# UM11065

## LPC804 User manual

Rev. 1.3 — 12 July 2018 User manual

### Document information

| Info | Content |
|---|---|
| **Keywords** | LPC804, LPC804 UM, LPC804 user manual |
| **Abstract** | LPC804 User manual |

## Revision history

| Rev | Date | Description |
|---|---|---|
| 1.3 | 20180712 | LPC804 User manual. |
| Modifications: | • Added LPC804UK device.<br>• Updated Section 22.3.1 “Perform a single ADC conversion using a software trigger”: Changed the result of the conversion: The ADC converts an analog input signal VIN on the ADC_[11:0]. The VREFP and VREFN pins provide a positive and negative reference voltage input. The result of the conversion is (4095 x (VIN-VREFN))/(VREFP - VREFN).<br>• Updated Table 84 “Power configuration register (PDRUNCFG, address 0x4004 8238) bit description”: Reset value of BOD_PD is 0. | |
| 1.2 | 20180321 | LPC804 User manual. |

**Revision history** *...continued*

| Rev | Date | Description |
|---|---|---|
| | | • Updated Table 52 “SYSCON pin description”: SWM register for CLKOUT function. Changed to PINASSGN5.<br>• Updated Section 17.2 “Features”:The watchdog reload or watchdog feed sequence can optionally be protected so that it can only be performed after the “warning interrupt” time is reached.<br>• Updated Section 17.5 “General description”. Added a bullet: Set the Watchdog timer update mode (WDPROTECT) in the MOD register after a delay of three WDCLK clock cycles.<br>• Updated Section 17.5.3 “Using the WWDT lock features”.<br>• Updated Section 17.5.3.2 “Changing the WWDT reload value”.<br>• Updated Table 238 “Watchdog mode register (MOD, 0x4000 0000) bit description”. New text for enumerated values of bit 4, WDPROTECT.<br>• Updated Section 17.6.2 “Watchdog Timer Constant register”. New text: If the WDPROTECT bit in WDMOD = 1, an attempt to perform the watchdog reload or watchdog feed sequence before the watchdog timer is below the values of WDWARNINT and WDWINDOW will cause a watchdog feed error and set the WDTOF flag.<br>• Updated Table 261 “Register overview: base address 0x4006 0000”. Changed address offset of INTENCLR to 0x014.<br>• Updated Section 19.6.6 “Idle channel register”.<br>• Updated Table 274 “Register overview : ADC (base address 0x4001 C000 )”: Changed access for SEQB_CTRL and SEQA_GDAT to RO.<br>• Updated Table 248 “Register overview: MRT (base address 0x4000 4000)”. Changed access for Timer0, 1, 2, and 3.<br>• Updated Table 193 “SPI Interrupt Enable read and Set register (INTENSET, addresses 0x4005 800C (SPI) bit description”, Table 194 “SPI Interrupt Enable clear register (INTENCLR, addresses 0x4005 8010 (SPI) bit description”, and Table 200 “SPI Interrupt Status register (INTSTAT, addresses 0x4005 8028 (SPI) bit description”.<br>• Updated Section 22.6.3 “A/D Conversion Sequence B Control Register” remark to: Set the BURST and SEQB_ENA bits at the same time. Fixed title of Table 277 “A/D Conversion Sequence B Control Register (SEQB_CTRL, address 0x4001 C00C) bit description”.<br>• Updated Table 276 “A/D Conversion Sequence A Control Register (SEQA_CTRL, address 0x4001 C008) bit description”. Removed TSAMP.<br>• Updated Section 22.6.6 “A/D Compare Low Threshold Registers 0 and 1”: Renamed THCMP_RANGE to THCMPRANGE and THCMP_CROSS to THCMPCROSS.<br>• Updated Table 255 “Global interrupt flag register (IRQ_FLAG, address 0x4000 40F8) bit description”.<br>• Updated Table 261 “Register overview: base address 0x4006 0000”: STATUS access is R/W.<br>• Updated Table 12 “USART ISP Write to RAM command”, Table 13 “USART ISP Read Memory command”, Table 15 “USART ISP Copy command”, Table 16 “USART ISP Go command”, Table 23 “USART ISP Compare command” and Table 25 “USART ISP Read CRC checksum command”: added on-chip RAM.<br>• Removed IBARO register from Chapter 5 “LPC804 Nested Vectored Interrupt Controller (NVIC)”. |

## Contact information

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

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

**Revision history** *...continued*

| Rev | Date | Description |
|---|---|---|
| 1.2 | 20180321 | LPC804 User manual. |
| | | • Added a remark to Table 102 “Pin enable register 0 (PINENABLE0, address 0x4000 C1C0) bit description”, and Section 9.4.5 “Analog mode” : In analog mode, the internal pull-up must be disabled via the IOCON register.<br>• Updated Table 89 “Movable functions (assign to pins PIO0_0 to PIO0_5 and PIO0_7 to PIO0_30 through switch matrix)”: Changed access of CAPT_YH to I/O.<br>• Updated Section 12.7.7 “Deep power-down mode”: added text: Five general-purpose registers are available to store information during deep power-down mode.<br>• Updated Section 22.3.1 “Perform a single ADC conversion using a software trigger”: Changed the result of the conversion: The ADC converts an analog input signal VIN on the ADC_[11:0]. The VREFP and VREFN pins provide a positive and negative reference voltage input. The result of the conversion is (4095 x (VIN-Vss))/(VREFP - Vss). |
| 1.1 | 20180213 | LPC804 User manual. |
| Modifications: | | • Added Level Shifter functionality to Section 1.2 “Features”.<br>• Updated Table 40 “Interrupt Set Enable Register 0 register (ISER0, address 0xE000 E100) bit description”, Table 41 “Interrupt clear enable register 0 (ICER0, address 0xE000 E180)”, Table 42 “Interrupt set pending register 0 register (ISPR0, address 0xE000 E200) bit description”, Table 43 “Interrupt clear pending register 0 register (ICPR0, address 0xE000 E280) bit description”. PLU, Bit 6 is reserved.<br>• Added Section 8.4.3 “Level Shifter function”.<br>• Added Section 19.6.5 “Module Configuration register”. |
| 1 | 20180124 | Initial revision. LPC804 User manual. |

## 1.1 Introduction

The LPC80x are an Arm® Cortex®-M0+ based, low-cost 32-bit MCU family operating at CPU frequencies of up to 15 MHz. The LPC804 support up to 32 KB of flash memory and 4 KB of SRAM.

The peripheral complement of the LPC80x includes a CRC engine, two $I^2C$-bus interfaces, two USARTs, one SPI interface, Capacitive Touch Interface (Cap Touch), one multi-rate timer, self-wake-up timer, one general purpose 32-bit counter/timer, one 12-bit ADC, one10-bit DAC, one analog comparator, function-configurable I/O ports through a switch matrix, an input pattern match engine, Programmable Logic Unit (PLU), and up to 30 general-purpose I/O pins.

**Remark:** For additional documentation, see Section 29.2 “References”.

## 1.2 Features

- System:
  - Arm Cortex-M0+ processor (revision r0p1), running at frequencies of up to 15 MHz with single-cycle multiplier and fast single-cycle I/O port.
  - Arm Cortex-M0+ built-in Nested Vectored Interrupt Controller (NVIC).
  - System tick timer.
  - AHB multilayer matrix.
  - Serial Wire Debug (SWD) with four break points and two watch points. JTAG boundary scan (BSDL) supported.
- Memory:
  - Up to 32 KB on-chip flash programming memory.
  - Code Read Protection (CRP).
  - 4 KB of SRAM.
- Dual I/O power (LPC804M111JDH24)
  - Independent supplies on each package side permitting level shifting signals from one off-chip voltage domain to another and/or interfacing directly to off-chip peripherals operating at different supply levels.
  - The switch matrix provides level shifter functionality to allow up to two selected signals to be routed from user-selected pins in one voltage domain to selected pins in the alternate domain. This feature can also be used on a single supply device if voltage level shifting is not required.
- ROM API support:
  - Bootloader.
  - Supports Flash In-Application Programming (IAP).
  - Supports In-System Programming (ISP) through USART.
  - On-chip ROM APIs for integer divide.

- Free Running Oscillator (FRO) API.

- Digital peripherals:
  - High-speed GPIO interface connected to the Arm Cortex-M0+ IO bus with up to 30 General-Purpose I/O (GPIO) pins with configurable pull-up/pull-down resistors, programmable open-drain mode, and input inverter. GPIO direction control supports independent set/clear/toggle of individual bits.
  - High-current source output driver (20 mA) on five pins.
  - GPIO interrupt generation capability with boolean pattern-matching feature on eight GPIO inputs.
  - Switch matrix for flexible configuration of each I/O pin function.
  - CRC engine.
  - Capacitive Touch Interface.
  - Programmable Logic Unit (PLU) to create small combinatorial and/or sequential logic networks including simple state machines.
- Timers:
  - One 32-bit general purpose counter/timer, with four match outputs and three capture inputs. Supports PWM mode, external count.
  - Four channel Multi-Rate Timer (MRT) for repetitive interrupt generation at up to four programmable, fixed rates.
  - Self-Wake-up Timer (WKT) clocked from either Free Running Oscillator (FRO), a low-power, low-frequency internal oscillator, or an external clock input.
  - Windowed Watchdog timer (WWDT).
- Analog peripherals:
  - One 12-bit ADC with up to 12 input channels with multiple internal and external trigger inputs and with sample rates of up to 480 Ksamples/s. The ADC supports two independent conversion sequences.
  - Comparator with up to five input pins and external or internal reference voltage.
  - One 10-bit DAC.
- Serial peripherals:
  - Two USART interfaces with pin functions assigned through the switch matrix and one fractional baud rate generators.
  - One SPI controller with pin functions assigned through the switch matrix.
  - Two $I^2C$-bus interfaces. It supports data rates up to 400 kbit/s on standard digital pins.
- Clock generation:
  - Free Running Oscillator (FRO). This oscillator provides a selectable 9 MHz, 12 MHz, and 15 MHz outputs that can be used as a system clock. The FRO is trimmed to ±1 % accuracy over the entire voltage and temperature range of 0 C to 70 C.
  - 1 MHz low power oscillator can be used as a clock source.
  - Clock output function with divider that can reflect all internal clock sources.
- Power control:

  - 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 $I^2C$ peripherals.
  - Wake-up from deep power-down mode on multiple pins.
  - Timer-controlled self wake-up from sleep, deep-sleep, and power-down modes.
  - Power-On Reset (POR).
  - Brownout detect (BOD).
- Unique device serial number for identification.
- Single or dual power supplies (1.71 V to 3.6 V).
- Operating temperature range -40 °C to +105 °C.
- Available in WLCSP20, TSSOP20, TSSOP24, and HVQFN33 packages.

## 1.3 Ordering options

**Table 1. Ordering information**

| Type number | Package | | |
|---|---|---|---|
| | Name | Description | Version |
| LPC804M101JDH20 | TSSOP20 | plastic thin shrink small outline package; 20 leads; body width 4.4 mm | SOT360-1 |
| LPC804M101JDH24 | TSSOP24 | plastic thin shrink small outline package; 24 leads; body width 4.4 mm | SOT355-1 |
| LPC804M111JDH24 | TSSOP24 | plastic thin shrink small outline package; 24 leads; body width 4.4 mm | SOT355-1 |
| LPC804M101JHI33 | HVQFN33 | HVQFN: plastic thermal enhanced very thin quad flat package; no leads; 33 terminals; body 5 × 5 × 0.85 mm | SOT617-11 |
| LPC804UK | WLCSP20 | wafer level chip-size package; 20 (5 × 4) bumps; 2.50 ×1.84 × 0.5 mm | SOT1397-8 |

**Table 2. Ordering options**

| Type number | Flash/KB | SRAM/KB | USART | I²C | SPI | DAC | Capacitive Touch | PLU | GPIO | Dual I/O power supply | Package |
|---|---|---|---|---|---|---|---|---|---|---|---|
| LPC804M101JDH20 | 32 | 4 | 2 | 2 | 1 | - | yes | yes | 17 | - | TSSOP20 |
| LPC804M101JDH24 | 32 | 4 | 2 | 2 | 1 | 1 | yes | yes | 21 | - | TSSOP24 |
| LPC804M111JDH24 | 32 | 4 | 2 | 2 | 1 | 1 | yes | yes | 20 | yes | TSSOP24 |
| LPC804M101JHI33 | 32 | 4 | 2 | 2 | 1 | 1 | yes | yes | 30 | - | HVQFN33 |
| LPC804UK | 32 | 4 | 2 | 2 | 1 | - | yes | yes | 17 | - | WLCSP20 |

## 1.4 General description

### 1.4.1 Arm Cortex-M0+ core configuration

The Arm Cortex-M0+ core runs at an operating frequency of up to 15 MHz. Integrated in the core are the NVIC and Serial Wire Debug with four breakpoints and two watch points. The Arm Cortex-M0+ core supports a single-cycle I/O enabled port (IOP) for fast GPIO access at address 0xA000 0000. The Arm Cortex M0+ core version is r0p1.

The core includes a single-cycle multiplier and a system tick timer (SysTick).

## 1.5 Block diagram

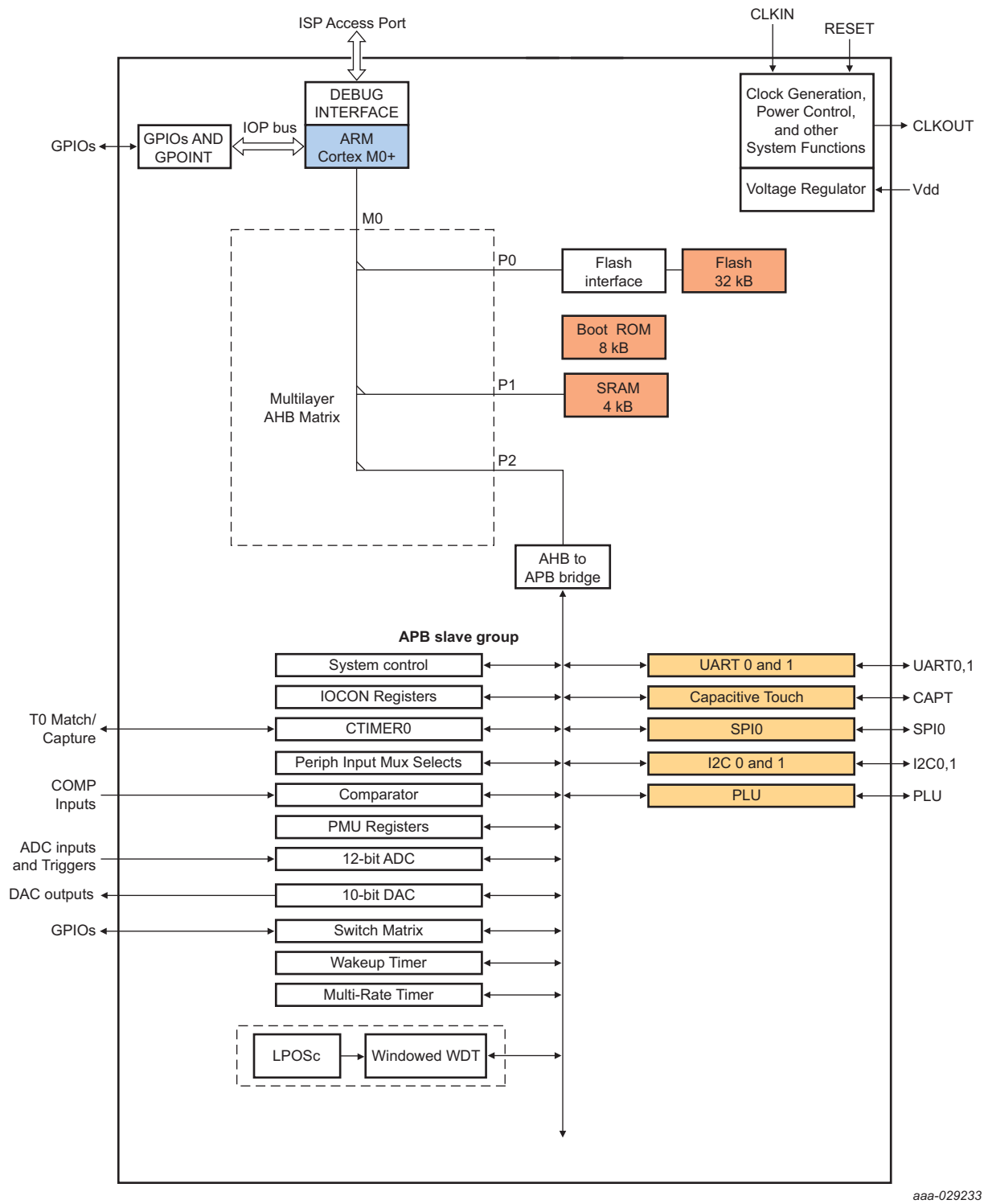



**Fig 1. LPC80x block diagram (aaa-029233)**

## 2.1 How to read this chapter

The memory mapping is identical for all LPC80x parts.

## 2.2 General description

The LPC80x 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 Arm Cortex-M0+ core, such as NVIC, SysTick, and sleep mode control, are located on the private peripheral bus.

The GPIO port and pin interrupt/pattern match registers are accessed by the Arm Cortex-M0+ single-cycle I/O enabled port (IOP).

