT MUX)

| | | | | | |
|------------|--|------------|-------|---|-----|
| 9.1 | How to read this chapter | 106 | 9.6.1 | Pin interrupt select registers | 109 |
| 9.2 | Features | 106 | 9.6.2 | DMA trigger input mux registers 0 to 21 | 110 |
| 9.3 | Basic configuration | 106 | 9.6.3 | DMA output trigger feedback mux registers 0 to 3 110 | |
| 9.4 | Pin description | 106 | 9.6.4 | Frequency measure function reference clock select register | 112 |
| 9.5 | General description | 106 | 9.6.5 | Frequency measure function target clock select register | 112 |
| 9.5.1 | Pin interrupt input multiplexing | 107 | | | |
| 9.5.2 | DMA trigger input multiplexing | 107 | | | |
| 9.6 | Register description | 108 | | | |

Chapter 10: LPC5410x General Purpose I/O (GPIO)

| | | | | | |
|-------------|---|------------|-------------|--|------------|
| 10.1 | How to read this chapter | 113 | 10.5.6 | GPIO masked port pin registers | 117 |
| 10.2 | Basic configuration | 113 | 10.5.7 | GPIO port set registers | 117 |
| 10.3 | Features | 113 | 10.5.8 | GPIO port clear registers | 117 |
| 10.4 | General description | 113 | 10.5.9 | GPIO port toggle registers | 118 |
| 10.5 | Register description | 114 | 10.6 | Functional description | 119 |
| 10.5.1 | GPIO port byte pin registers | 115 | 10.6.1 | Reading pin state | 119 |
| 10.5.2 | GPIO port word pin registers | 115 | 10.6.2 | GPIO output | 119 |
| 10.5.3 | GPIO port direction registers | 115 | 10.6.3 | Masked I/O | 120 |
| 10.5.4 | GPIO port mask registers | 116 | 10.6.4 | Recommended practices | 120 |
| 10.5.5 | GPIO port pin registers | 116 | | | |

Chapter 11: LPC5410x Pin interrupt and pattern match (PINT)

| | | | | | |
|-------------|--|------------|-------------|---|------------|
| 11.1 | How to read this chapter | 121 | 11.6.4 | Pin interrupt level or rising edge interrupt clear register | 129 |
| 11.2 | Features | 121 | 11.6.5 | Pin interrupt active level or falling edge interrupt enable register | 129 |
| 11.3 | Basic configuration | 122 | 11.6.6 | Pin interrupt active level or falling edge interrupt set register | 129 |
| 11.3.1 | Configure pins as pin interrupts or as inputs to the pattern match engine | 122 | 11.6.7 | Pin interrupt active level or falling edge interrupt clear register | 130 |
| 11.4 | Pin description | 123 | 11.6.8 | Pin interrupt rising edge register | 130 |
| 11.5 | General description | 123 | 11.6.9 | Pin interrupt falling edge register | 131 |
| 11.5.1 | Pin interrupts | 123 | 11.6.10 | Pin interrupt status register | 131 |
| 11.5.2 | Pattern match engine | 123 | 11.6.11 | Pattern Match Interrupt Control Register | 131 |
| 11.5.2.1 | Example | 126 | 11.6.12 | Pattern Match Interrupt Bit-Slice Source register. 132 | |
| 11.6 | Register description | 127 | 11.6.13 | Pattern Match Interrupt Bit Slice Configuration register | 134 |
| 11.6.1 | Pin interrupt mode register | 128 | 11.7 | Functional description | 140 |
| 11.6.2 | Pin interrupt level or rising edge interrupt enable register | 128 | | | |
| 11.6.3 | Pin interrupt level or rising edge interrupt set register | 128 | | | |

| | | | | | |
|--------|--|-----|--------|---|-----|
| 11.7.1 | Pin interrupts | 140 | 11.7.3 | Pattern match engine edge detect examples | 142 |
| 11.7.2 | Pattern Match engine example | 140 | | | |

Chapter 12: LPC5410x Group GPIO input interrupt (GINT0/1)

| | | | | | |
|-------------|---------------------------------------|------------|-------------|--|------------|
| 12.1 | Features | 144 | 12.4.1 | Grouped interrupt control register | 146 |
| 12.2 | Basic configuration | 144 | 12.4.2 | GPIO grouped interrupt port polarity registers | 146 |
| 12.3 | General description | 144 | 12.4.3 | GPIO grouped interrupt port enable registers | 146 |
| 12.4 | Register description | 145 | 12.5 | Functional description | 147 |

Chapter 13: LPC5410x DMA controller

| | | | | | |
|-------------|--|------------|-------------|--|------------|
| 13.1 | How to read this chapter | 148 | 13.6.3 | SRAM Base address register | 158 |
| 13.2 | Features | 148 | 13.6.4 | Enable read and Set registers | 159 |
| 13.3 | Basic configuration | 148 | 13.6.5 | Enable Clear register | 159 |
| 13.3.1 | Hardware triggers | 149 | 13.6.6 | Active status register | 160 |
| 13.3.2 | Trigger outputs | 149 | 13.6.7 | Busy status register | 160 |
| 13.3.3 | DMA requests | 150 | 13.6.8 | Error Interrupt register | 161 |
| 13.3.4 | DMA in sleep mode | 150 | 13.6.9 | Interrupt Enable read and Set register | 161 |
| 13.4 | Pin description | 151 | 13.6.10 | Interrupt Enable Clear register | 161 |
| 13.5 | General description | 151 | 13.6.11 | Interrupt A register | 161 |
| 13.5.1 | DMA requests and triggers | 151 | 13.6.12 | Interrupt B register | 162 |
| 13.5.2 | DMA Modes | 152 | 13.6.13 | Set Valid register | 162 |
| 13.5.3 | Single buffer | 153 | 13.6.14 | Set Trigger register | 162 |
| 13.5.4 | Ping-Pong | 153 | 13.6.15 | Abort registers | 163 |
| 13.5.5 | Linked transfers (linked list) | 154 | 13.6.16 | Channel configuration registers | 164 |
| 13.5.6 | Address alignment for data transfers | 154 | 13.6.17 | Channel control and status registers | 166 |
| 13.6 | Register description | 155 | 13.6.18 | Channel transfer configuration registers | 167 |
| 13.6.1 | Control register | 158 | 13.7 | Functional description | 169 |
| 13.6.2 | Interrupt Status register | 158 | 13.7.1 | Trigger operation | 169 |

Chapter 14: LPC5410x SCTimer/PWM (SCT0)

| | | | | | |
|-------------|--|------------|---------|--|-----|
| 14.1 | How to read this chapter | 170 | 14.6.4 | SCT limit event select register | 185 |
| 14.2 | Features | 170 | 14.6.5 | SCT halt event select register | 185 |
| 14.3 | Basic configuration | 171 | 14.6.6 | SCT stop event select register | 186 |
| 14.4 | Pin description | 172 | 14.6.7 | SCT start event select register | 186 |
| 14.5 | General description | 173 | 14.6.8 | SCT counter register | 187 |
| 14.6 | Register description | 175 | 14.6.9 | SCT state register | 187 |
| 14.6.1 | Register functional grouping | 178 | 14.6.10 | SCT input register | 188 |
| 14.6.1.1 | Counter configuration and control registers | 180 | 14.6.11 | SCT match/capture mode register | 189 |
| 14.6.1.2 | Event configuration registers | 180 | 14.6.12 | SCT output register | 189 |
| 14.6.1.3 | Match and capture registers | 180 | 14.6.13 | SCT bi-directional output control register | 190 |
| 14.6.1.4 | Event select registers for the counter operations | 180 | 14.6.14 | SCT conflict resolution register | 191 |
| 14.6.1.5 | Event select registers for setting or clearing the outputs | 181 | 14.6.15 | SCT DMA request 0 and 1 registers | 191 |
| 14.6.1.6 | Event select registers for capturing a counter value | 181 | 14.6.16 | SCT event interrupt enable register | 192 |
| 14.6.1.7 | Event select register for initiating DMA transfers | 181 | 14.6.17 | SCT event flag register | 192 |
| 14.6.1.8 | Interrupt handling registers | 181 | 14.6.18 | SCT conflict interrupt enable register | 192 |
| 14.6.1.9 | Registers for controlling SCT inputs and outputs by software | 181 | 14.6.19 | SCT conflict flag register | 193 |
| 14.6.2 | SCT configuration register | 182 | 14.6.20 | SCT match registers 0 to 12 (REGMODEn bit = 0) | 193 |
| 14.6.3 | SCT control register | 183 | 14.6.21 | SCT capture registers 0 to 12 (REGMODEn bit = 1) | 194 |
| | | | 14.6.22 | SCT match reload registers 0 to 12 (REGMODEn bit = 0) | 194 |
| | | | 14.6.23 | SCT capture control registers 0 to 12 (REGMODEn bit = 1) | 194 |

| | | | | | |
|-------------|-------------------------------------|------------|-----------|---|-----|
| 14.6.24 | SCT event enable registers 0 to 12 | 195 | 14.7.8 | Match vs. I/O events | 201 |
| 14.6.25 | SCT event control registers 0 to 12 | 195 | 14.7.9 | SCT operation | 202 |
| 14.6.26 | SCT output set registers 0 to 7 | 197 | 14.7.10 | Configure the SCT | 203 |
| 14.6.27 | SCT output clear registers 0 to 7 | 197 | 14.7.10.1 | Configure the counter | 203 |
| 14.7 | Functional description | 199 | 14.7.10.2 | Configure the match and capture registers | 203 |
| 14.7.1 | Match logic | 199 | 14.7.10.3 | Configure events and event responses | 203 |
| 14.7.2 | Capture logic | 199 | 14.7.10.4 | Configure multiple states | 204 |
| 14.7.3 | Event selection | 199 | 14.7.10.5 | Miscellaneous options | 204 |
| 14.7.4 | Output generation | 200 | 14.7.11 | Run the SCT | 204 |
| 14.7.5 | State logic | 200 | 14.7.12 | Configure the SCT without using states | 205 |
| 14.7.6 | Interrupt generation | 201 | 14.7.13 | SCT PWM Example | 205 |
| 14.7.7 | Clearing the prescaler | 201 | | | |