### 2.2.1 Memory mapping

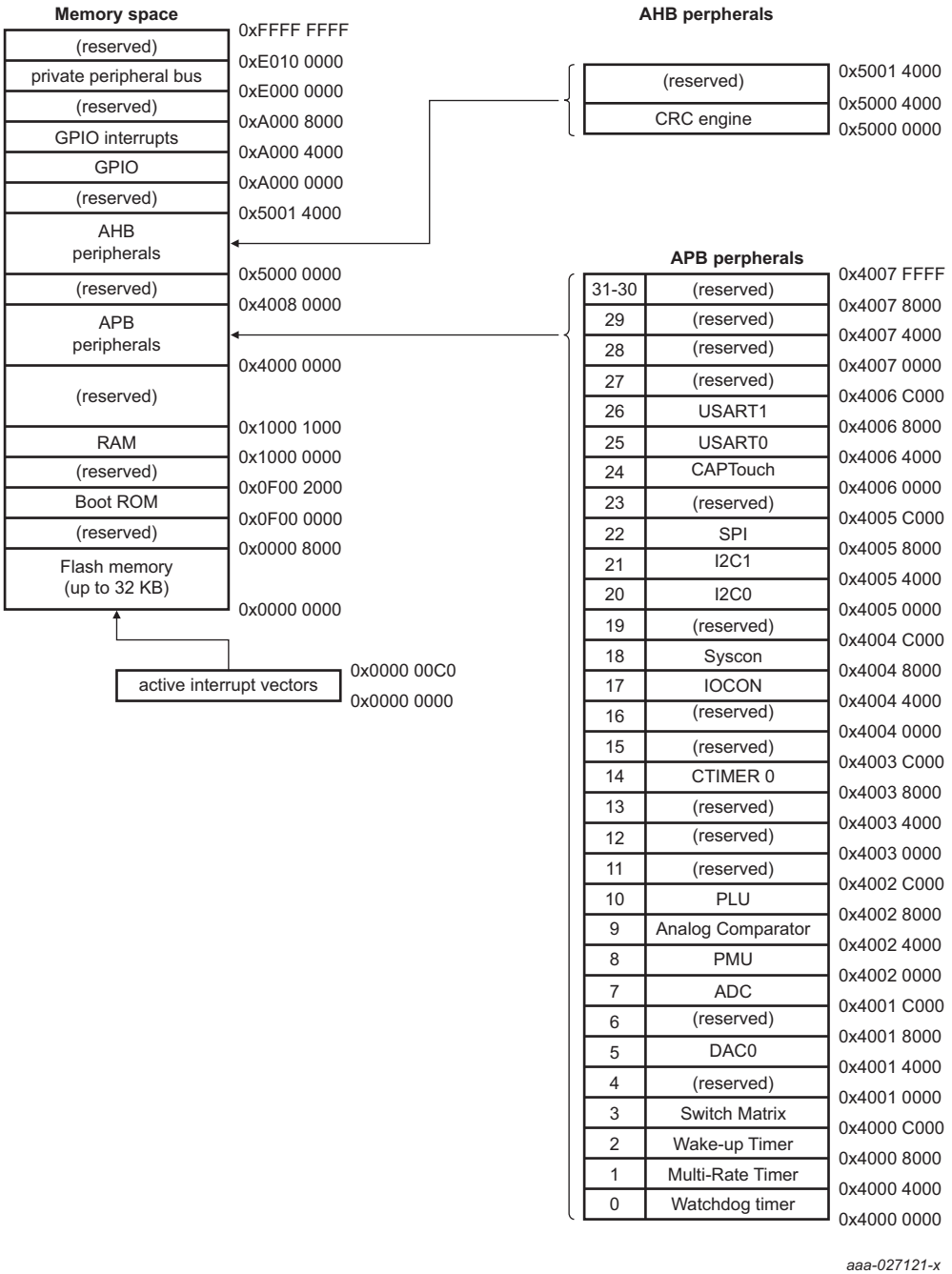



**Fig 2. LPC80x Memory mapping <Final>**

## 3.1 How to read this chapter

The bootloader is identical for all parts.

## 3.2 Features

- 8 KB on-chip boot ROM
- Contains the bootloader with In-System Programming (ISP) facility over the peripheral communication (UART) and the following API:
  - In-Application Programming (IAP) of flash memory.
  - Integer divide routines.
  - FRO frequency select API.

## 3.3 Pin description

When the ISP entry pin (PIO0_12) is pulled LOW on reset, the part enters ISP mode and the ISP command handler starts up.

**Table 3. Pin location in ISP mode**

| ISP mode | Default configuration |
|---|---|
| USART ISP (single supply device) | PIO0_4 is UART0 TX.<br>PIO0_0 is UART0 RX. |
| USART ISP (dual supply device) | PIO0_9 is UART0 TX.<br>PIO0_8 is UART0 RX. |

## 3.4 General description

### 3.4.1 Bootloader

The bootloader executes every time the device is powered on or reset. Based on the chip configuration information, the bootloader controls initial operation after reset, including setting internal voltage regulator, system clock, flash controller, miscellaneous factory trimming value, and then allows programming and reprogramming of internal flash via a set of commands from the USART bus.

During the boot process, a LOW level after reset on the ISP pin is considered as an external hardware request to start the ISP command handler via the USART interface. Otherwise, the bootloader checks if there is valid user code in flash. If the valid user code is not found, the bootloader enters the ISP mode.

**Remark:** The sampling of ISP entry pin can be disabled through programming flash location 0x0000 02FC (see Section 4.3.6 “Code Read Protection (CRP)”).

See Chapter 4 “LPC804 ISP and IAP” for more details.

### 3.4.2 ROM-based APIs

Once the part has booted, the user can access several APIs located in the boot ROM. The ROM API supports:

- Boot loader.
- Flash In-Application Programming (IAP).
- In-System Programming (ISP) through USART.
- On-chip ROM APIs for integer divide.
- FRO frequency select API.

The structure of the boot ROM APIs is shown in Figure 3.

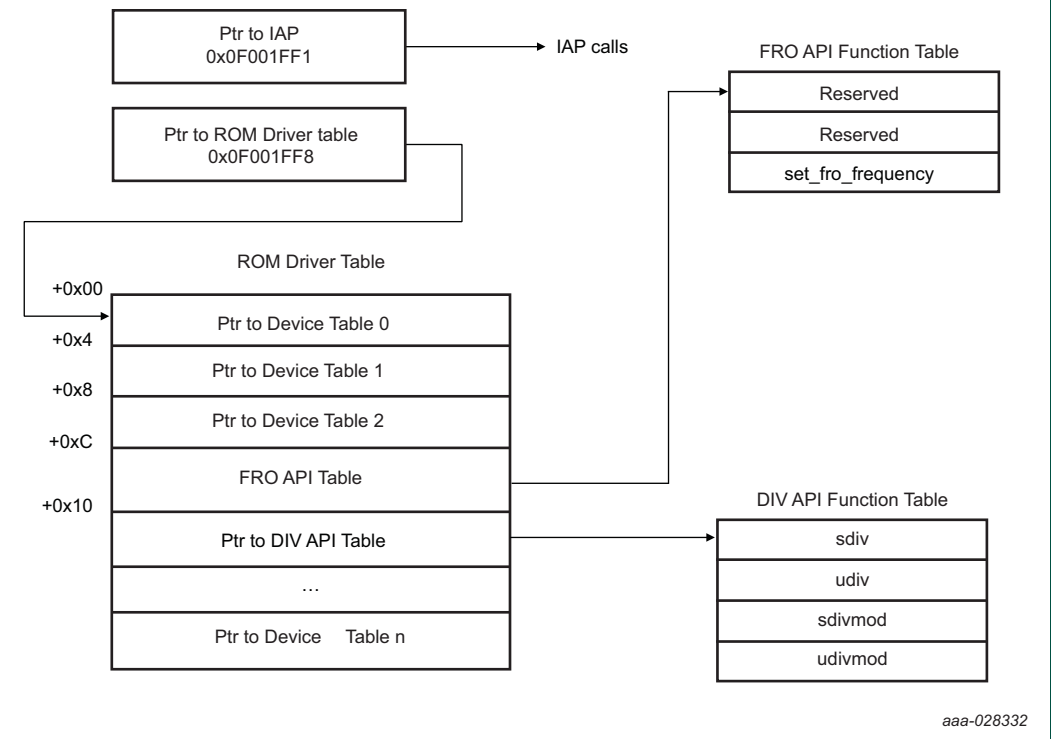



**Fig 3. Boot ROM structure**

The boot ROM structure should be included as follows:

```
typedef struct {
     const uint32_t reserved0; /*!< Reserved */
     const uint32_t reserved1; /*!< Reserved */
     const uint32_t reserved2; /*!< Reserved */
     const uint32_t reserved3; /*!< Reserved */
     const ROM_DIV_API_T *divApiBase; /*!< Divider API function table base address */
} LPC_ROM_API_T;

#define ROM_DRIVER_BASE (0x0F001FF8)
```

**Table 4. API calls**

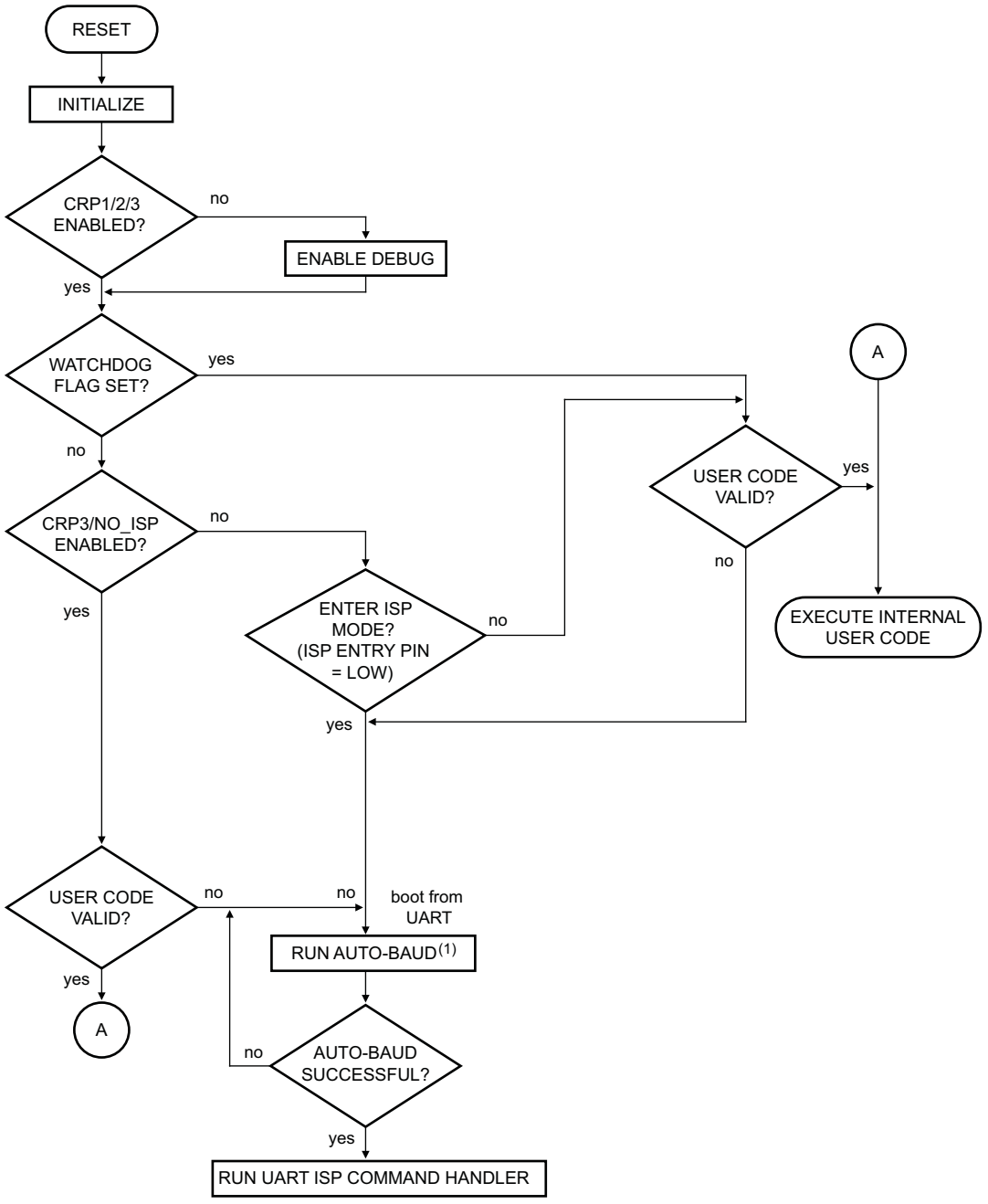
| API | Description | Reference |
|---|---|---|
| Flash IAP | Flash In-Application programming | Table 27 |
| Integer divider API | 32-bit integer divide routines | Table 308 |
| FRO API | Free Running Oscillator configuration Section | Section 7.4 |

## 3.5 Functional description

### 3.5.1 Memory map after any reset

The boot ROM block is 8 KB in size. The boot block is located in the memory region starting from address 0x0F00 0000. The bootloader 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 in Section 4.3.7 “ISP interrupt and SRAM use”. The interrupt vectors residing in the boot block of the on-chip flash memory also become active after reset, that is, the bottom 512 bytes of the boot block are also visible in the memory region starting from the address 0x0000 0000.

### 3.5.2 Boot process




(1) The boot-code is implementing auto-baud in software.

**Fig 4. Boot process flowchart**

## 4.1 How to read this chapter

All LPC804 devices include ROM-based services for programming and reading the flash memory in addition to other functions. In-System Programming works on an unprogrammed or previously programmed device. In-Application Programming allows application software to do the same kinds of operations.

## 4.2 Features

- In-System Programming: In-System programming (ISP) is programming or reprogramming the on-chip flash memory, using the boot loader software and 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 (64 byte) page erase programming.

## 4.3 General description

### 4.3.1 Boot loader

For the boot loader operation and ISP pin, see Chapter 3 “LPC804 Boot Process”.

The boot loader version can be read by ISP/IAP calls (see Section 4.5.13 or Section 4.6.6).

### 4.3.2 Memory map after any reset

The boot ROM is located in the memory region starting from the address 0x0F00 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 4.3.7.

### 4.3.3 Flash content protection mechanism

The LPC804 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 the ECC is transparent to the running application. The ECC content itself is stored in a flash memory not accessible by the user’s code to either read from it or write into it on its own. Six bits of ECC corresponds to every consecutive 32 bit of the user accessible Flash. Consequently, Flash bytes from 0x0000 0000 to 0x0000 0003 are protected by the first 6-bit ECC, Flash bytes from 0x0000 0004 to 0x0000 0007 are protected by the second 6-bit ECC byte, etc.

Whenever the CPU requests a read from user's Flash, both 32 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 4 bytes (or multiples of 4), aligned as described above.

### 4.3.4 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 checksum 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 checksum is not zero indicating valid user code is not found, the bootloader will enter USART ISP mode automatically. When the table entries 0 through 7 are 0, even the checksum is 0, the user code is still invalid.

### 4.3.5 Flash partitions

Some IAP and ISP commands operate on virtual sectors and specify virtual sector numbers. In addition, a page erase command is available. The size of a virtual sector is 1 KB and the size of a page is 64 Byte. One virtual sector contains 16 pages. Page 510 and page 511 in sector 31 are not available for user code because of the boot block.

**Table 5. LPC804 flash configuration**

| Virtual sector number | Virtual sector size [KB] | Page number | Address range | 32 KB flash |
|---|---|---|---|---|
| 0 | 1 | 0 -15 | 0x0000 0000 - 0x0000 03FF | yes |
| 1 | 1 | 16 - 31 | 0x0000 0400 - 0x0000 07FF | yes |
| 2 | 1 | 32 - 47 | 0x0000 0800 - 0x0000 0BFF | yes |
| 3 | 1 | 48 - 63 | 0x0000 0C00 - 0x0000 0FFF | yes |
| 4 | 1 | 64 - 79 | 0x0000 1000 - 0x0000 13FF | yes |
| 5 | 1 | 80 - 95 | 0x0000 1400 - 0x0000 17FF | yes |
| 6 | 1 | 96 - 111 | 0x0000 1800 - 0x0000 1BFF | yes |
| 7 | 1 | 112 - 127 | 0x0000 1C00 - 0x0000 1FFF | yes |
| 8 | 1 | 128 - 143 | 0x0000 2000 - 0x0000 23FF | yes |
| 9 | 1 | 144 - 159 | 0x0000 2400 - 0x0000 27FF | yes |
| 10 | 1 | 160 - 175 | 0x0000 2800 - 0x0000 2BFF | yes |
| 11 | 1 | 176 - 191 | 0x0000 2C00 - 0x0000 2FFF | yes |
| 12 | 1 | 192 - 207 | 0x0000 3000 - 0x0000 33FF | yes |
| 13 | 1 | 208 - 223 | 0x0000 3400 - 0x0000 37FF | yes |
| 14 | 1 | 224 - 239 | 0x0000 3800 - 0x0000 3BFF | yes |
| 15 | 1 | 240 - 255 | 0x0000 3C00 - 0x0000 3FFF | yes |

Table 5. LPC804 flash configuration

| Virtual sector number | Virtual sector size [KB] | Page number | Address range | 32 KB flash |
|---|---|---|---|---|
| 16 | 1 | 256 - 271 | 0x0000 4000 - 0x0000 43FF | yes |
| 17 | 1 | 272 - 287 | 0x0000 4400 - 0x0000 47FF | yes |
| 18 | 1 | 288 - 303 | 0x0000 4800 - 0x0000 4BFF | yes |
| 19 | 1 | 304 - 319 | 0x0000 4C00 - 0x0000 4FFF | yes |
| 20 | 1 | 320 - 335 | 0x0000 5000 - 0x0000 53FF | yes |
| 21 | 1 | 336 - 351 | 0x0000 5400 - 0x0000 57FF | yes |
| 22 | 1 | 352 - 367 | 0x0000 5800 - 0x0000 5BFF | yes |
| 23 | 1 | 368 - 383 | 0x0000 5C00 - 0x0000 5FFF | yes |
| 24 | 1 | 384 - 399 | 0x0000 6000 - 0x0000 63FF | yes |
| 25 | 1 | 400 - 415 | 0x0000 6400 - 0x0000 67FF | yes |
| 26 | 1 | 416 - 431 | 0x0000 6800 - 0x0000 6BFF | yes |
| 27 | 1 | 432 - 447 | 0x0000 6C00 - 0x0000 6FFF | yes |
| 28 | 1 | 448 - 463 | 0x0000 7000 - 0x0000 73FF | yes |
| 29 | 1 | 464 - 479 | 0x0000 7400 - 0x0000 77FF | yes |
| 30 | 1 | 480 - 495 | 0x0000 7800 - 0x0000 7BFF | yes |
| 31 | 1 | 496 - 511 | 0x0000 7C00 - 0x0000 7FFF | yes |

### 4.3.6 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 the flash image at offset 0x0000 02FC. IAP commands are not affected by the code read protection.