Chapter 15: LPC5410x Standard counter/timers (CT32B0/1/2/3/4)

| | | | | | |
|-------------|---------------------------------|------------|-------------|--|------------|
| 15.1 | How to read this chapter | 208 | 15.7.3 | Timer Counter registers | 214 |
| 15.2 | Basic configuration | 208 | 15.7.4 | Prescale register | 215 |
| 15.3 | Features | 208 | 15.7.5 | Prescale Counter register | 215 |
| 15.4 | Applications | 209 | 15.7.6 | Match Control Register | 215 |
| 15.5 | General description | 209 | 15.7.7 | Match Registers | 216 |
| 15.5.1 | Capture inputs | 209 | 15.7.8 | Capture Control Register | 217 |
| 15.5.2 | Match outputs | 209 | 15.7.9 | Capture Registers | 218 |
| 15.5.3 | Applications | 209 | 15.7.10 | External Match Register | 218 |
| 15.5.4 | Architecture | 209 | 15.7.11 | Count Control Register | 220 |
| 15.6 | Pin description | 211 | 15.7.12 | PWM Control Register | 221 |
| 15.6.1 | Multiple CAP and MAT pins | 211 | 15.8 | Functional description | 222 |
| 15.7 | Register description | 212 | 15.8.1 | Rules for single edge controlled PWM outputs | 223 |
| 15.7.1 | Interrupt Register | 213 | 15.8.2 | DMA operation | 224 |
| 15.7.2 | Timer Control Register | 213 | | | |

Chapter 16: LPC5410x Windowed Watchdog Timer (WWDT)

| | | | | | |
|-------------|---------------------------------|------------|-------------|---|------------|
| 16.1 | How to read this chapter | 225 | 16.5.3.2 | Changing the WWDT reload value | 228 |
| 16.2 | Features | 225 | 16.6 | Register description | 229 |
| 16.3 | Basic configuration | 226 | 16.6.1 | Watchdog mode register | 229 |
| 16.4 | Pin description | 226 | 16.6.2 | Watchdog Timer Constant register | 231 |
| 16.5 | General description | 226 | 16.6.3 | Watchdog Feed register | 231 |
| 16.5.1 | Block diagram | 227 | 16.6.4 | Watchdog Timer Value register | 231 |
| 16.5.2 | Clocking and power control | 228 | 16.6.5 | Watchdog Timer Warning Interrupt register | 232 |
| 16.5.3 | Using the WWDT lock features | 228 | 16.6.6 | Watchdog Timer Window register | 232 |
| 16.5.3.1 | Disabling the WWDT clock source | 228 | 16.7 | Functional description | 233 |

Chapter 17: LPC5410x Real-Time Clock (RTC)

| | | | | | |
|-------------|---------------------------------|------------|-------------|--------------------------------------|------------|
| 17.1 | How to read this chapter | 234 | 17.4.3 | RTC power | 236 |
| 17.2 | Features | 234 | 17.5 | Pin description | 236 |
| 17.3 | Basic configuration | 234 | 17.6 | Register description | 237 |
| 17.3.1 | RTC timers | 235 | 17.6.1 | RTC CTRL register | 237 |
| 17.4 | General description | 236 | 17.6.2 | RTC match register | 238 |
| 17.4.1 | Real-time clock | 236 | 17.6.3 | RTC counter register | 238 |
| 17.4.2 | High-resolution/wake-up timer | 236 | 17.6.4 | RTC high-resolution/wake-up register | 239 |

Chapter 18: LPC5410x Multi-Rate Timer (MRT)

| | | | | | |
|-------------|---------------------------------|------------|-------------|----------------------------|------------|
| 18.1 | How to read this chapter | 240 | 18.3 | Basic configuration | 240 |
| 18.2 | Features | 240 | 18.4 | Pin description | 240 |

| | | | | | |
|-------------|--|------------|--------|---|-----|
| 18.5 | General description | 240 | 18.6.2 | Timer register (TIMER) | 244 |
| 18.5.1 | Repeat interrupt mode | 241 | 18.6.3 | Control register (CTRL) | 245 |
| 18.5.2 | One-shot interrupt mode. | 241 | 18.6.4 | Status register (STAT) | 245 |
| 18.5.3 | One-shot stall mode | 242 | 18.6.5 | Module Configuration register (MODCFG) | 246 |
| 18.6 | Register description | 243 | 18.6.6 | Idle channel register (IDLE_CH) | 246 |
| 18.6.1 | Time interval register (INTVAL). | 244 | 18.6.7 | Global interrupt flag register (IRQ_FLAG) | 247 |