Table 1 shows the limitations of the USART ISP commands when CRP (CRP1, CRP2, or CRP3) is enabled.

**Note: Any CRP change becomes effective only after the device has gone through a power cycle.**

Table 6. USART ISP command limitations in CRP modes

| Name | Pattern programmed in 0x0000 02FC | Description |
|---|---|---|
| NO_ISP | 0x4E69 7370 | Prevents sampling of the pins for entering ISP mode. ISP sampling pin is available for other applications. |

**Table 6. USART ISP command limitations in CRP modes**

| Name | Pattern programmed in 0x0000 02FC | Description |
|---|---|---|
| CRP1 | 0x1234 5678 | Access to chip via the SWD pins is disabled. This mode allows partial flash update using the following USART ISP commands and restrictions:<br>• Write to RAM command cannot access RAM below 0x1000 0380.<br>• Copy RAM to flash command cannot write to Sector 0.<br>• Erase command can erase Sector 0 only when all sectors are selected for erase.<br>• Compare command is disabled.<br>• Read Memory command is disabled.<br>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. |
| CRP2 | 0x8765 4321 | Access to chip via the SWD pins is disabled. The following ISP commands are disabled:<br>• Read Memory<br>• Write to RAM<br>• Go<br>• Copy RAM to flash<br>• Compare<br>When CRP2 is enabled the ISP erase command only allows erasure of all user sectors. |
| CRP3 | 0x4321 8765 | 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.<br>This mode effectively disables ISP override using the entry pin. It is up to the application of the user to provide a flash update mechanism using IAP calls or call reinvoke ISP command to enable flash update via USART.<br>**Caution: If CRP3 is selected, no future factory testing can be performed on the device.** |

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.

#### 4.3.6.1 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.

#### 4.3.6.2 ISP entry configuration and detection

The LPC804 USART ISP mode allows programming and reprogramming of the internal FLASH via a set of commands on the USART bus.

### 4.3.7 ISP interrupt and SRAM use

#### 4.3.7.1 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.

#### 4.3.7.2 RAM used by ISP command handlers

The stack of UART ISP commands is located at address 0x1000 03A8. The maximum stack usage is 640 bytes (0x280) and grows downwards.

The ISP flash programming commands use the top 4 bytes of on-chip RAM.

#### 4.3.7.3 RAM used by IAP command handlers

The IAP flash programming commands use the top 4 bytes of on-chip RAM.

## 4.4 USART ISP 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.

### 4.4.1 USART ISP initialization

Once the USART ISP mode is entered, the auto-baud routine needs to synchronize with the host via the serial port (USART).

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. In USART ISP mode, the part is clocked by the FRO 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 4.5 "USART ISP commands".

### 4.4.2 USART ISP command format

"Command Parameter_0 Parameter_1 ... Parameter_n<CR><LF>" "Data" (Data only for Write commands). By default, all commands (all characters) sent from the host are echoed. The echo command can be issued to disable this.

### 4.4.3 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).

### 4.4.4 USART ISP data format

The data stream is in plain binary format.

## 4.5 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 7. USART ISP command summary**

| ISP Command | Usage | Section |
|---|---|---|
| Unlock | U <Unlock Code> | 4.5.1 |
| Set Baud Rate | B <Baud Rate> <stop bit> | 4.5.2 |
| Echo | A <setting> | 4.5.3 |
| Write to RAM | W <start address> <number of bytes> | 4.5.4 |
| Read Memory | R <address> <number of bytes> | 4.5.5 |
| Prepare sectors for write operation | P <start sector number> <end sector number> | 4.5.6 |
| Copy RAM to flash | C <Flash address> <RAM address> <number of bytes> | 4.5.7 |
| Go | G <address> <Mode> | 4.5.8 |
| Erase sector(s) | E <start sector number> <end sector number> | 4.5.9 |
| Erase page(s) | X <start page number> <end page number> | 4.5.10 |
| Blank check sector(s) | I <start sector number> <end sector number> | 4.5.11 |
| Read Part ID | J | 4.5.12 |
| Read Boot code version | K | 4.5.13 |
| Compare | M <address1> <address2> <number of bytes> | 4.5.14 |
| ReadUID | N | 4.5.15 |
| Read CRC checksum | S <address> <number of bytes> | 4.5.16 |

Table 8 lists the supported USART ISP commands for each CRP level.

**Table 8. 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 0x1000 0600 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 |

**Table 8. ISP commands allowed for different CRP levels**

| ISP command | CRP1 | CRP2 | CRP3 (no entry in ISP mode allowed) |
|---|---|---|---|
| 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 |
| 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 |
| Read CRC | no | no | n/a |

### 4.5.1 Unlock

**Table 9. USART ISP Unlock command**

| Command | U |
|---|---|
| Input | Unlock code: $23130_{10}$ |
| Return Code | CMD_SUCCESS \|<br>INVALID_CODE \|<br>PARAM_ERROR |
| 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. |

### 4.5.2 Set Baud Rate

**Table 10. USART ISP Set Baud Rate command**

| Command | B |
|---|---|
| Input | Baud Rate: 9600 \| 19200 \| 38400 \| 57600 \| 115200 \| 230400 \| 460800<br>Stop bit: 1 \| 2<br>**Remark:** In asynchronous USART mode, the USART clock U_PCLK = FRGCLKDIV/ (1+(MULT/DIV)), thus, the U_PCLK is 12MHz. Then, the maximum baud rate = U_PCLK/16 x (BRGVAL+1), where BRGVAL is the value of the BRG register. |
| Return Code | CMD_SUCCESS \|<br>INVALID_BAUD_RATE \|<br>INVALID_STOP_BIT \|<br>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. |

### 4.5.3 Echo

**Table 11. USART ISP Echo command**

| Command | A |
|---|---|
| Input | Setting: ON = 1 \| OFF = 0 |
| Return Code | CMD_SUCCESS \|<br>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. |

### 4.5.4 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 12. USART ISP Write to RAM command**

| Command | W |
|---|---|
| Input | **Start Address:** On-chip RAM address where data bytes are to be written. This address should be a word boundary.<br>**Number of Bytes:** Number of bytes to be written. Count should be a multiple of 4 |
| Return Code | CMD_SUCCESS \|<br>ADDR_ERROR (Address not on word boundary) \|<br>ADDR_NOT_MAPPED \|<br>COUNT_ERROR (Byte count is not multiple of 4) \|<br>PARAM_ERROR \|<br>CODE_READ_PROTECTION_ENABLED |
| Description | This command is used to download data to on-chip RAM. This command is blocked when code read protection levels 2 or 3 are enabled. Writing to addresses below 0x1000 03A8 is disabled for CRP1. |
| Example | "W 268437504 4<CR><LF>" writes 4 bytes of data to address 0x1000 0800. |

### 4.5.5 Read Memory

Reads the plain binary code of the data stream, followed by the CMD_SUCCESS return code.

**Table 13. 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.<br>**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)> \|<br>ADDR_ERROR (Address not on word boundary) \|<br>ADDR_NOT_MAPPED \|<br>COUNT_ERROR (Byte count is not a multiple of 4) \|<br>PARAM_ERROR \|<br>CODE_READ_PROTECTION_ENABLED |
| Description | This command is used to read data from on-chip RAM or flash memory. This command is blocked when code read protection is enabled. |
| Example | "R 268437504 4<CR><LF>" reads 4 bytes of data from address 0x1000 0800. |

### 4.5.6 Prepare sectors for write operation

This command makes flash write/erase operation a two-step process.

**Table 14. USART ISP Prepare sectors for write operation command**

| Command | P |
|---|---|
| Input | **Start Sector Number**<br>**End Sector Number:** Should be greater than or equal to start sector number. |
| Return Code | CMD_SUCCESS \|<br>BUSY \|<br>INVALID_SECTOR \|<br>PARAM_ERROR |
| Description | This command must be executed before executing "Copy RAM to flash" or "Erase Sector(s)", or “Erase Pages” command. Successful execution of the "Copy RAM to flash" or "Erase Sector(s)" or “Erase Pages” 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. |

### 4.5.7 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 64 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 4.3.3.
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 15. USART ISP Copy command**

| Command | C |
|---|---|
| Input | **Flash Address(DST):** Destination flash address where data bytes are to be written. The destination address should be a 64 byte boundary.<br>**RAM Address(SRC):** Source on-chip RAM address from where data bytes are to be read.<br>**Number of Bytes:** Number of bytes to be written. Should be 64 \| 128 \| 256 \| 512 \| 1024 |
| Return Code | CMD_SUCCESS \|<br>SRC_ADDR_ERROR (Address not on word boundary) \|<br>DST_ADDR_ERROR (Address not on correct boundary) \|<br>SRC_ADDR_NOT_MAPPED \|<br>DST_ADDR_NOT_MAPPED \|<br>COUNT_ERROR (Byte count is not 64 \| 128 \| 256 \| 512 \| 1024) \|<br>SECTOR_NOT_PREPARED_FOR WRITE_OPERATION \|<br>BUSY \|<br>CMD_LOCKED \|<br>PARAM_ERROR \|<br>CODE_READ_PROTECTION_ENABLED |
| Description | 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 4.3.3 for the number of bytes that can be written. |
| Example | "C 0 268437504 512<CR><LF>" copies 512 bytes from the RAM address 0x1000 0800 to the flash address 0. |

### 4.5.8 Go

**Table 16. USART ISP Go command**

| Command | G |
|---|---|
| Input | **Address:** Flash or on-chip RAM address from which the code execution is to be started. This address should be on a word boundary.<br>**Mode:** T (Execute program in Thumb Mode) \| |
| Return Code | CMD_SUCCESS \|<br>ADDR_ERROR \|<br>ADDR_NOT_MAPPED \|<br>CMD_LOCKED \|<br>PARAM_ERROR \|<br>CODE_READ_PROTECTION_ENABLED |
| Description | 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. |
| Example | "G 0 T<CR><LF>" branches to address 0x0000 0000 in Thumb mode only. |

### 4.5.9 Erase sectors

**Remark:** The blank (erased) state of a page flash memory is a logic 0.

**Table 17. USART ISP Erase sector command**

| Command | E |
|---|---|
| Input | **Start Sector Number**<br>**End Sector Number:** Should be greater than or equal to start sector number. |
| Return Code | CMD_SUCCESS \|<br>BUSY \|<br>INVALID_SECTOR \|<br>SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION \|<br>CMD_LOCKED \|<br>PARAM_ERROR \|<br>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. |

### 4.5.10 Erase pages

**Remark:** The blank (erased) state of a page flash memory is a logic 0. In sector 31, page 510 and page 511 cannot be erased since boot block resides in these pages. Page 510 and page 511 cannot pass as a parameter.

**Table 18. USART ISP Erase page command**

| Command | X |
|---|---|
| Input | **Start Page Number**<br>**End Page Number:** Should be greater than or equal to start page number. |
| Return Code | CMD_SUCCESS \|<br>BUSY \|<br>INVALID_PAGE \|<br>SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION \|<br>CMD_LOCKED \|<br>PARAM_ERROR \|<br>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. |

### 4.5.11 Blank check sectors

**Remark:** The blank (erased) state of a page flash memory is a logic 0.

**Table 19. USART ISP Blank check sector command**

| Command | I |
|---|---|
| Input | **Start Sector Number:**<br>**End Sector Number:** Should be greater than or equal to start sector number. |
| Return Code | CMD_SUCCESS \|<br>SECTOR_NOT_BLANK (followed by <Offset of the first non blank word location> <Contents of non blank word location>) \|<br>INVALID_SECTOR \|<br>PARAM_ERROR |
| Description | This command is used to blank check one or more sectors of on-chip flash memory. |
| Example | "I 2 3<CR><LF>" blank checks the flash sectors 2 and 3. |

### 4.5.12 Read Part Identification number

**Table 20. USART ISP Read Part Identification command**

| Command | J |
|---|---|
| Input | None. |
| Return Code | CMD_SUCCESS followed by part identification number (see Table 21). |
| Description | This command is used to read the part identification number. |

**Table 21. LPC804 device identification numbers**

| Part number | Part ID |
|---|---|
| LPC804M101JBD64 | 0x00008040 |
| LPC804M101JDH20 | 0x00008041 |
| LPC804M101JDH24 | 0x00008042 |
| LPC804M111JDH24 | 0x00008043 |
| LPC804M101JHI33 | 0x00008044 |

### 4.5.13 Read Boot code version number

**Table 22. 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. |

### 4.5.14 Compare

**Table 23. USART ISP Compare command**

| Command | M |
|---|---|
| Input | **Address1 (DST):** Starting flash or on-chip RAM address of data bytes to be compared. This address should be a word boundary.<br>**Address2 (SRC):** Starting flash or RAM address of data bytes to be compared. This address should be a word boundary.<br>**Number of Bytes:** Number of bytes to be compared; should be a multiple of 4. |
| Return Code | CMD_SUCCESS \| (Source and destination data are equal) \|<br>COMPARE_ERROR \| (Followed by the offset of first mismatch) \|<br>COUNT_ERROR (Byte count is not a multiple of 4) \|<br>ADDR_ERROR \|<br>ADDR_NOT_MAPPED \|<br>PARAM_ERROR \|<br>CODE_READ_PROTECTION_ENABLED |
| Description | This command is used to compare the memory contents at two locations. |
| Example | "M 8192 268437504 4<CR><LF>" compares 4 bytes from the RAM address 0x1000 0800 to the 4 bytes from the flash address 0x2000. |

### 4.5.15 ReadUID

**Table 24. 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. |

### 4.5.16 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: x32 + x26 + x23 + x22 + x16 + x12 + x11 + x10 + x8 + x7 + x5 + x4 + x2 + x + 1

Seed Value: 0xFFFF FFFF

**Table 25. USART ISP Read CRC checksum command**

| Command | S |
|---|---|
| Input | **Address:** The data are read from this address for CRC checksum calculation. This address must be on a word boundary.<br>**Number of Bytes:** Number of bytes to be calculated for the CRC checksum; must be a multiple of 4. |
| Return Code | CMD_SUCCESS followed by data in decimal format \|<br>ADDR_ERROR (address not on word boundary) \|<br>ADDR_NOT_MAPPED \|<br>COUNT_ERROR (byte count is not a multiple of 4) \|<br>PARAM_ERROR \|<br>CODE_READ_PROTECTION_ENABLED |
| Description | This command is used to read the CRC checksum of a block of on-chip RAM or flash memory. This command is blocked when code read protection is enabled. |
| Example | "S 33587200 4<CR><LF>" reads the CRC checksum for 4 bytes of data from address 0x0200 8000.<br>If checksum value is 0xCBF43926, then the host will receive:<br>"3421780262 <CR><LF>" |

### 4.5.17 ISP/IAP Error codes

**Table 26. ISP/IAP Error codes**

| Return Code | Error code | Description |
|---|---|---|
| 0x0 | CMD_SUCCESS | Command is executed successfully. Sent by ISP handler only when command given by the host has been completely and successfully executed. |
| 0x1 | INVALID_COMMAND | Invalid command. |
| 0x2 | SRC_ADDR_ERROR | Source address is not on word boundary. |
| 0x3 | DST_ADDR_ERROR | Destination address is not on a correct boundary. |
| 0x4 | SRC_ADDR_NOT_MAPPED | Source address is not mapped in the memory map. Count value is taken into consideration where applicable. |
| 0x5 | DST_ADDR_NOT_MAPPED | Destination address is not mapped in the memory map. Count value is taken into consideration where applicable. |
| 0x6 | COUNT_ERROR | Byte count is not multiple of 4 or is not a permitted value. |
| 0x7 | INVALID_SECTOR/INVALID_PAGE | Sector/page number is invalid or end sector number is greater than start sector number. |
| 0x8 | SECTOR_NOT_BLANK | Sector is not blank. |
| 0x9 | SECTOR_NOT_PREPARED_ FOR_WRITE_OPERATION | Command to prepare sector for write operation was not executed. |
| 0xA | COMPARE_ERROR | Source and destination data not equal. |
| 0xB | BUSY | Flash programming hardware interface is busy. |
| 0xC | PARAM_ERROR | Insufficient number of parameters or invalid parameter. |
| 0xD | ADDR_ERROR | Address is not on word boundary. |
| 0xE | ADDR_NOT_MAPPED | Address is not mapped in the memory map. Count value is taken into consideration where applicable. |
| 0xF | CMD_LOCKED | Command is locked. |
| 0x10 | INVALID_CODE | Unlock code is invalid. |
| 0x11 | INVALID_BAUD_RATE | Invalid baud rate setting. |
| 0x12 | INVALID_STOP_BIT | Invalid stop bit setting. |
| 0x13 | CODE_READ_ PROTECTION_ENABLED | Code read protection enabled. |
| 0x14 | - | Reserved. |
| 0x15 | USER_CODE_CHECKSUM | User code checksum is invalid. |
| 0x16 | - | Reserved. |
| 0x17 | EFRO_NO_POWER | FRO not turned on in the PDRUNCFG register. |
| 0x18 | FLASH_NO_POWER | Flash not turned on in the PDRUNCFG register. |
| 0x19 | - | Reserved. |
| 0x20 | FLASH_ERASE_PROGRAM | Flash erase/program not successful. |
| 0x21 | INVALID_PAGE | Page is invalid. |
| 0x1A | - | Reserved. |
| 0x1B | FLASH_NO_CLOCK | Flash clock disabled in the AHBCLKCTRL register. |
| 0x1C | REINVOKE_ISP_CONFIG | Reinvoke ISP not successful. |
| 0x1D | NO_VALID_IMAGE | Invalid image |

## 4.6 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 5.

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 location 0x0F001FF0 and it is thumb code, therefore called as 0x0F001FF1 by the Cortex-M0+ to ensure 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-M0+.