Chapter 19: LPC5410x Repetitive Interrupt Timer (RIT)

| | | | | | |
|-------------|---|------------|-------------|---|------------|
| 19.1 | How to read this chapter | 248 | 19.5.3 | RI Control register | 250 |
| 19.2 | Basic configuration | 248 | 19.5.4 | RI Counter LSB register | 251 |
| 19.3 | Features | 248 | 19.5.5 | RI Compare Value MSB register | 251 |
| 19.4 | General description | 248 | 19.5.6 | RI Mask MSB register | 251 |
| 19.5 | Register description | 250 | 19.5.7 | RI Counter MSB register | 252 |
| 19.5.1 | RI Compare Value LSB register | 250 | 19.6 | RI timer operation | 252 |
| 19.5.2 | RI Mask LSB register | 250 | | | |

Chapter 20: LPC5410x CPU system tick timer (SYSTICK)

| | | | | | |
|-------------|--|------------|-------------|---|------------|
| 20.1 | How to read this chapter | 253 | 20.5.3 | System Timer Current value register | 256 |
| 20.2 | Basic configuration | 253 | 20.5.4 | System Timer Calibration value register | 256 |
| 20.3 | Features | 253 | 20.6 | Functional description | 257 |
| 20.4 | General description | 253 | 20.7 | Example timer calculations | 257 |
| 20.5 | Register description | 255 | | System clock = 72 MHz | 257 |
| 20.5.1 | System Timer Control and status register | 255 | | System tick timer clock = 24 MHz | 257 |
| 20.5.2 | System Timer Reload value register | 255 | | System clock = 12 MHz | 257 |

Chapter 21: LPC5410x Micro-tick timer (UTICK)

| | | | | | |
|-------------|---|------------|-------------|---------------------------------------|------------|
| 21.1 | How to read this chapter | 258 | 21.4 | General description | 258 |
| 21.2 | Features | 258 | 21.5 | Register description | 259 |
| 21.3 | Basic configuration | 258 | 21.5.1 | CTRL register | 259 |
| | | | 21.5.2 | Status register | 259 |

Chapter 22: LPC5410x USARTs (USART0/1/2/3)

| | | | | | |
|-------------|--|------------|-------------|--|------------|
| 22.1 | How to read this chapter | 260 | 22.6.9 | USART Baud Rate Generator register | 276 |
| 22.2 | Features | 260 | 22.6.10 | USART Interrupt Status register | 277 |
| 22.3 | Basic configuration | 261 | 22.6.11 | Oversample selection register | 277 |
| 22.3.1 | Configure the USART clock and baud rate. | 261 | 22.6.12 | Address register | 278 |
| 22.3.2 | Configure the USART for wake-up | 262 | 22.7 | Functional description | 279 |
| 22.3.2.1 | Wake-up from Sleep mode | 262 | 22.7.1 | Clocking and baud rates | 279 |
| 22.3.2.2 | Wake-up from Deep-sleep or Power-down mode. | 263 | 22.7.1.1 | Fractional Rate Generator (FRG) | 279 |
| 22.4 | Pin description | 264 | 22.7.1.2 | Baud Rate Generator (BRG) | 279 |
| 22.5 | General description | 265 | 22.7.1.3 | Baud rate calculations | 279 |
| 22.6 | Register description | 267 | 22.7.1.4 | 32 kHz mode | 279 |
| 22.6.1 | USART Configuration register | 268 | 22.7.2 | DMA | 280 |
| 22.6.2 | USART Control register | 271 | 22.7.3 | Synchronous mode | 280 |
| 22.6.3 | USART Status register | 272 | 22.7.4 | Flow control | 280 |
| 22.6.4 | USART Interrupt Enable read and set register | 273 | 22.7.4.1 | Hardware flow control | 280 |
| 22.6.5 | USART Interrupt Enable Clear register | 274 | 22.7.4.2 | Software flow control | 281 |
| 22.6.6 | USART Receiver Data register | 274 | 22.7.5 | Autobaud function | 281 |
| 22.6.7 | USART Receiver Data with Status register | 275 | 22.7.6 | RS-485 support | 281 |
| 22.6.8 | USART Transmitter Data Register | 275 | 22.7.7 | Oversampling | 282 |
| | | | 22.7.8 | Break generation and detection | 282 |
| | | | 22.7.9 | LIN bus | 282 |
| | | | 22.7.10 | IrDA communication | 283 |