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 *) 0x...;
unsigned int * 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 *(volatile unsigned int *)(0x0F001FF1)
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.

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 be use this space if IAP flash programming is permitted in the application.

**Table 27. IAP Command Summary**

| IAP Command | Command code | Section |
|---|---|---|
| Prepare sector(s) for write operation | 50 (decimal) | 4.6.1 |
| Copy RAM to flash | 51 (decimal) | 4.6.2 |
| Erase sector(s) | 52 (decimal) | 4.6.3 |
| Blank check sector(s) | 53 (decimal) | 4.6.4 |
| Read Part ID | 54 (decimal) | 4.6.5 |
| Read Boot code version | 55 (decimal) | 4.6.6 |
| Compare | 56 (decimal) | 4.6.7 |
| Reinvoke ISP | 57 (decimal) | 4.6.8 |
| Read UID | 58 (decimal) | 4.6.9 |
| Erase page(s) | 59 (decimal) | 4.6.10 |

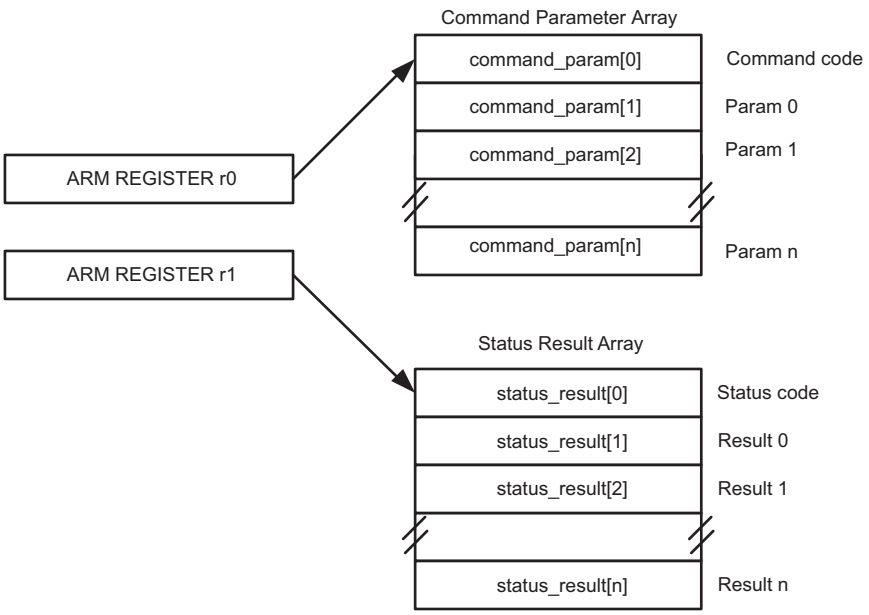



**Fig 5. IAP parameter passing**

### 4.6.1 Prepare sector(s) for write operation

This command makes flash write/erase operation a two step process.

**Table 28. IAP Prepare sector(s) for write operation command**

| Command | Prepare sector(s) for write operation |
|---|---|
| Input | **Command code: 50 (decimal)**<br>**Param0:** Start Sector Number<br>**Param1:** End Sector Number (should be greater than or equal to start sector number). |
| Status code | CMD_SUCCESS \|<br>BUSY \|<br>INVALID_SECTOR |
| Result | None |
| Description | This command must be executed before executing "Copy RAM to flash" or "Erase Sector(s)" or “Erase page(s)” command. Successful execution of the "Copy RAM to flash" or "Erase Sector(s)" or “Erase page(s)” command causes relevant sectors to be protected again. To prepare a single sector use the same "Start" and "End" sector numbers. |

### 4.6.2 Copy RAM to flash

See Section 4.5.7 for limitations on the write-to-flash process.

**Table 29. IAP Copy RAM to flash command**

| Command | Copy RAM to flash |
|---|---|
| Input | **Command code: 51 (decimal)**<br>**Param0(DST):** Destination flash address where data bytes are to be written. This address should be a 64 byte boundary.<br>**Param1(SRC):** Source RAM address from which data bytes are to be read. This address should be a word boundary.<br>**Param2:** Number of bytes to be written. Should be 64 \| 128 \| 256 \| 512 \| 1024.<br>**Param3:** NULL. |
| Status code | CMD_SUCCESS \|<br>SRC_ADDR_ERROR (Address not a word boundary) \|<br>DST_ADDR_ERROR (Address not on correct boundary) \|<br>SRC_ADDR_NOT_MAPPED \|<br>DST_ADDR_NOT_MAPPED \|<br>COUNT_ERROR (Byte count is not 64 \| 128 \| 256 \| 512 \| 1024) \|<br>SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION \|<br>BUSY |
| Result | None |
| Description | 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. Also see Section 4.3.3 for the number of bytes that can be written.<br>**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. Param3 is overwritten by the fixed value of 12 MHz, which is the FRO reference clock used by the flash controller. |

### 4.6.3 Erase Sector(s)

**Remark:** The blank (erased) state of a page flash memory is a logic 0.

**Table 30. IAP Erase Sector(s) command**

| Command | Erase Sector(s) |
|---|---|
| Input | **Command code: 52 (decimal)**<br>**Param0:** Start Sector Number<br>**Param1:** End Sector Number (should be greater than or equal to start sector number).<br>**Param2:** NULL. |
| Status code | CMD_SUCCESS \|<br>BUSY \|<br>SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION \|<br>INVALID_SECTOR |
| Result | None |
| Description | This command is used to erase a sector or multiple sectors of on-chip flash memory. To erase a single sector use the same "Start" and "End" sector numbers.<br>**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. Param2 is overwritten by the fixed value of 12 MHz, which is the FRO reference clock used by the flash controller. |

### 4.6.4 Blank check sector(s)

**Remark:** The blank (erased) state of a page flash memory is a logic 0.

**Table 31. IAP Blank check sector(s) command**

| Command | Blank check sector(s) |
|---|---|
| Input | **Command code: 53 (decimal)**<br>**Param0:** Start Sector Number<br>**Param1:** End Sector Number (should be greater than or equal to start sector number). |
| Status code | CMD_SUCCESS \|<br>BUSY \|<br>SECTOR_NOT_BLANK \|<br>INVALID_SECTOR |
| Result | **Result0:** Offset of the first non blank word location if the status code is SECTOR_NOT_BLANK.<br>**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. |

### 4.6.5 Read Part Identification number

**Table 32. IAP Read Part Identification command**

| Command | Read part identification number |
|---|---|
| Input | **Command code: 54 (decimal)**<br>**Parameters:** None |
| Status code | CMD_SUCCESS |
| Result | **Result0:** Part Identification Number. |
| Description | This command is used to read the part identification number. |

### 4.6.6 Read Boot code version number

**Table 33. IAP Read Boot Code version number command**

| Command | Read boot code version number |
|---|---|
| Input | **Command code: 55 (decimal)**<br>**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. |

### 4.6.7 Compare <address1> <address2> <no of bytes>

**Table 34. IAP Compare command**

| Command | Compare |
|---|---|
| Input | **Command code: 56 (decimal)**<br>**Param0(DST):** Starting flash or RAM address of data bytes to be compared; should be a word boundary.<br>**Param1(SRC):** Starting flash or RAM address of data bytes to be compared; should be a word boundary.<br>**Param2:** Number of bytes to be compared; should be a multiple of 4. |
| Status code | CMD_SUCCESS \|<br>COMPARE_ERROR \|<br>COUNT_ERROR (Byte count is not a multiple of 4) \|<br>ADDR_ERROR \|<br>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. |

### 4.6.8 Reinvoke ISP

**Table 35. Reinvoke ISP**

| Command | Compare |
|---|---|
| Input | **Command code: 57 (decimal)**<br>**Param0(mode):** ISP interface selection<br>1 - USART ISP |
| Status code | ERR_ISP_REINVOKE_ISP_CONFIG |
| Result | None. |
| Description | This command is used to invoke the ISP. If the ISP is invoked, then the CPU clock is switched to FRO.<br>This command is used to invoke the boot loader in ISP mode. It maps boot vectors and configures the peripherals for ISP.<br>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.<br>If using USART ISP mode, enable the clocks to the default before calling this command. |

### 4.6.9 ReadUID

**Table 36. 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).<br>**Result1:** The second 32-bit word.<br>**Result2:** The third 32-bit word.<br>**Result3:** The fourth 32-bit word. |
| Description | This command is used to read the unique ID. |

### 4.6.10 Erase page

**Remark:** The blank (erased) state of a page flash memory is a logic 0. In sector 31, page 510 and page 511 cannot be erased since boot block resides in these pages. Page 510 and page 511 cannot pass as a parameter.

**Table 37. IAP Erase page command**

| Command | Erase page |
|---|---|
| Input | **Command code: 59 (decimal)**<br>**Param0:** Start page number.<br>**Param1:** End page number (should be greater than or equal to start page)<br>**Param2:** NULL |
| Status code | CMD_SUCCESS \|<br>BUSY \|<br>SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION \|<br>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.<br>**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. Param2 is overwritten by the fixed value of 12 MHz, which is the FRO reference clock used by the flash controller. |

### 4.6.11 IAP Error Codes

See Table 26 “ISP/IAP Error codes”.

## 5.1 How to read this chapter

The NVIC is identical on all LPC80x parts.

## 5.2 Features

- Nested Vectored Interrupt Controller that is an integral part of the Arm Cortex-M0+.
- Tightly coupled interrupt controller provides low interrupt latency.
- Controls system exceptions and peripheral interrupts.
- The NVIC supports 32 vectored interrupts.
- Four programmable interrupt priority levels with hardware priority level masking.
- Software interrupt generation using the Arm exceptions SVCall and PendSV (see Ref. 3).
- Support for NMI.
- Arm Cortex M0+ Vector table offset register VTOR implemented.

## 5.3 General description

The Nested Vectored Interrupt Controller (NVIC) is an integral part of the Cortex-M0+. The tight coupling to the CPU allows for low interrupt latency and efficient processing of late arriving interrupts.

### 5.3.1 Interrupt sources

Table 38 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. Interrupts with the same priority level are serviced in the order of their interrupt number.

See Ref. 3 for a detailed description of the NVIC and the NVIC register description.

**Table 38. Connection of interrupt sources to the NVIC**

| Interrupt number | Name | Description | Flags |
|---|---|---|---|
| 0 | SPI0_IRQ | SPI0 interrupt | See Table 193 “SPI Interrupt Enable read and Set register (INTENSET, addresses 0x4005 800C (SPI) bit description”. |
| 1 | - | Reserved | - |
| 2 | DAC0_IRQ | DAC0 interrupt | - |
| 3 | UART0_IRQ | USART0 interrupt | See Table 179 “USART Interrupt Enable read and set register (INTENSET, address 0x4006 400C (USART0), 0x4006 800C (USART1)) bit description” |
| 4 | UART1_IRQ | USART1 interrupt | Same as UART0_IRQ |

**Table 38. Connection of interrupt sources to the NVIC**

| Interrupt number | Name | Description | Flags |
|---|---|---|---|
| 5 | - | Reserved | - |
| 6 | - | Reserved | - |
| 7 | I2C1_IRQ | I2C1 interrupt | See Table 209 “Interrupt Enable Clear register (INTENCLR, address 0x4005 000C (I2C0), 0x4005 400C (I2C1)) bit description”. |
| 8 | I2C0_IRQ | I2C0 interrupt | See Table 209 “Interrupt Enable Clear register (INTENCLR, address 0x4005 000C (I2C0), 0x4005 400C (I2C1)) bit description”. |
| 9 | - | Reserved | - |
| 10 | MRT_IRQ | Multi-rate timer interrupt | Global MRT interrupt.<br>GFLAG0<br>GFLAG1<br>GFLAG2<br>GFLAG3 |
| 11 | CMP_IRQ or CAPT_IRQ | Analog comparator interrupt or Capacitive Touch interrupt | COMPEDGE - rising, falling, or both edges can set the bit. Capacitive Touch interrupt. |
| 12 | WDT_IRQ | Windowed watchdog timer interrupt | WARNINT - watchdog warning interrupt |
| 13 | BOD_IRQ | BOD interrupts | BODINTVAL - BOD interrupt level |
| 14 | - | Reserved | - |
| 15 | WKT_IRQ | Self-wake-up timer interrupt | ALARMFLAG |
| 16 | ADC_SEQA_IRQ | ADC sequence A completion | - |
| 17 | ADC_SEQB_IRQ | ADC sequence B completion | - |
| 18 | ADC_THCMP_IRQ | ADC threshold compare | - |
| 19 | ADC_OVR_IRQ | ADC overrun | - |
| 20 | - | Reserved | - |
| 21 | - | Reserved | - |
| 22 | - | Reserved | - |
| 23 | CT32B0_IRQ | Timer interrupt | - |
| 24 | PININT0_IRQ | Pin interrupt 0 or pattern match engine slice 0 interrupt | PSTAT - pin interrupt status |
| 25 | PININT1_IRQ | Pin interrupt 1 or pattern match engine slice 1 interrupt | PSTAT - pin interrupt status |
| 26 | PININT2_IRQ | Pin interrupt 2 or pattern match engine slice 2 interrupt | PSTAT - pin interrupt status |
| 27 | PININT3_IRQ | Pin interrupt 3 or pattern match engine slice 3 interrupt | PSTAT - pin interrupt status |

**Table 38. Connection of interrupt sources to the NVIC**

| Interrupt number | Name | Description | Flags |
|---|---|---|---|
| 28 | PININT4_IRQ | Pin interrupt 4 or pattern match engine slice 4 interrupt | PSTAT - pin interrupt status |
| 29 | PININT5_IRQ | Pin interrupt 5 or pattern match engine slice 5 interrupt. | PSTAT - pin interrupt status |
| 30 | PININT6_IRQ | Pin interrupt 6 or pattern match engine slice 6 interrupt. | PSTAT - pin interrupt status |
| 31 | PININT7_IRQ | Pin interrupt 7 or pattern match engine slice 7 interrupt. | PSTAT - pin interrupt status |

### 5.3.2 Non-Maskable Interrupt (NMI)

The part supports the NMI, which can be triggered by an peripheral interrupt or triggered by software. The NMI has the highest priority exception other than the reset.

You can set up any peripheral interrupt listed in Table 38 as NMI using the NMISRC register in the SYSCON block (Table 78). To avoid using the same peripheral interrupt as NMI exception and normal interrupt, disable the interrupt in the NVIC when you configure it as NMI.

### 5.3.3 Vector table offset

The vector table contains the reset value of the stack pointer and the start addresses, also called exception vectors, for all exception handlers. On system reset, the vector table is located at address 0x0000 0000. Software can write to the VTOR register in the NVIC to relocate the vector table start address to a different memory location. For a description of the VTOR register, see the Arm Cortex-M0+ documentation (Ref. 3).

## 5.4 Register description

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

**Table 39. Register overview: NVIC (base address 0xE000 E000)**

| Name | Access | Address offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| ISER0 | RW | 0x100 | Interrupt Set Enable Register 0. This register allows enabling interrupts and reading back the interrupt enables for specific peripheral functions. | 0 | Table 40 |
| - | - | 0x104 | Reserved. | - | - |
| ICER0 | RW | 0x180 | Interrupt Clear Enable Register 0. This register allows disabling interrupts and reading back the interrupt enables for specific peripheral functions. | 0 | Table 41 |
| - | - | 0x184 | Reserved. | 0 | - |
| ISPR0 | RW | 0x200 | Interrupt Set Pending Register 0. This register allows changing the interrupt state to pending and reading back the interrupt pending state for specific peripheral functions. | 0 | Table 42 |
| - | - | 0x204 | Reserved. | 0 | - |
| ICPR0 | RW | 0x280 | Interrupt Clear Pending Register 0. This register allows changing the interrupt state to not pending and reading back the interrupt pending state for specific peripheral functions. | 0 | Table 43 |
| - | - | 0x284 | Reserved. | 0 | - |
| - | - | 0x300 | Reserved. | 0 | - |
| - | - | 0x304 | Reserved. | 0 | - |
| IPR0 | RW | 0x400 | Interrupt Priority Registers 0. This register allows assigning a priority to each interrupt. This register contains the 2-bit priority fields for interrupts 0 to 3. | 0 | Table 44 |
| IPR1 | RW | 0x404 | Interrupt Priority Registers 1 This register allows assigning a priority to each interrupt. This register contains the 2-bit priority fields for interrupts 4 to 7. | 0 | Table 45 |
| IPR2 | RW | 0x408 | Interrupt Priority Registers 2. This register allows assigning a priority to each interrupt. This register contains the 2-bit priority fields for interrupts 8 to 11. | 0 | Table 46 |
| IPR3 | RW | 0x40C | Interrupt Priority Registers 3. This register allows assigning a priority to each interrupt. This register contains the 2-bit priority fields for interrupts 12 to 15. | 0 | Table 47 |
| IPR4 | RW | 0x410 | Interrupt Priority Registers 4. This register allows assigning a priority to each interrupt. This register contains the 2-bit priority fields for interrupts 16 to 19. | 0 | Table 48 |
| IPR5 | RW | 0x414 | Interrupt Priority Registers 5. This register allows assigning a priority to each interrupt. This register contains the 2-bit priority fields for interrupts 20 to 23. | 0 | Table 49 |
| IPR6 | RW | 0x418 | Interrupt Priority Registers 6. This register allows assigning a priority to each interrupt. This register contains the 2-bit priority fields for interrupts 24 to 27. | 0 | Table 50 |
| IPR7 | RW | 0x41C | Interrupt Priority Registers 7. This register allows assigning a priority to each interrupt. This register contains the 2-bit priority fields for interrupts 28 to 31. | 0 | Table 51 |

### 5.4.1 Interrupt Set Enable Register 0 register

The ISER0 register allows to enable peripheral interrupts or to read the enabled state of those interrupts. Disable interrupts through the ICER0 (Section 5.4.2).

The bit description is as follows for all bits in this register:

**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.

**Table 40. Interrupt Set Enable Register 0 register (ISER0, address 0xE000 E100) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 0 | ISE_SPI0 | Interrupt enable. | 0 |
| 1 | - | Reserved. | 0 |
| 2 | DAC0 | Interrupt enable. | 0 |
| 3 | ISE_UART0 | Interrupt enable. | 0 |
| 4 | ISE_UART1 | Interrupt enable. | 0 |
| 5 | - | Reserved. | 0 |
| 6 | - | Reserved. | 0 |
| 7 | ISE_I2C1 | Interrupt enable. | 0 |
| 8 | ISE_I2C0 | Interrupt enable. | 0 |
| 9 | - | Reserved. | 0 |
| 10 | ISE_MRT | Interrupt enable. | 0 |
| 11 | ISE_CMP or ISE_CAPT | Interrupt enable for both comparator and Capacitive Touch. | 0 |
| 12 | ISE_WDT | Interrupt enable. | 0 |
| 13 | ISE_BOD | Interrupt enable. | 0 |
| 14 | ISE_FLASH | Interrupt enable. | 0 |
| 15 | ISE_WKT | Interrupt enable. | 0 |
| 16 | ISE_ADC_SEQA | Interrupt enable. | 0 |
| 17 | ISE_ADC_SEQB | Interrupt enable. | 0 |
| 18 | ISE_ADC_THCMP | Interrupt enable. | 0 |
| 19 | ISE_ADC_OVR | Interrupt enable. | 0 |
| 20 | - | Reserved. | 0 |
| 21 | - | Reserved. | 0 |
| 22 | - | Reserved. | 0 |
| 23 | ISE_CT32b0 | Interrupt enable. | 0 |
| 24 | ISE_PININT0 | Interrupt enable. | 0 |
| 25 | ISE_PININT1 | Interrupt enable. | 0 |
| 26 | ISE_PININT2 | Interrupt enable. | 0 |
| 27 | ISE_PININT3 | Interrupt enable. | 0 |
| 28 | ISE_PININT4 | Interrupt enable. | 0 |
| 29 | ISE_PININT5 | Interrupt enable. | 0 |
| 30 | ISE_PININT6 | Interrupt enable. | 0 |
| 31 | ISE_PININT7 | Interrupt enable. | 0 |

### 5.4.2 Interrupt clear enable register 0

The ICER0 register allows disabling the peripheral interrupts, or for reading the enabled state of those interrupts. Enable interrupts through the ISER0 registers (Section 5.4.1).

The bit description is as follows for all bits in this register:

**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.

**Table 41. Interrupt clear enable register 0 (ICER0, address 0xE000 E180)**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 0 | ICE_SPI0 | Interrupt disable. | 0 |
| 1 | - | Reserved | 0 |
| 2 | ICE_DAC0 | Interrupt disable. | 0 |
| 3 | ICE_UART0 | Interrupt disable. | 0 |
| 4 | ICE_UART1 | Interrupt disable. | 0 |
| 5 | - | Reserved | 0 |
| 6 | - | Reserved | 0 |
| 7 | ICE_I2C1 | Interrupt disable. | 0 |
| 8 | ICE_I2C0 | Interrupt disable. | 0 |
| 9 | - | Reserved | 0 |
| 10 | ICE_MRT | Interrupt disable. | 0 |
| 11 | ICE_CMP or ICE_CAPT | Interrupt disable for both comparator and Capacitive Touch. | ISE_CMP or ISE_CAPT |
| 12 | ICE_WDT | Interrupt disable. | 0 |
| 13 | ICE_BOD | Interrupt disable. | 0 |
| 14 | ICE_FLASH | Interrupt disable. | 0 |
| 15 | ICE_WKT | Interrupt disable. | 0 |
| 16 | ICE_ADC_SEQA | Interrupt disable. | 0 |
| 17 | ICE_ADC_SEQB | Interrupt disable. | 0 |
| 18 | ICE_ADC_THCMP | Interrupt disable. | 0 |
| 19 | ICE_ADC_OVR | Interrupt disable. | 0 |
| 20 | - | Reserved | 0 |
| 21 | - | Reserved | 0 |
| 22 | - | Reserved | 0 |
| 23 | ICE_CT32B0 | Interrupt disable. | 0 |
| 24 | ICE_PININT0 | Interrupt disable. | 0 |
| 25 | ICE_PININT1 | Interrupt disable. | 0 |
| 26 | ICE_PININT2 | Interrupt disable. | 0 |
| 27 | ICE_PININT3 | Interrupt disable. | 0 |
| 28 | ICE_PININT4 | Interrupt disable. | 0 |
| 29 | ICE_PININT5 | Interrupt disable. | 0 |
| 30 | ICE_PININT6 | Interrupt disable. | 0 |
| 31 | ICE_PININT7 | Interrupt disable. | 0 |

### 5.4.3 Interrupt Set Pending Register 0 register

The ISPR0 register allows setting the pending state of the peripheral interrupts, or for reading the pending state of those interrupts. Clear the pending state of interrupts through the ICPR0 registers (Section 5.4.4).

The bit description is as follows for all bits in this register:

**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.

**Table 42. Interrupt set pending register 0 register (ISPR0, address 0xE000 E200) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 0 | ISP_SPI0 | Interrupt pending set. | 0 |
| 1 | - | Reserved | 0 |
| 2 | ISP_DAC0 | Interrupt pending set. | 0 |
| 3 | ISP_UART0 | Interrupt pending set. | 0 |
| 4 | ISP_UART1 | Interrupt pending set. | 0 |
| 5 | - | Reserved. | 0 |
| 6 | - | Reserved | 0 |
| 7 | ISP_I2C1 | Interrupt pending set. | 0 |
| 8 | ISP_I2C0 | Interrupt pending set. | 0 |
| 9 | - | Reserved. | 0 |
| 10 | ISP_MRT | Interrupt pending set. | 0 |
| 11 | ISP_CMP or ISP_CAPT | Interrupt pending set for both comparator and Capacitive Touch. | 0 |
| 12 | ISP_WDT | Interrupt pending set. | 0 |
| 13 | ISP_BOD | Interrupt pending set. | 0 |
| 14 | ISP_FLASH | Interrupt pending set. | 0 |
| 15 | ISP_WKT | Interrupt pending set. | 0 |
| 16 | ISP_ADC_SEQA | Interrupt pending set. | 0 |
| 17 | ISP_ADC_SEQB | Interrupt pending set. | 0 |
| 18 | ISP_ADC_THCMP | Interrupt pending set. | 0 |
| 19 | ISP_ADC_OVR | Interrupt pending set. | 0 |
| 20 | - | Reserved. | 0 |
| 21 | - | Reserved. | 0 |
| 22 | - | Reserved. | 0 |
| 23 | ISP_CT32B0 | Interrupt pending set. | 0 |
| 24 | ISP_PININT0 | Interrupt pending set. | 0 |
| 25 | ISP_PININT1 | Interrupt pending set. | 0 |
| 26 | ISP_PININT2 | Interrupt pending set. | 0 |
| 27 | ISP_PININT3 | Interrupt pending set. | 0 |
| 28 | ISP_PININT4 | Interrupt pending set. | 0 |

**Table 42. Interrupt set pending register 0 register (ISPR0, address 0xE000 E200) bit description** *...continued*

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 29 | ISP_PININT5 | Interrupt pending set. | 0 |
| 30 | ISP_PININT6 | Interrupt pending set. | 0 |
| 31 | ISP_PININT7 | Interrupt pending set. | 0 |

### 5.4.4 Interrupt Clear Pending Register 0 register

The ICPR0 register allows clearing the pending state of the peripheral interrupts, or for reading the pending state of those interrupts. Set the pending state of interrupts through the ISPR0 register (Section 5.4.3).

The bit description is as follows for all bits in this register:

**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.

**Table 43. Interrupt clear pending register 0 register (ICPR0, address 0xE000 E280) bit description**

| Bit | Symbol | Function | Reset value |
|---|---|---|---|
| 0 | ICP_SPI0 | Interrupt pending clear. | 0 |
| 1 | - | Reserved. | 0 |
| 2 | ICP_DAC0 | Interrupt pending clear. | 0 |
| 3 | ICP_UART0 | Interrupt pending clear. | 0 |
| 4 | ICP_UART1 | Interrupt pending clear. | 0 |
| 5 | - | Reserved. | 0 |
| 6 | - | Reserved. | 0 |
| 7 | ICP_I2C1 | Interrupt pending clear. | 0 |
| 8 | ICP_I2C0 | Interrupt pending clear. | 0 |
| 9 | - | Reserved. | 0 |
| 10 | ICP_MRT | Interrupt pending clear. | 0 |
| 11 | ICP_CMP or ICP_CAPT | Interrupt pending clear for both comparator and Capacitive Touch. | 0 |
| 12 | ICP_WDT | Interrupt pending clear. | 0 |
| 13 | ICP_BOD | Interrupt pending clear. | 0 |
| 14 | ICP_FLASH | Interrupt pending clear. | 0 |
| 15 | ICP_WKT | Interrupt pending clear. | 0 |
| 16 | ICP_ADC_SEQA | Interrupt pending clear. | 0 |
| 17 | ICP_ADC_SEQB | Interrupt pending clear. | 0 |
| 18 | ICP_ADC_THCMP | Interrupt pending clear. | 0 |
| 19 | ICP_ADC_OVR | Interrupt pending clear. | 0 |
| 20 | - | Reserved. | 0 |
| 21 | - | Reserved. | 0 |
| 22 | - | Reserved. | 0 |
| 23 | ICP_CT32B0 | Interrupt pending clear. | 0 |
| 24 | ICP_PININT0 | Interrupt pending clear. | 0 |
| 25 | ICP_PININT1 | Interrupt pending clear. | 0 |

**Table 43. Interrupt clear pending register 0 register (ICPR0, address 0xE000 E280) bit description** *...continued*

| Bit | Symbol | Function | Reset value |
|---|---|---|---|
| 26 | ICP_PININT2 | Interrupt pending clear. | 0 |
| 27 | ICP_PININT3 | Interrupt pending clear. | 0 |
| 28 | ICP_PININT4 | Interrupt pending clear. | 0 |
| 29 | ICP_PININT5 | Interrupt pending clear. | 0 |
| 30 | ICP_PININT6 | Interrupt pending clear. | 0 |
| 31 | ICP_PININT7 | Interrupt pending clear. | 0 |

### 5.4.5 Interrupt Priority Register 0

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

**Table 44. Interrupt Priority Register 0 (IPR0, address 0xE000 E400) bit description**

| Bit | Symbol | Description |
|---|---|---|
| 5:0 | - | These bits ignore writes, and read as 0. |
| 7:6 | IP_SPI0 | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |
| 13:8 | - | These bits ignore writes, and read as 0. |
| 15:14 | - | Reserved. |
| 21:16 | - | These bits ignore writes, and read as 0. |
| 23:22 | IP_DAC0 | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |
| 29:24 | - | Reserved. |
| 31:30 | IP_UART0 | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |

### 5.4.6 Interrupt Priority Register 1

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

**Table 45. Interrupt Priority Register 1 (IPR1, address 0xE000 E404) bit description**

| Bit | Symbol | Description |
|---|---|---|
| 5:0 | - | These bits ignore writes, and read as 0. |
| 7:6 | IP_UART1 | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |
| 13:8 | - | These bits ignore writes, and read as 0. |
| 15:14 | - | Reserved. |
| 21:16 | - | These bits ignore writes, and read as 0. |
| 23:22 | - | Reserved. |
| 29:24 | - | These bits ignore writes, and read as 0. |
| 31:30 | IP_I2C1 | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |

### 5.4.7 Interrupt Priority Register 2

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

**Table 46. Interrupt Priority Register 2 (IPR2, address 0xE000 E408) bit description**

| Bit | Symbol | Description |
|---|---|---|
| 5:0 | - | These bits ignore writes, and read as 0. |
| 7:6 | IP_I2C0 | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |
| 13:8 | - | These bits ignore writes, and read as 0. |
| 15:14 | - | Reserved. |
| 21:16 | - | These bits ignore writes, and read as 0. |
| 23:22 | IP_MRT | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |
| 29:24 | - | These bits ignore writes, and read as 0. |
| 31:30 | IP_CMP or IP_CAPT | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |

### 5.4.8 Interrupt Priority Register 3

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

**Table 47. Interrupt Priority Register 3 (IPR3, address 0xE000 E40C) bit description**

| Bit | Symbol | Description |
|---|---|---|
| 5:0 | - | These bits ignore writes, and read as 0. |
| 7:6 | IP_WDT | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |
| 13:8 | - | These bits ignore writes, and read as 0. |
| 15:14 | IP_BOD | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |
| 21:16 | - | These bits ignore writes, and read as 0. |
| 23:22 | IP_FLASH | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |
| 29:24 | - | These bits ignore writes, and read as 0. |
| 31:30 | IP_WKT | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |

### 5.4.9 Interrupt Priority Register 4

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

**Table 48. Interrupt Priority Register 4 (IPR4, address 0xE000 E410) bit description**

| Bit | Symbol | Description |
|---|---|---|
| 5:0 | - | These bits ignore writes, and read as 0. |
| 7:6 | IP_ADC_SEQA | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |
| 13:8 | - | These bits ignore writes, and read as 0. |
| 15:14 | IP_ADC_SEQB | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |
| 21:16 | - | These bits ignore writes, and read as 0. |
| 23:22 | IP_ADC_THCMP | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |
| 29:24 | - | These bits ignore writes, and read as 0. |
| 31:30 | IP_ADC_OVR | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |

### 5.4.10 Interrupt Priority Register 5

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

**Table 49. Interrupt Priority Register 5 (IPR5, address 0xE000 E414) bit description**

| Bit | Symbol | Description |
|---|---|---|
| 5:0 | - | These bits ignore writes, and read as 0. |
| 7:6 | - | Reserved. |
| 13:8 | - | These bits ignore writes, and read as 0. |
| 15:14 | - | Reserved. |
| 21:16 | - | These bits ignore writes, and read as 0. |
| 23:22 | - | Reserved. |
| 29:24 | - | Reserved. |
| 31:30 | IP_CT32B0 | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |

### 5.4.11 Interrupt Priority Register 6

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

**Table 50. Interrupt Priority Register 6 (IPR6, address 0xE000 E418) bit description**

| Bit | Symbol | Description |
|---|---|---|
| 5:0 | - | These bits ignore writes, and read as 0. |
| 7:6 | IP_PININT0 | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |
| 13:8 | - | These bits ignore writes, and read as 0. |
| 15:14 | IP_PININT1 | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |
| 21:16 | - | These bits ignore writes, and read as 0. |
| 23:22 | IP_PININT2 | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |
| 29:24 | - | These bits ignore writes, and read as 0. |
| 31:30 | IP_PININT3 | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |

### 5.4.12 Interrupt Priority Register 7

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

**Table 51. Interrupt Priority Register 7 (IPR7, address 0xE000 E41C) bit description**

| Bit | Symbol | Description |
|---|---|---|
| 5:0 | - | These bits ignore writes, and read as 0. |
| 7:6 | IP_PININT4 | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |
| 13:8 | - | These bits ignore writes, and read as 0. |
| 15:14 | IP_PININT5 | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |
| 21:16 | - | These bits ignore writes, and read as 0. |
| 23:22 | IP_PININT6 | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |
| 29:24 | - | These bits ignore writes, and read as 0. |
| 31:30 | IP_PININT7 | Interrupt Priority. 0 = highest priority. 3 = lowest priority. |

## 6.1 How to read this chapter

The system configuration block is identical for all LPC804 parts.