Chapter 23: LPC5410x Serial Peripheral Interfaces (SPI0/1)

| | | | | | |
|-------------|--|------------|-------------|--|------------|
| 23.1 | How to read this chapter | 284 | 23.6.8 | SPI Transmitter Data Register | 297 |
| 23.2 | Features | 284 | 23.6.9 | SPI Transmitter Control register | 297 |
| 23.3 | Basic configuration | 284 | 23.6.10 | SPI Divider register | 298 |
| 23.3.1 | Configure the SPI for wake-up | 284 | 23.6.11 | SPI Interrupt Status register | 298 |
| 23.3.1.1 | Wake-up from Sleep mode | 285 | 23.7 | Functional description | 299 |
| 23.3.1.2 | Wake-up from Deep-sleep or Power-down mode | 285 | 23.7.1 | Operating modes: clock and phase selection | 299 |
| 23.4 | Pin description | 286 | 23.7.2 | Frame delays | 300 |
| 23.5 | General description | 287 | 23.7.2.1 | Pre_delay and Post_delay | 300 |
| 23.6 | Register description | 288 | 23.7.2.2 | Frame_delay | 301 |
| 23.6.1 | SPI Configuration register | 288 | 23.7.2.3 | Transfer_delay | 302 |
| 23.6.2 | SPI Delay register | 290 | 23.7.3 | Clocking and data rates | 303 |
| 23.6.3 | SPI Status register | 291 | 23.7.3.1 | Data rate calculations | 303 |
| 23.6.4 | SPI Interrupt Enable read and Set register | 292 | 23.7.4 | Slave select | 303 |
| 23.6.5 | SPI Interrupt Enable Clear register | 293 | 23.7.5 | DMA operation | 304 |
| 23.6.6 | SPI Receiver Data register | 294 | 23.7.5.1 | DMA master mode End-Of-Transfer | 304 |
| 23.6.7 | SPI Transmitter Data and Control register | 295 | 23.7.6 | Data lengths greater than 16 bits | 304 |
| | | | 23.7.7 | Data stalls | 304 |

Chapter 24: LPC5410x I2C-bus interfaces (I2C0/1/2)

| | | | | | |
|-------------|--|------------|-------------|---|------------|
| 24.1 | How to read this chapter | 306 | 24.6.9 | Master Time register | 327 |
| 24.2 | Features | 306 | 24.6.10 | Master Data register | 328 |
| 24.3 | Pin description | 307 | 24.6.11 | Slave Control register | 329 |
| 24.4 | Basic configuration | 307 | 24.6.12 | Slave Data register | 329 |
| 24.4.1 | I2C transmit/receive in master mode | 307 | 24.6.13 | Slave Address registers | 330 |
| 24.4.1.1 | Master write to slave | 308 | 24.6.14 | Slave address Qualifier 0 register | 330 |
| 24.4.1.2 | Master read from slave | 309 | 24.6.15 | Monitor data register | 332 |
| 24.4.2 | I2C receive/transmit in slave mode | 309 | 24.7 | Functional description | 333 |
| 24.4.2.1 | Slave read from master | 310 | 24.7.1 | Bus rates and timing considerations | 333 |
| 24.4.2.2 | Slave write to master | 311 | 24.7.1.1 | Rate calculations | 333 |
| 24.4.3 | Configure the I2C for wake-up | 311 | | Master timing | 333 |
| 24.4.3.1 | Wake-up from Sleep mode | 311 | | Slave timing | 333 |
| 24.4.3.2 | Wake-up from Deep-sleep and Power-down modes | 312 | 24.7.1.2 | Bus rate support | 333 |
| 24.5 | General description | 312 | 24.7.1.2.1 | High-speed mode support | 334 |
| 24.6 | Register description | 313 | 24.7.1.2.2 | Clock stretching | 334 |
| 24.6.1 | I2C Configuration register | 315 | 24.7.2 | Time-out | 335 |
| 24.6.2 | I2C Status register | 317 | 24.7.3 | Ten-bit addressing | 335 |
| 24.6.3 | Interrupt Enable Set and read register | 321 | 24.7.4 | Clocking and power considerations | 336 |
| 24.6.4 | Interrupt Enable Clear register | 322 | 24.7.5 | Interrupt handling | 336 |
| 24.6.5 | Time-out value register | 323 | 24.7.6 | DMA | 336 |
| 24.6.6 | Clock Divider register | 324 | 24.7.6.1 | DMA as a Master transmitter | 336 |
| 24.6.7 | Interrupt Status register | 324 | 24.7.6.2 | DMA as a Master receiver | 337 |
| 24.6.8 | Master Control register | 326 | 24.7.6.3 | DMA as a Slave transmitter | 337 |
| | | | 24.7.6.4 | DMA as a Slave receiver | 337 |

Chapter 25: LPC5410x System FIFO for Serial Peripherals

| | | | | | |
|-------------|--|------------|----------|--|-----|
| 25.1 | How to read this chapter | 338 | 25.5.3 | FIFO configuration register for USART0 to USART3 | 344 |
| 25.2 | Basic configuration | 338 | 25.5.4 | SPI FIFO global control register | 345 |
| 25.3 | Features | 338 | 25.5.5 | SPI FIFO global reset register | 345 |
| 25.4 | Architecture | 339 | 25.5.6 | FIFO configuration register for SPI0 and SPI1 | 346 |
| 25.5 | Register description | 340 | 25.5.7 | Configuration register for USARTn | 347 |
| 25.5.1 | USART FIFO global control register | 343 | 25.5.7.1 | Receiver Timeout | 347 |
| 25.5.2 | USART FIFO global update register | 343 | | | |

| | | | | | |
|-----------|---|-----|-------------|---|------------|
| 25.5.8 | Status register for USARTn | 348 | 25.5.21 | Transmit data register for SPIn | 356 |
| 25.5.9 | Interrupt status register for USARTn | 349 | 25.6 | Operational details | 359 |
| 25.5.10 | Control read and set register for USARTn | 349 | 25.6.1 | Configuring peripheral FIFOs | 359 |
| 25.5.11 | Control clear register for USARTn | 350 | 25.6.2 | Peripheral status | 359 |
| 25.5.12 | Received data register for USARTn | 350 | 25.6.3 | Receiving data | 359 |
| 25.5.13 | Received data with status register for USARTn | 351 | 25.6.3.1 | Receiver Timeouts | 359 |
| 25.5.14 | Transmit data register for USARTn | 351 | 25.6.4 | Transmitting data | 360 |
| 25.5.15 | Configuration register for SPI0 and SPI1 | 352 | 25.6.5 | Channel Priority | 360 |
| 25.5.15.1 | Receiver Timeout | 353 | 25.6.6 | Errors | 360 |
| 25.5.16 | Status register for SPIs | 353 | 25.7 | Generic FIFO setup | 361 |
| 25.5.17 | Interrupt status register for SPI0 and SPI1 | 354 | 25.7.1 | System FIFO activation | 361 |
| 25.5.18 | Control read and set register for SPIn | 354 | 25.7.2 | FIFO setup for USARTs | 361 |
| 25.5.19 | Control clear register for SPI0 and SPI1 | 355 | 25.7.3 | FIFO setup for SPIs | 361 |
| 25.5.20 | Received data register for SPI0 and SPI1 | 355 | | | |

Chapter 26: LPC5410x 12-bit ADC controller (ADC0)

| | | | | | |
|-------------|--|------------|-------------|--|------------|
| 26.1 | How to read this chapter | 362 | 26.6.12 | ADC Calibration register | 387 |
| 26.2 | Features | 362 | 26.7 | Functional description | 388 |
| 26.3 | Basic configuration | 362 | 26.7.1 | Conversion Sequences | 388 |
| 26.4 | Pin description | 364 | 26.7.2 | Hardware-triggered conversion | 388 |
| 26.5 | General description | 366 | 26.7.2.1 | Avoiding spurious hardware triggers | 389 |
| 26.6 | Register description | 367 | 26.7.3 | Software-triggered conversion | 389 |
| 26.6.1 | ADC Control Register | 369 | 26.7.4 | Interrupts | 389 |
| 26.6.2 | ADC Conversion Sequence A Control Register | 371 | 26.7.4.1 | Conversion-Complete or Sequence-Complete interrupts | 389 |
| 26.6.3 | ADC Conversion Sequence B Control Register | 374 | 26.7.4.2 | Threshold-Compare Out-of-Range Interrupt | 390 |
| 26.6.4 | ADC Global Data Register A and B | 376 | 26.7.4.3 | Data Overrun Interrupt | 390 |
| 26.6.5 | ADC Channel Data Registers 0 to 11 | 378 | 26.7.5 | Optional Operating Modes | 390 |
| 26.6.6 | ADC Compare Low Threshold Registers 0 and 1 | 380 | 26.7.6 | Offset calibration and enabling the ADC | 391 |
| 26.6.7 | ADC Compare High Threshold Registers 0 and 1 | 381 | 26.7.7 | ADC vs. digital receiver | 391 |
| 26.6.8 | ADC Channel Threshold Select register | 382 | 26.7.8 | DMA control | 391 |
| 26.6.9 | ADC Interrupt Enable Register | 383 | 26.7.9 | ADC hardware trigger inputs | 392 |
| 26.6.10 | ADC Flags register | 385 | 26.7.10 | Sample time | 392 |
| 26.6.11 | ADC Startup register | 387 | 26.8 | Examples | 394 |
| | | | 26.8.1 | Perform a single ADC conversion triggered by software | 394 |
| | | | 26.8.2 | Perform a sequence of conversions triggered by an external pin | 395 |

Chapter 27: LPC5410x CRC engine

| | | | | | |
|-------------|---|------------|-------------|---|------------|
| 27.1 | How to read this chapter | 396 | 27.6.2 | CRC seed register | 398 |
| 27.2 | Features | 396 | 27.6.3 | CRC checksum register | 399 |
| 27.3 | Basic configuration | 396 | 27.6.4 | CRC data register | 399 |
| 27.4 | Pin description | 396 | 27.7 | Functional description | 400 |
| 27.5 | General description | 397 | 27.7.1 | CRC-CCITT set-up | 400 |
| 27.6 | Register description | 398 | 27.7.2 | CRC-16 set-up | 400 |
| 27.6.1 | CRC mode register | 398 | 27.7.3 | CRC-32 set-up | 400 |