## 6.2 Features

- Clock control
  - Configure one low power oscillator and FRO oscillator.
  - Enable clocks to individual peripherals and memories.
  - Configure clock output.
  - Configure clock dividers and USART baud rate clock.
  - Configure ADC clock.
  - Configure Capacitive Touch clock.
- Monitor and release reset to individual peripherals.
- Select pins for external pin interrupts and pattern match engine.
- Configuration of reduced power modes.
- Wake-up control.
- BOD configuration.
- Interrupt latency control.
- Select a source for the NMI.
- Calibrate system tick timer.

## 6.3 Basic configuration

Configure the SYSCON block as follows:

- The SYSCON uses the CLKIN, CLKOUT, and $\overline{\text{RESET}}$ pins. Configure the pin functions through the switch matrix. See Section 6.4.
- No clock configuration is needed. The clock to the SYSCON block is always enabled. By default, the SYSCON block is clocked by the FRO.

### 6.3.1 Set up the FRO

The FRO provides a selectable fro_oscout of 18 MHz, 24 MHz, and 30 MHz outputs, which are divided to 9 MHz, 12 MHz, 15 MHz outputs to use as a system clock. Also, these frequencies can be divided down to provide frequencies (fro_div) of 4.5 MHz, 6 MHz, and 7.5 MHz for the system clock.

By default, the fro_oscout is 24 MHz and is divided by 2 to provide a default system (CPU) clock frequency of 12 MHz.

1. By default, the FRO is enabled. If required, the FRO can be enabled in the PDRUNCFG register:

Section 6.6.30 “Power configuration register”

2. Select the fro_oscout (30 MHz/24 MHz/18 MHz) using the set_fro_frequency API call: Chapter 7 “LPC804 FRO API ROM routine” and Figure 9 “LPC804 FRO subsystem”.

### 6.3.2 Configure the main clock and system clock

The clock source for the registers and memories is derived from main clock.

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. You have the following options:
   - FRO: 12 MHz internal oscillator (default).
   - External clock input: CLKIN from external pin.
   - Low power oscillator.
   - FRO DIV: 6 MHz (default).

   Section 6.6.3 “Main clock source select register”
2. Update the main clock source.

   Section 6.6.4 “Main clock source update enable register”
3. Select the divider value for the system clock. A divider value of 0 disables the system clock.

   Section 6.6.5 “System clock divider register”
4. Select the memories and peripherals that are operating in your application and therefore must have an active clock. The core is always clocked.

   Section 6.6.10 “System clock control 0 register”

## 6.4 Pin description

The SYSCON inputs and outputs are assigned to external pins through the switch matrix.

See Section 8.3.1 “Connect an internal signal to a package pin” to assign the CLKOUT function to a pin.

See Section 8.3.2 to enable the clock input and the external reset input.

**Table 52. SYSCON pin description**

| Function | Direction | Pin | Description | SWM register | Reference |
|---|---|---|---|---|---|
| CLKOUT | O | any | CLKOUT clock output. | PINASSIGN5 | Table 99 |
| CLKIN | I | PIO0_1/ACMP_I2/CLKIN/ ADC_0 | External clock input to the main clock. Disable the ACMP_I2/ADC_0 function in the PINENABLE register. | PINENABLE0 | Table 102 |
| $\overline{\text{RESET}}$ | I | $\overline{\text{RESET}}$/PIO0_5 | External reset input | PINENABLE0 | Table 102 |

## 6.5 General description

### 6.5.1 Clock generation

The system control block generates all clocks for the chip. Except for the USART clock, SPI clock, I2C clock, and ADC clock, the clocks to the core and peripherals run at the same frequency. The maximum system clock frequency is 15 MHz. See Figure 6 “LPC804 clock generation aaa-get number”.

**Table 53. Clocking diagram signal name descriptions**

| Name | Description |
|---|---|
| clk_in | The internal clock that comes from the main CLK_IN pin function. That function must be connected to the pin by selecting it in the SWM block. |
| frg_clk | The output of the Fractional Rate Generator. The FRG and its source selection are shown in Figure 6. |
| fro_div | Divided output of the currently selected on-chip FRO oscillator. See Figure 9 “LPC804 FRO subsystem”. |
| fro | The output of the currently selected on-chip FRO oscillator. See Figure 9 “LPC804 FRO subsystem”. |
| main_clk | The main clock used by the CPU and AHB bus, and potentially many others. The main clock and its source selection are shown in Figure 6. |
| “none” | A tied-off source that should be selected to save power when the output of the related multiplexer is not used. |
| lposc_clk | The output of the 1 MHz low power oscillator. It must also be enabled in the PDRUNCFG0 register. See Section 6.6.30. |

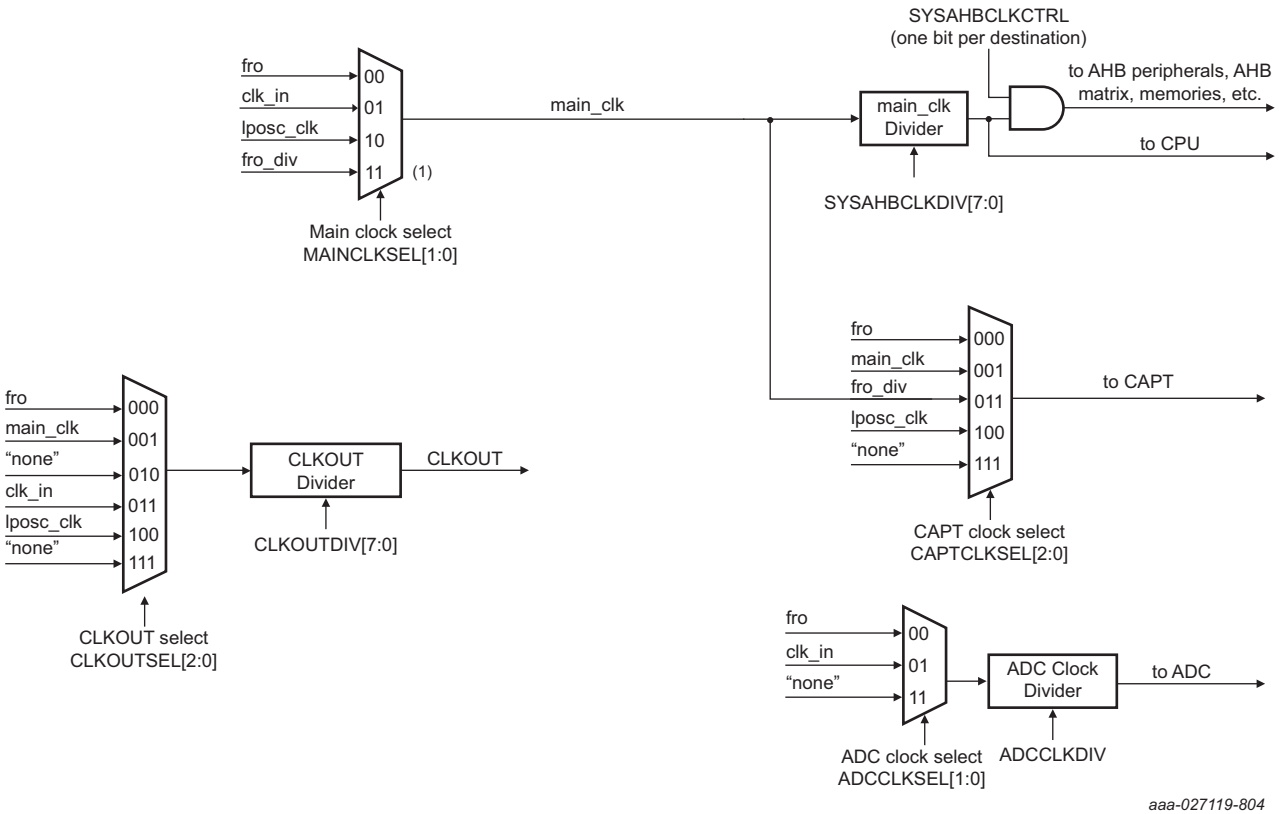



**Fig 6. LPC804 clock generation aaa-get number**

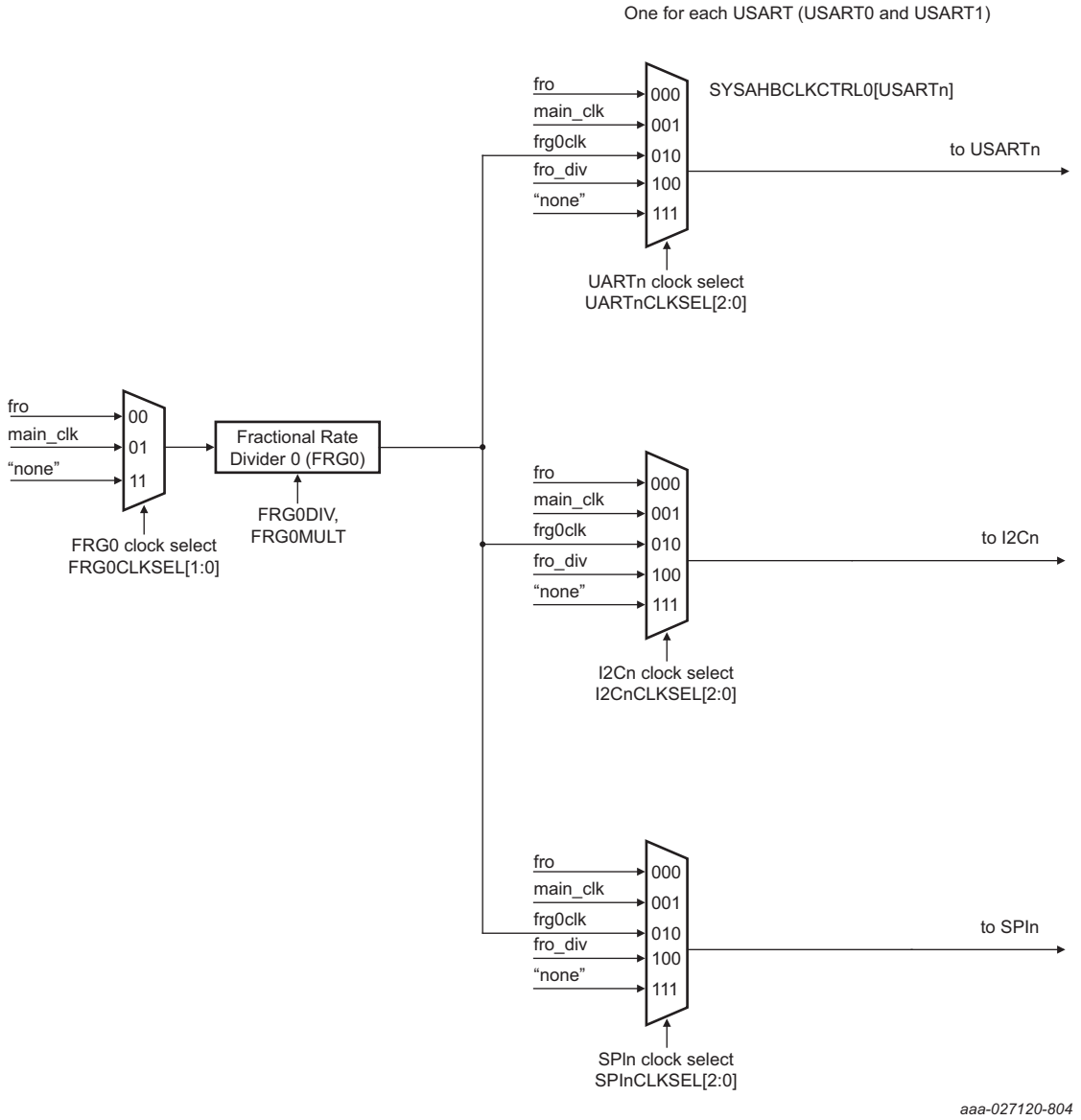



**Fig 7. LPC804 clock generation (continued) aaa-get number**

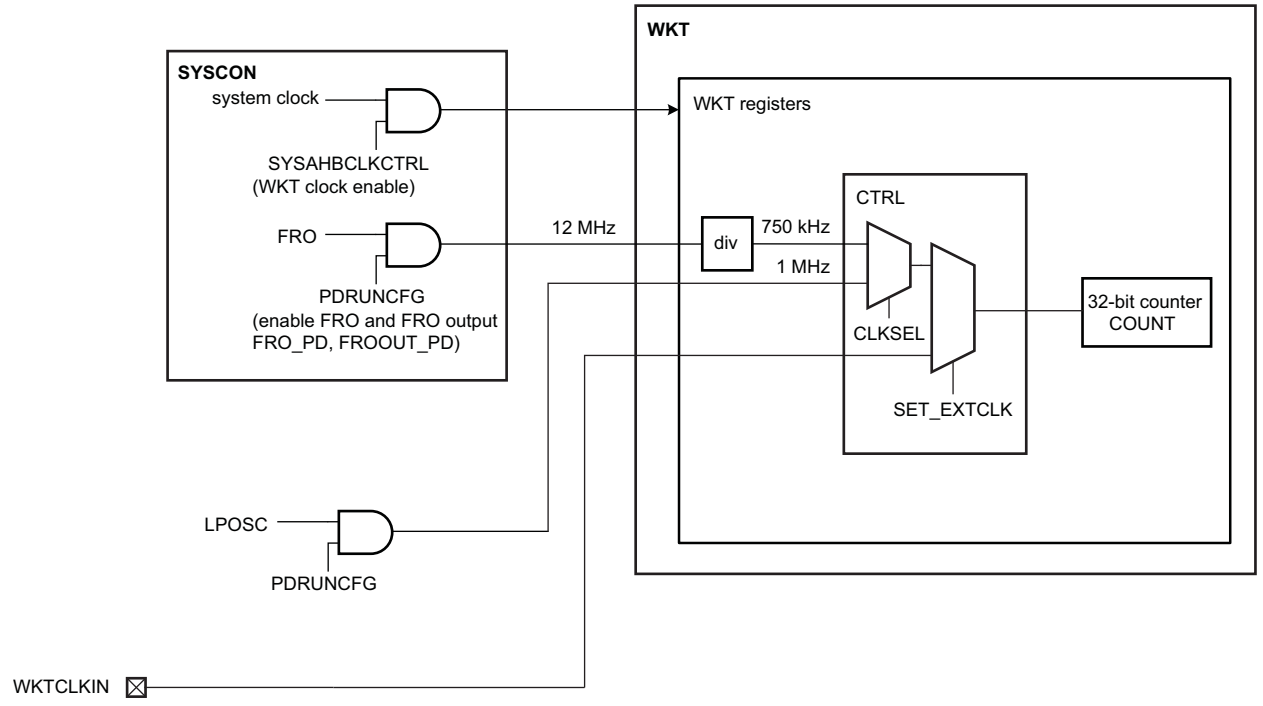



**Fig 8. LPC804 WKT clocking**

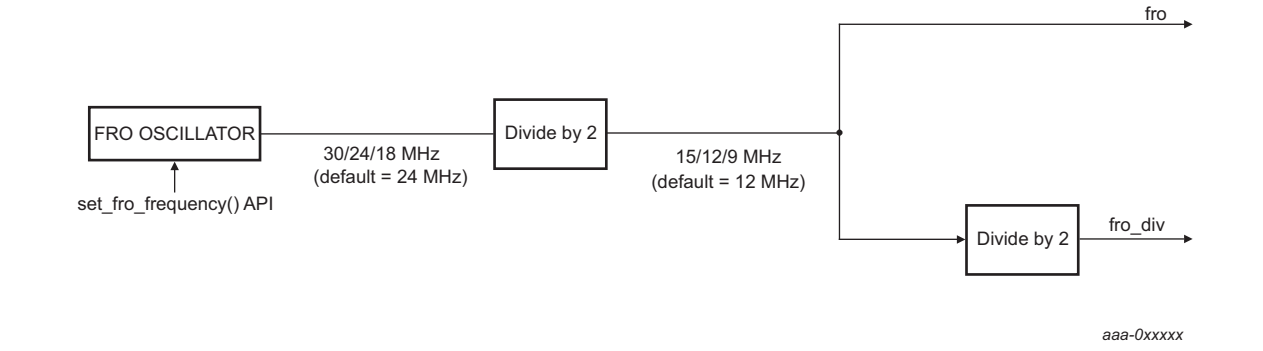



**Fig 9. LPC804 FRO subsystem**

### 6.5.2 Power control of analog components

The system control block controls the power to the analog components such as the low-power oscillator, the BOD, and the analog comparator. For details, see the following registers:

Section 6.6.28 “Deep-sleep mode configuration register”

Section 6.6.9 “WDT and Wake Timer clock enable control register”

### 6.5.3 Configuration of reduced power-modes

The system control block configures analog blocks that can remain running in the reduced power modes (the BOD and the low power oscillator for safe operation) and enables various interrupts to wake up the chip when the internal clocks are shut down in deep-sleep and power-down modes. For details, see the following registers:

Section 6.6.30 “Power configuration register”

Section 6.6.27 “Start logic 1 interrupt wake-up enable register”

### 6.5.4 Reset and interrupt control

The peripheral reset control register in the system control register allows to assert and release individual peripheral resets.

Up to eight external pin interrupts can be assigned to any digital pin in the system control block (see Section 6.6.25 “Pin interrupt select registers”).

## 6.6 Register description

All system control block registers reside on word address boundaries. Details of the registers appear in the description of each function.

Reset values describe the content of the registers after the bootloader has executed.

All address offsets shown in Table 54 as reserved should not be written to.

**Table 54. Register overview: System configuration (base address 0x4004 8000)**

| Name | Access | Offset | Description | Reset value | Section |
|---|---|---|---|---|---|
| SYSMEMREMAP | R/W | 0x000 | System memory remap. | 0x2 | 6.6.1 |
| - | - | 0x004 - 0x034 | Reserved. | - | - |
| SYSRSTSTAT | R/W | 0x038 | System reset status register. | 0 | 6.6.2 |
| - | - | 0x040 - 0x04C | Reserved. | - | - |
| MAINCLKSEL | R/W | 0x050 | Main clock source select. | 0 | 6.6.3 |
| MAINCLKUEN | R/W | 0x054 | Main clock source update enable. | 0 | 6.6.4 |
| SYSAHBCLKDIV | R/W | 0x058 | System clock divider. | 1 | 6.6.5 |
| | - | 0x05C | Reserved. | - | - |
| CAPTCLKSEL | R/W | 0x060 | Capacitive Touch clock source select. | 0x7 | 6.6.6 |
| ADCCLKSEL | R/W | 0x064 | ADC clock source select. | 0 | 6.6.7 |
| ADCCLKDIV | R/W | 0x068 | ADC clock divider. | 0 | 6.6.8 |
| - | - | 0x06C - 0x078 | Reserved. | 0 | - |
| LPOSCCLKEN | R/W | 0x07C | WDT and Wake Timer clock enable control. | 0x1 | 6.6.9 |
| SYSAHBCLKCTRL0 | R/W | 0x080 | System clock control 0. | 0x1F | 6.6.10 |
| SYSAHBCLKCTRL1 | R/W | 0x084 | System clock control 0. | 0x0 | 6.6.11 |
| PRESETCTRL0 | R/W | 0x088 | Peripheral reset control 0. | 0xFFFFFFFF | 6.6.12 |
| PRESETCTRL1 | R/W | 0x08C | Peripheral reset control 1. | 0x3F | 6.6.13 |
| UART0CLKSEL | R/W | 0x090 | Function clock source select for UART0. | 0x7 | 6.6.14 |
| UART1CLKSEL | R/W | 0x094 | Function clock source select for UART1. | 0x7 | 6.6.14 |

**Table 54. Register overview: System configuration (base address 0x4004 8000)** *...continued*

| Name | Access | Offset | Description | Reset value | Section |
|---|---|---|---|---|---|
| - | R/W | 0x098 - 0x0A0 | Reserved. | - | - |
| I2C0CLKSEL | R/W | 0x0A4 | Function clock source select for $I^2C0$. | 0x7 | 6.6.14 |
| I2C1CLKSEL | R/W | 0x0A8 | Function clock source select for $I^2C1$. | 0x7 | 6.6.14 |
| - | R/W | 0x0AC - 0x0B0 | Reserved. | - | - |
| SPI0CLKSEL | R/W | 0x0B4 | Function clock source select for SPI0. | 0x7 | 6.6.14 |
| - | R/W | 0x0B8 - 0x0CC | Reserved. | - | - |
| FRG0DIV | R/W | 0x0D0 | Fractional generator divider value. | 0x0 | 6.6.15 |
| FRG0MULT | R/W | 0x0D4 | Fractional generator multiplier value. | 0x0 | 6.6.16 |
| FRG0CLKSEL | R/W | 0x0D8 | FRG0 clock source select. | 0 | 6.6.17 |
| - | - | 0x0DC - 0x0EC | Reserved. | - | - |
| CLKOUTSEL | R/W | 0x0F0 | CLKOUT clock source select. | 0 | 6.6.18 |
| CLKOUTDIV | R/W | 0x0F4 | CLKOUT clock divider. | 0 | 6.6.19 |
| - | - | 0x0F8 - 0x0FC | Reserved. | - | - |
| PIOPORCAP0 | R | 0x100 | POR captured PIO0 status 0. | User dependent | 6.6.20 |
| - | R | 0x104 - 0x14C | Reserved. | - | - |
| BODCTRL | R/W | 0x150 | Brown-Out Detect. | 0x10 | 6.6.21 |
| SYSTCKCAL | R/W | 0x154 | System tick counter calibration. | 0 | 6.6.22 |
| - | R/W | 0x158 - 0x16C | Reserved. | - | - |
| IRQLATENCY | R/W | 0x170 | IRQ delay. Allows trade-off between interrupt latency and determinism. | 0x0000 0010 | 6.6.23 |
| NMISRC | R/W | 0x174 | NMI Source Control. | 0 | 6.6.24 |
| PINTSEL0 | R/W | 0x178 | GPIO Pin Interrupt Select register 0. | 0 | 6.6.25 |
| PINTSEL1 | R/W | 0x17C | GPIO Pin Interrupt Select register 1. | 0 | 6.6.25 |
| PINTSEL2 | R/W | 0x180 | GPIO Pin Interrupt Select register 2. | 0 | 6.6.25 |
| PINTSEL3 | R/W | 0x184 | GPIO Pin Interrupt Select register 3. | 0 | 6.6.25 |
| PINTSEL4 | R/W | 0x188 | GPIO Pin Interrupt Select register 4. | 0 | 6.6.25 |
| PINTSEL5 | R/W | 0x18C | GPIO Pin Interrupt Select register 5. | 0 | 6.6.25 |
| PINTSEL6 | R/W | 0x190 | GPIO Pin Interrupt Select register 6. | 0 | 6.6.25 |
| PINTSEL7 | R/W | 0x194 | GPIO Pin Interrupt Select register 7. | 0 | 6.6.25 |
| - | - | 0x198 - 0x200 | Reserved. | - | - |
| STARTERP0 | R/W | 0x204 | Start logic 0 pin wake-up enable register . | 0 | 6.6.26 |
| - | - | 0x208 - 0x210 | Reserved. | - | - |
| STARTERP1 | R/W | 0x214 | Start logic 1 interrupt wake-up enable register. | 0 | 6.6.27 |
| - | - | 0x218 - 0x22C | Reserved. | - | - |
| PDSLEEPCFG | R/W | 0x230 | Power-down states in deep-sleep mode. | 0xFFFF | 6.6.28 |
| PDAWAKECFG | R/W | 0x234 | Power-down states for wake-up from deep-sleep. | 0xA050 | 6.6.29 |
| PDRUNCFG | R/W | 0x238 | Power configuration register. | 0xA050 | 6.6.30 |
| - | - | 0x23C - 0x3F4 | Reserved. | - | - |
| DEVICE_ID | R | 0x3F8 | Device ID. | part dependent | 6.6.31 |

### 6.6.1 System memory remap register

The system memory remap register selects whether the exception vectors are read from boot ROM, flash, or SRAM. By default, the flash memory is mapped to address 0x0000 0000. When the MAP bits in the SYSMEMREMAP register are set to 0x0 or 0x1, the boot ROM or RAM respectively are mapped to the bottom 512 bytes of the memory map (addresses 0x0000 0000 to 0x0000 0200).

**Table 55. System memory remap register (SYSMEMREMAP, address 0x4004 8000) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 1:0 | MAP | | System memory remap. Value 0x3 is reserved. | 0x2 |
| | | 0x0 | Bootloader Mode. Interrupt vectors are re-mapped to Boot ROM. | |
| | | 0x1 | User RAM Mode. Interrupt vectors are re-mapped to Static RAM. | |
| | | 0x2 | User Flash Mode. Interrupt vectors are not re-mapped and reside in Flash. | |
| 31:2 | - | - | Reserved | - |

### 6.6.2 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 $\overline{\text{RESET}}$ pin - remains asserted after the POR signal is negated, then its bit is set to detected. Write a one to clear the reset.

The reset value given in Table 56 applies to the POR reset.

**Table 56. System reset status register (SYSRSTSTAT, address 0x4004 8038) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | POR_BOD | | POR and BOD reset status | 0 |
| | | 0 | No POR and BOD detected | |
| | | 1 | POR and BOD detected. Writing a one clears this reset. | |
| 1 | EXTRST | | Status of the external $\overline{\text{RESET}}$ pin. External reset status. | 0 |
| | | 0 | No reset event detected. | |
| | | 1 | Reset detected. Writing a one clears this reset. | |
| 2 | WDT | | Status of the Watchdog reset. | 0 |
| | | 0 | No WDT reset detected. | |
| | | 1 | WDT reset detected. Writing a one clears this reset. | |
| 3 | - | - | Reserved | - |
| 4 | SYSRST | | Status of the software system reset | 0 |
| | | 0 | No System reset detected | |
| | | 1 | System reset detected. Writing a one clears this reset. | |
| 31:5 | - | - | Reserved | - |

### 6.6.3 Main clock source select register

The MAINCLKSEL register selects the main_clock, which can be the FRO, external clock, low power oscillator, or FRO_DIV.

Bit 0 of the MAINCLKUEN register (Section 6.6.4) must be toggled from 0 to 1 for the update to take effect.

**Table 57. Main clock source select register (MAINCLKSEL, address 0x4004 8050) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 1:0 | SEL | | Clock source for main clock. | 0 |
| | | 0x0 | FRO. | |
| | | 0x1 | External clock. | |
| | | 0x2 | Low power oscillator. | |
| | | 0x3 | FRO_DIV. | |
| 31:2 | - | - | Reserved. | - |

### 6.6.4 Main clock source update enable register

The **MAINCLKUEN** register updates the clock source of the main clock with the new input clock after the MAINCLKSEL register has been written to. In order for the update to take effect, first write a zero to bit 0 of this register, then write a one.

**Table 58. Main clock source update enable register (MAINCLKUEN, address 0x4004 8054) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | ENA | | Enable main clock source update. | 0 |
| | | 0 | No change. | |
| | | 1 | Update clock source. | |
| 31:1 | - | - | Reserved. | - |

### 6.6.5 System clock divider register

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

**Table 59. System clock divider register (SYSAHBCLKDIV, address 0x4004 8058) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 7:0 | DIV | System AHB clock divider values:<br>0: System clock disabled.<br>1: Divide by 1.<br>…<br>255: Divide by 255. | 0x01 |
| 31:8 | - | Reserved. | - |

### 6.6.6 Capacitive Touch clock source select register

The CAPTCLKSEL register selects the CAPT clock, which can be the FRO, main clock, FRO_DIV, or low power oscillator.

**Table 60. CAPT clock source select register (CAPTCLKSEL, address 0x4004 8060) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | SEL | | Peripheral clock source. | 0x7 |
| | | 0x0 | FRO. | |
| | | 0x1 | Main clock. | |
| | | 0x2 | Reserved. | |
| | | 0x3 | FRO_DIV = FRO/2. | |
| | | 0x4 | Low power oscillator. | |
| | | 0x5 | Reserved. | |
| | | 0x6 | Reserved. | |
| | | 0x7 | None. | |
| 31:3 | - | | Reserved. | - |

### 6.6.7 ADC clock source select register

The ADCCLKSEL register selects the ADC clock, which can be the FRO or external_clk.

**Table 61. ADC clock source select register (ADCCLKSEL, address 0x4004 8064) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 1:0 | SEL | | Clock source for ADC clock. | 0x0 |
| | | 0x0 | FRO. | |
| | | 0x1 | External clock. | |
| | | 0x2 | Reserved. | |
| | | 0x3 | None. | |
| 31:3 | - | | Reserved. | - |

### 6.6.8 ADC clock divider register

The ADCCLKDIV register controls how the ADC clock is divided to provide the ADC clock to the ADC controller. The ADC clock can be shut down completely by setting the DIV field to zero.

**Table 62. ADC clock divider register (ADCCLKDIV, address 0x4004 8068) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 7:0 | DIV | | ADC clock divider values<br>0: ADC clock disabled.<br>1: Divide by 1.<br>…<br>255: Divide by 255. | 0x0 |
| 31:8 | - | | Reserved. | - |

### 6.6.9 WDT and Wake Timer clock enable control register

**Table 63. WDT and Wake Timer clock enable control register (LPOSCCLKEN, address 0x4004807C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | WDT | | Enables clock for WDT | 1 |
| | | 0 | Disable | |
| | | 1 | Enable | |
| 1 | WKT | | Enables clock for Wake Timer | 0 |
| | | 0 | Disable | |
| | | 1 | Enable | |
| 31:2 | - | | Reserved | 0 |

### 6.6.10 System clock control 0 register

The SYSAHBCLKCTRL0 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 Arm Cortex-M0+, the SYSCON block, and the PMU. This clock cannot be disabled.

**Table 64. System clock control 0 register (SYSAHBCLKCTRL0, address 0x4004 8080) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | SYS | | Enables the clock for the AHB, the APB bridge, the Cortex-M0+ core clocks, SYSCON, and the PMU. This bit is read only and always reads as 1. | 1 |
| 1 | ROM | | Enables clock for ROM. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 2 | RAM0 | | Enables clock for SRAM0. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 3 | | | Reserved. | 1 |
| 4 | FLASH | | Enables clock for flash. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 5 | I2C0 | | Enables clock for I2C0. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | GPIO0 | | Enables clock for GPIO0 port registers. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 7 | SWM | | Enables clock for switch matrix. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 8 | - | - | Reserved. | 0 |

**Table 64. System clock control 0 register (SYSAHBCLKCTRL0, address 0x4004 8080) bit description** *...continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 9 | WKT | | Enables clock for self-wake-up timer. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 10 | MRT | | Enables clock for multi-rate timer. | |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 11 | SPI0 | | Enables clock for SPI0. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 12 | | | Reserved. | |
| 13 | CRC | | Enables clock for CRC. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 14 | UART0 | | Enables clock for USART0. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 15 | UART1 | | Enables clock for USART1. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 16 | - | - | Reserved. | 0 |
| 17 | WWDT | | Enables clock for WWDT. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 18 | IOCON | | Enables clock for IOCON block. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 19 | ACMP | | Enables clock to analog comparator. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 20 | - | - | Reserved. | 0 |
| 21 | I2C1 | | Enables clock for I2C1. | 0 |
| | | 0x2 | Disable. | |
| | | 1 | Enable. | |
| 23:22 | - | - | Reserved. | 0 |
| 24 | ADC | | Enables clock to ADC. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 25 | CTIMER0 | - | Enables clock for CTIMER0. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |

**Table 64. System clock control 0 register (SYSAHBCLKCTRL0, address 0x4004 8080) bit description** *...continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 26 | - | - | Reserved. | 0 |
| 27 | DAC | | Enables clock for DAC. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 28 | GPIO_INT | | Enables clock for GPIO pin interrupt registers. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 31:29 | - | - | Reserved. | 0 |

### 6.6.11 System clock control 1 register

The SYSAHBCLKCTRL1 register enables the clocks to peripheral blocks.

**Table 65. System clock control 1 register (SYSAHBCLKCTRL1, address 0x4004 8084) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | CAPT | | Enables the clock for Capacitive Touch. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 4:1 | - | - | Reserved. | 0 |
| 5 | PLU | | Enables clock for PLU. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 31:6 | - | - | Reserved. | 0 |

### 6.6.12 Peripheral reset control 0 register

The PRESET0CTRL register allows software to reset specific peripherals. A zero in any assigned bit in this register resets the specified peripheral. A 1 clears the reset and allows the peripheral to operate.