Chapter 28: LPC5410x Mailbox

| | | | | | |
|-------------|---|------------|-------------|---|------------|
| 28.1 | How to read this chapter | 401 | 28.5 | General description | 401 |
| 28.2 | Features | 401 | 28.6 | Register description | 402 |
| 28.3 | Basic configuration | 401 | 28.6.1 | M0+ interrupt register | 402 |
| 28.4 | Pin description | 401 | 28.6.2 | Cortex M0+ interrupt set register | 402 |

| | | | | | |
|--------|------------------------------|-----|--------|-----------------------------|-----|
| 28.6.3 | M0+ interrupt clear register | 402 | 28.6.6 | M4 interrupt clear register | 403 |
| 28.6.4 | M4 interrupt register | 403 | 28.6.7 | Mutual Exclusion register | 403 |
| 28.6.5 | M4 interrupt set register | 403 | | | |

Chapter 29: LPC5410x flash controller

| | | | | | |
|-------------|-------------------------------|------------|----------|----------------------|-----|
| 29.1 | Functional description | 404 | 29.1.1.1 | Signature generation | 404 |
| 29.1.1 | Flash signature generation | 404 | 29.1.1.2 | Content verification | 404 |

Chapter 30: LPC5410x Serial Wire Debug (SWD)

| | | | | | |
|-------------|--|------------|-------------|---|------------|
| 30.1 | How to read this chapter | 406 | 30.6.4.4 | Error handling | 410 |
| 30.2 | Features | 406 | 30.6.4.5 | Register description | 410 |
| 30.3 | Basic configuration | 406 | 30.6.4.5.1 | Command and Status Word register (CSW, offset 0x00) bit description | 411 |
| 30.4 | Pin description | 406 | 30.6.4.5.2 | Request value register (REQUEST, offset 0x04) bit description | 411 |
| 30.5 | General description | 408 | 30.6.4.5.3 | Return value register (RETURN, offset 0x08) bit description | 411 |
| 30.6 | Functional description | 408 | 30.6.4.5.4 | Identification register (ID, offset 0xFC) bit description | 412 |
| 30.6.1 | Debug limitations | 408 | 30.6.4.6 | ISP-AP commands | 412 |
| 30.6.2 | Debug connections for SWD | 408 | 30.6.4.7 | ISP-AP return codes | 412 |
| 30.6.3 | Boundary scan | 410 | 30.7 | Debug configuration | 412 |
| 30.6.4 | In-System Programming Access Port (ISP-AP) | 410 | 30.7.1 | Cortex-M4 | 412 |
| 30.6.4.1 | Resynchronization request | 410 | 30.7.2 | Cortex-M0+ (present on LPC54102 devices) | 412 |
| 30.6.4.2 | Acknowledgement of resynchronization request | 410 | | | |
| 30.6.4.3 | Return phase | 410 | | | |

Chapter 31: LPC5410x Power profiles/Power control API

| | | | | | |
|-------------|---------------------------------|------------|-------------|---|------------|
| 31.1 | How to read this chapter | 413 | 31.4.2.2 | Param1: frequency | 416 |
| 31.2 | Features | 413 | 31.4.2.3 | Error or return codes | 416 |
| 31.3 | General description | 413 | 31.4.3 | power_mode_configure | 416 |
| 31.4 | API description | 414 | 31.4.3.1 | Param0: mode | 416 |
| 31.4.1 | set_pll | 415 | 31.4.3.2 | Param1: peripheral | 417 |
| 31.4.1.1 | Param0: multiplier | 415 | 31.5 | Functional description | 417 |
| 31.4.1.2 | Param1: input_freq | 415 | 31.5.1 | Example low power mode control | 417 |
| 31.4.1.3 | Error or return codes | 415 | 31.5.1.1 | Enter sleep mode | 417 |
| 31.4.2 | set_voltage | 415 | 31.5.1.2 | Enter power-down mode and set up WWDT and BOD for wake-up | 417 |
| 31.4.2.1 | Param0: mode | 416 | | | |
| | Low power Mode | 416 | | | |

Chapter 32: LPC5410x Flash API

| | | | | | |
|-------------|------------------------------------|------------|-------------|-------------------------------------|------------|
| 32.1 | How to read this chapter | 418 | 32.4 | USART communication protocol | 423 |
| 32.2 | Features | 418 | 32.4.1 | USART ISP command format | 423 |
| 32.3 | General description | 418 | 32.4.2 | USART ISP response format | 423 |
| 32.3.1 | Boot loader | 418 | 32.4.3 | USART ISP data format | 423 |
| 32.3.2 | Memory map after any reset | 418 | 32.5 | USART ISP commands | 424 |
| 32.3.3 | Flash content protection mechanism | 418 | 32.5.1 | Unlock | 424 |
| 32.3.4 | Criteria for Valid User Code | 419 | 32.5.2 | Set Baud Rate | 425 |
| 32.3.5 | Flash partitions | 420 | 32.5.3 | Echo | 425 |
| 32.3.6 | Code Read Protection (CRP) | 420 | 32.5.4 | Write to RAM | 425 |
| 32.3.6.1 | ISP entry protection | 422 | 32.5.5 | Read Memory | 426 |
| 32.3.7 | ISP interrupt and SRAM use | 422 | 32.5.6 | Prepare sectors for write operation | 426 |
| 32.3.7.1 | Interrupts during IAP | 422 | 32.5.7 | Copy RAM to flash | 426 |
| 32.3.7.2 | RAM used by ISP command handler | 422 | 32.5.8 | Go | 427 |
| 32.3.7.3 | RAM used by IAP command handler | 422 | 32.5.9 | Erase sectors | 428 |
| | | | 32.5.10 | Erase pages | 428 |

| | | | | | |
|-------------|---|------------|---------|--|-----|
| 32.5.11 | Blank check sectors | 429 | 32.6.3 | Erase Sector(s) | 437 |
| 32.5.12 | Read Part Identification number | 429 | 32.6.4 | Blank check sector(s) | 437 |
| 32.5.13 | Read Boot code version number | 429 | 32.6.5 | Read Part Identification number | 437 |
| 32.5.14 | Compare | 430 | 32.6.6 | Read Boot code version number | 438 |
| 32.5.15 | ReadUID | 430 | 32.6.7 | Compare <address1> <address2> <no of bytes> 438 | |
| 32.5.16 | Read CRC checksum | 430 | 32.6.8 | Reinvoke ISP | 438 |
| 32.5.17 | Read flash signature | 431 | 32.6.9 | ReadUID | 439 |
| 32.5.18 | ISP Error codes | 431 | 32.6.10 | Erase page | 439 |
| 32.6 | IAP commands | 434 | 32.6.11 | Read Signature | 439 |
| 32.6.1 | Prepare sector(s) for write operation | 436 | 32.6.12 | IAP Status Codes | 440 |
| 32.6.2 | Copy RAM to flash | 436 | | | |

Chapter 33: ARM Cortex Appendix

| | | | | | |
|-------------|--|------------|-------------|---|------------|
| 33.1 | ARM Cortex-M4 Details | 441 | 33.2 | ARM Cortex-M0+ Details (present on LPC54102 devices) | 441 |
| 33.1.1 | Cortex-M4 implementation options | 441 | 33.2.1 | Cortex-M0+ implementation options | 442 |

Chapter 34: Supplementary information

| | | | | | |
|-------------|------------------------------------|------------|-------------|---------------------------|------------|
| 34.1 | Abbreviations | 443 | 34.3.3 | Trademarks | 445 |
| 34.2 | References | 444 | 34.4 | Tables | 446 |
| 34.3 | Legal information | 445 | 34.5 | Figures | 455 |
| 34.3.1 | Definitions | 445 | 34.6 | Contents | 456 |
| 34.3.2 | Disclaimers | 445 | | | |

Please be aware that important notices concerning this document and the product(s) described herein, have been included in section 'Legal information'.

© NXP B.V. 2015.

All rights reserved.

For more information, please visit: <http://www.nxp.com>

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 10 April 2015

Document identifier: UM10850

Mouser Electronics

Authorized Distributor

Click to View Pricing, Inventory, Delivery & Lifecycle Information:

[NXP:](#)

[OM13077](#) [OM13078](#)

Данный компонент на территории Российской Федерации

Вы можете приобрести в компании MosChip.

Для оперативного оформления запроса Вам необходимо перейти по данной ссылке:

<http://moschip.ru/get-element>

Вы можете разместить у нас заказ для любого Вашего проекта, будь то серийное производство или разработка единичного прибора.

В нашем ассортименте представлены ведущие мировые производители активных и пассивных электронных компонентов.

Нашей специализацией является поставка электронной компонентной базы двойного назначения, продукции таких производителей как XILINX, Intel (ex.ALTERA), Vicor, Microchip, Texas Instruments, Analog Devices, Mini-Circuits, Amphenol, Glenair.

Сотрудничество с глобальными дистрибьюторами электронных компонентов, предоставляет возможность заказывать и получать с международных складов практически любой перечень компонентов в оптимальные для Вас сроки.

На всех этапах разработки и производства наши партнеры могут получить квалифицированную поддержку опытных инженеров.

Система менеджмента качества компании отвечает требованиям в соответствии с ГОСТ Р ИСО 9001, ГОСТ РВ 0015-002 и ЭС РД 009

Офис по работе с юридическими лицами:

105318, г.Москва, ул.Щербаковская д.3, офис 1107, 1118, ДЦ «Щербаковский»

Телефон: +7 495 668-12-70 (многоканальный)

Факс: +7 495 668-12-70 (доб.304)

E-mail: info@moschip.ru

Skype отдела продаж:

moschip.ru

moschip.ru_4

moschip.ru_6

moschip.ru_9