**Table 66. Peripheral reset control 0 register (PRESETCTRL0, address 0x4004 8088) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 3:0 | - | - | Reserved. | 4'b1111 |
| 4 | FLASH_RST_N | | Flash controller reset control. | 1 |
| | | 0 | Assert the flash controller reset. | |
| | | 1 | Clear the flash controller reset. | |
| 5 | I2C0_RST_N | | $I^2C0$ reset control. | 1 |
| | | 0 | Assert the $I^2C0$ reset. | |
| | | 1 | Clear the $I^2C0$ reset. | |
| 6 | GPIO0_RST_N | | GPIO0 reset control. | 1 |
| | | 0 | Assert the GPIO0 reset | |
| | | 1 | Clear the GPIO0 reset. | |
| 7 | SWM_RST_N | | SWM reset control. | 1 |
| | | 0 | Assert the SWM reset. | |
| | | 1 | Clear the SWM reset. | |
| 8 | - | - | Reserved. | 1 |
| 9 | WKT_RST_N | | Self-wake-up timer (WKT) reset control. | 1 |
| | | 0 | Assert the WKT reset. | |
| | | 1 | Clear the WKT reset. | |
| 10 | MRT_RST_N | | Multi-rate timer (MRT) reset control. | 1 |
| | | 0 | Assert the MRT reset. | |
| | | 1 | Clear the MRT reset. | |
| 11 | SPI0_RST_N | | SPI0 reset control. | 1 |
| | | 0 | Assert the SPI0 reset. | |
| | | 1 | Clear the SPI0 reset. | |
| 12 | - | - | Reserved. | 1 |
| 13 | CRC_RST_N | | CRC engine reset control | 1 |
| | | 0 | Assert the CRC reset. | |
| | | 1 | Clear the CRC reset. | |
| 14 | UART0_RST_N | | UART0 reset control. | 1 |
| | | 0 | Assert the UART0 reset. | |
| | | 1 | Clear the flash UART0 reset. | |
| 15 | UART1_RST_N | | UART1 reset control. | 1 |
| | | 0 | Assert the UART1 reset. | |
| | | 1 | Clear the UART1 reset. | |
| 17:16 | - | - | Reserved. | 1 |

**Table 66. Peripheral reset control 0 register (PRESETCTRL0, address 0x4004 8088) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 18 | IOCON_RST_N | | IOCON reset control. | 1 |
| | | 0 | Assert the IOCON reset. | |
| | | 1 | Clear the IOCON reset. | |
| 19 | ACMP_RST_N | | Analog comparator reset control. | 1 |
| | | 0 | Assert the analog comparator reset. | |
| | | 1 | Clear the analog comparator reset. | |
| 20 | - | - | Reserved. | 1 |
| 21 | I2C1_RST_N | | I2C1 reset control. | 1 |
| | | 0 | Assert the I2C1 reset. | |
| | | 1 | Clear the I2C1 reset. | |
| 23:22 | - | - | Reserved. | 2’b11 |
| 24 | ADC_RST_N | | ADC reset control. | 1 |
| | | 0 | Assert the ADC reset. | |
| | | 1 | Clear the ADC reset. | |
| 25 | CTIMER0_RST_N | | CTIMER reset control. | 1 |
| | | 0 | Assert the CTIMER reset. | |
| | | 1 | Clear the CTIMER reset. | |
| 26 | - | - | Reserved. | 1 |
| 27 | DAC0_RST_N | | DAC0 reset control. | 1 |
| | | 0 | Assert the DAC0 reset. | |
| | | 1 | Clear the DAC0 reset. | |
| 28 | GPIOINT_RST_N | | GPIOINT reset control. | 1 |
| | | 0 | Assert the GPIOINT reset. | |
| | | 1 | Clear the GPIOINT reset. | |
| 31:29 | - | - | Reserved. | 3’b11 |

### 6.6.13 Peripheral reset control 1 register

The PRESETCTRL1 register allows software to reset specific peripherals. A zero in any assigned bit in this register resets the specified peripheral. A 1 clears the reset and allows the peripheral to operate.

**Table 67. Peripheral reset control 1 register (PRESETCTRL1, address 0x4004 808C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | CAPT_RST_N | | Capacitive Touch reset control. | 1 |
| | | 0 | Assert Capacitive Touch reset. | |
| | | 1 | Clear Capacitive Touch reset. | |
| 2:1 | - | - | Reserved. | 2’b11 |
| 3 | FRG0_RST_N | | Fractional baud rate generator 0 reset control. | 1 |
| | | 0 | Assert the FRG0 reset. | |
| | | 1 | Clear the FRG0 reset. | |

Table 67. Peripheral reset control 1 register (PRESETCTRL1, address 0x4004 808C) bit description

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 4 | - | - | Reserved. | 1 |
| 5 | PLU_RST_N | | PLU reset control. | 1 |
| | | 0 | Assert PLU reset. | |
| | | 1 | Clear PLU reset. | |
| 31:6 | - | - | Reserved. | 0 |

### 6.6.14 Peripheral clock source select registers

The peripheral clock source select registers select function clock sources for the serial peripherals shown in the following list. The potential clock sources are the same for each peripheral. See Table 68.

- UART0 clock source select register (UART0CLKSEL, address 0x4004 8090).
- UART1 clock source select register (UART1CLKSEL, address 0x4004 8094).
- $I^2C0$ clock source select register (I2C0CLKSEL, address 0x4004 80A4).
- $I^2C1$ clock source select register (I2C1CLKSEL, address 0x4004 80A8).
- SPI0 clock source select register (SPI0CLKSEL, address 0x4004 80B4).

Table 68. Peripheral clock source select registers

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | SEL | | Peripheral clock source. | 0x7 |
| | | 0x0 | FRO. | |
| | | 0x1 | Main clock. | |
| | | 0x2 | FRG0 clock. | |
| | | 0x3 | Reserved. | |
| | | 0x4 | FRO_DIV. | |
| | | 0x5 | Reserved. | |
| | | 0x6 | Reserved. | |
| | | 0x7 | None. | |
| 31:3 | - | - | Reserved. | - |

### 6.6.15 Fractional generator 0 divider value register

The UART, $I^2C$, SPI clock come from the FCLK multiplexer. The FRGCLK0 is one clock source of the FCLK multiplexer and its output from the fractional generator 0 can be adjusted by a fractional divider:

frg0clk = frg0_src_clk/(1 + MULT/DIV).

FRG0_SRC_CLK is input clock of fractional generator 0, which can be the FRO or main clock.

The fractional portion (1 + MULT/DIV) is determined by the two fractional divider registers in the SYSCON block:

- The DIV value programmed in this register is the denominator of the divider used by the fractional rate generator to create the fractional component of FRG0CLK.
- The MULT value of the fractional divider is programmed in the FRG0MULT register. See Table 70.

**Remark:** To use of the fractional baud rate generator, you must write 0xFF to this register to yield a denominator value of 256. All other values are not supported.

See also:

Section 13.3.1 “Configure the USART clock and baud rate”.

Section 13.7.1 “Clocking and baud rates”.

**Table 69. Fractional generator 0 divider value register (FRG0DIV, address 0x4004 80D0) bit description**

| Bit | Symbol | Value | 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. | 0 |
| 31:8 | - | - | Reserved | - |

### 6.6.16 Fractional generator 0 multiplier value register

The UART, $I^2C$, and SPI clocks come from the FCLK multiplexer. The FRG0CLK is one clock source of the FCLK multiplexer and its output from the fractional generator 0 can be adjusted by a fractional divider:

frg0clk = frg0_src_clk/(1 + MULT/DIV).

FRG0_SRC_CLK is input clock of fractional generator 0, which can be the FRO or main clock.

The fractional portion (1 + MULT/DIV) is determined by the two fractional divider registers in the SYSCON block:

- The DIV denominator of the fractional divider value is programmed in the FRG0DIV register. See Table 69.
- The MULT value programmed in this register is the numerator of the fractional divider value used by the fractional rate generator to create the fractional component to the baud rate.

**Remark:** To use of the fractional baud rate generator, you must write 0xFF to this register to yield a denominator value of 256. All other values are not supported.

See also:

Section 13.3.1 “Configure the USART clock and baud rate”.

Section 13.7.1 “Clocking and baud rates”.

**Table 70. Fractional generator 0 multiplier value register (FRG0MULT, address 0x4004 80D4) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 7:0 | MULT | | Numerator of the fractional divider. MULT is equal to the programmed value. | 0 |
| 31:8 | - | - | Reserved. | - |

### 6.6.17 FRG0 clock source select register

The FRG0CLKSEL register selects the frg0_src clock, which can be the FRO or main clock.

**Table 71. FRG0 clock source select register (FRG0CLKSEL, address 0x4004 80D8) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 1:0 | SEL | | Clock source for FRG0_SRC clock. | 0x0 |
| | | 0x0 | FRO. | |
| | | 0x1 | Main clock. | |
| | | 0x2 | Reserved. | |
| | | 0x3 | None. | |
| 31:2 | - | - | Reserved. | - |

### 6.6.18 CLKOUT clock source select register

This register selects the signal visible on the CLKOUT pin. Any oscillator or the main clock can be selected.

**Table 72. CLKOUT clock source select register (CLKOUTSEL, address 0x4004 80F0) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | SEL | | CLKOUT clock source. | 0 |
| | | 0x0 | FRO. | |
| | | 0x1 | Main clock. | |
| | | 0x2 | Reserved. | |
| | | 0x3 | External clock. | |
| | | 0x4 | Low power oscillator. | |
| | | 0x5 | Reserved. | |
| | | 0x6 | Reserved. | |
| | | 0x7 | None. | |
| 31:3 | - | - | Reserved. | 0 |

### 6.6.19 CLKOUT clock divider register

The **CLKOUTDIV** register determines the divider value for the signal on the CLKOUT pin.

**Table 73. CLKOUT clock divider registers (CLKOUTDIV, address 0x4004 80F4) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 7:0 | DIV | CLKOUT clock divider values.<br>0: Disable CLKOUT clock divider.<br>1: Divide by 1.<br>…<br>255: Divide by 255. | 0 |
| 31:8 | - | Reserved. | - |

### 6.6.20 POR captured PIO0 status register 0

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

**Table 74. POR captured PIO status register 0 (PIOPORCAP0, address 0x4004 8100) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 31:0 | PIOSTAT | State of PIO0_0 to PIO0_5 and PIO0_7 to PIO0_30 at power-on reset. | Implementation dependent |

### 6.6.21 BOD control register

The BOD control register selects three separate threshold values for sending a BOD interrupt to the NVIC and one threshold level for forced reset. Reset and interrupt threshold values listed in Table 75 are typical values.

Both the BOD interrupt and the BOD reset, depending on the value of bit BODRSTENA in this register, can wake-up the chip from sleep, deep-sleep, and power-down modes.

See the LPC804 data sheet for the BOD reset and interrupt levels.

**Table 75. BOD control register (BODCTRL, address 0x4004 8150) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 1:0 | BODRSTLEV | | BOD reset level. | 0 |
| | | 0x0 | Level 0. | |
| 3:2 | BODINTVAL | | BOD interrupt level. | 0 |
| | | 0x0 | Reserved. | |
| | | 0x1 | Level 1. | |
| | | 0x2 | Level 2. | |
| | | 0x3 | Level 3. | |
| 4 | BODRSTENA | | BOD reset enable. | 1 |
| | | 0 | Disable reset function. | |
| | | 1 | Enable reset function. | |
| 31:5 | - | - | Reserved. | 0x00 |

### 6.6.22 System tick counter calibration register

This register determines the value of the SYST_CALIB register.

**Table 76. System tick timer calibration register (SYSTCKCAL, address 0x4004 8154) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 25:0 | CAL | System tick timer calibration value. | 0 |
| 31:26 | - | Reserved. | - |

### 6.6.23 IRQ latency register

The IRQLATENCY register is an eight-bit register which specifies the minimum number of cycles (0-255) permitted for the system to respond to an interrupt request. The intent of this register is to allow the user to select a trade-off between interrupt response time and determinism.

Setting this parameter to a very low value (for example, zero) will guarantee the best possible interrupt performance but will also introduce a significant degree of uncertainty and jitter. Requiring the system to always take a larger number of cycles (whether it needs it or not) will reduce the amount of uncertainty but may not necessarily eliminate it.

Theoretically, the Arm Cortex-M0+ core should always be able to service an interrupt request within 15 cycles. However, system factors external to the CPU, such as bus latencies or peripheral response times, can increase the time required to complete a previous instruction before an interrupt can be serviced. Therefore, accurately specifying a minimum number of cycles that will ensure determinism will depend on the application.

The default setting for this register is 0x010.

**Table 77. IRQ latency register (IRQLATENCY, address 0x4004 8170) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 7:0 | LATENCY | 8-bit latency value. | 0x010 |
| 31:8 | - | Reserved. | - |

### 6.6.24 NMI source selection register

The NMI source selection register selects a peripheral interrupt as source for the NMI interrupt of the Arm Cortex-M0+ core. For a list of all peripheral interrupts and their IRQ numbers see Table 38. For a description of the NMI functionality, see Section 5.3.2 “Non-Maskable Interrupt (NMI)”.

**Remark:** When you want to change the interrupt source for the NMI, you must first disable the NMI source by setting bit 31 in this register to 0. Then change the source by updating the IRQN bits and re-enable the NMI source by setting bit 31 to 1.

**Table 78. NMI source selection register (NMISRC, address 0x4004 8174) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 4:0 | IRQN | The IRQ number of the interrupt that acts as the Non-Maskable Interrupt (NMI) if bit 31 is 1. See Table 38 for the list of interrupt sources and their IRQ numbers. | 0 |
| 30:5 | - | Reserved. | - |
| 31 | NMIEN | Write a 1 to this bit to enable the Non-Maskable Interrupt (NMI) source selected by bits 4:0. | 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.

### 6.6.25 Pin interrupt select registers

Each of these 8 registers selects one pin from all digital pins as the source of a pin interrupt or as the input to the pattern match engine. To select a pin for any of the eight pin interrupts or pattern match engine inputs, write the GPIO port pin number as 0 to 5 and 7 to 30 for pins PIO0_0 to PIO0_5 and PIO0_7 to PIO0_30 to the INTPIN bits. For example, setting INTPIN to 0x5 in PINTSEL0 selects pin PIO0_5 for pin interrupt 0.

**Remark:** The GPIO port pin number serves to identify the pin to the PINTSEL register. Any digital input function, including GPIO, can be assigned to this pin through the switch matrix.

Each of the 8 pin interrupts must be enabled in the NVIC using interrupt slots # 24 to 31 (see Table 38).

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

**Table 79. Pin interrupt select registers (PINTSEL[0:7], address 0x4004 8178 (PINTSEL0) to 0x4004 8194 (PINTSEL7)) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 5:0 | INTPIN | Pin number select for pin interrupt or pattern match engine input. (PIO0_0 to PIO0_5 correspond to numbers 0 to 5. PIO0_7 to PIO0_30 correspond to numbers 7 to 30. | 0 |
| 31:6 | - | Reserved. | - |

### 6.6.26 Start logic 0 pin wake-up enable register

The STARTERP0 register enables the selected pin interrupts for wake-up from deep-sleep mode and power-down modes.

**Remark:** Also enable the corresponding interrupts in the NVIC. See Table 38 “Connection of interrupt sources to the NVIC”.

**Table 80. Start logic 0 pin wake-up enable register 0 (STARTERP0, address 0x4004 8204) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | PINT0 | | GPIO pin interrupt 0 wake-up. | 0 |
| | | 0 | Disabled. | |
| | | 1 | Enabled. | |
| 1 | PINT1 | | GPIO pin interrupt 1 wake-up. | 0 |
| | | 0 | Disabled. | |
| | | 1 | Enabled. | |
| 2 | PINT2 | | GPIO pin interrupt 2 wake-up. | 0 |
| | | 0 | Disabled . | |
| | | 1 | Enabled. | |
| 3 | PINT3 | | GPIO pin interrupt 3 wake-up. | 0 |
| | | 0 | Disabled. | |
| | | 1 | Enabled. | |
| 4 | PINT4 | | GPIO pin interrupt 4 wake-up. | 0 |
| | | 0 | Disabled . | |
| | | 1 | Enabled. | |
| 5 | PINT5 | | GPIO pin interrupt 5 wake-up. | 0 |
| | | 0 | Disabled. | |
| | | 1 | Enabled. | |
| 6 | PINT6 | | GPIO pin interrupt 6 wake-up. | 0 |
| | | 0 | Disabled. | |
| | | 1 | Enabled. | |
| 7 | PINT7 | | GPIO pin interrupt 7 wake-up. | 0 |
| | | 0 | Disabled. | |
| | | 1 | Enabled. | |
| 31:8 | - | - | Reserved. | - |

### 6.6.27 Start logic 1 interrupt wake-up enable register

This register selects which interrupts wake up the part from deep-sleep and power-down modes.

**Remark:** Also enable the corresponding interrupts in the NVIC. See Table 38 “Connection of interrupt sources to the NVIC”.

**Table 81. Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | SPI0 | | SPI0 interrupt wake-up. | 0 |
| | | 0 | Disabled. | |
| | | 1 | Enabled. | |
| 2:1 | - | | Reserved. | - |

**Table 81. Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 3 | USART0 | | USART0 interrupt wake-up. Configure USART in synchronous slave mode. | 0 |
| | | 0 | Disabled. | |
| | | 1 | Enabled. | |
| 4 | USART1 | | USART1 interrupt wake-up. Configure USART in synchronous slave mode. | 0 |
| | | 0 | Disabled. | |
| | | 1 | Enabled. | |
| 5 | - | | Reserved. | - |
| 6 | PLU | | PLU interrupt wake-up. | 0 |
| 7 | I2C1 | | I2C1 interrupt wake-up. | 0 |
| | | 0 | Disabled. | |
| | | 1 | Enabled. | |
| 8 | I2C0 | | I2C0 interrupt wake-up. | 0 |
| | | 0 | Disabled. | |
| | | 1 | Enabled. | |
| 10:9 | - | | Reserved. | - |
| 11 | CAPT | | CAPT interrupt wake-up. | |
| 12 | WWDT | | WWDT interrupt wake-up. | 0 |
| | | 0 | Disabled. | |
| | | 1 | Enabled. | |
| 13 | BOD | | BOD interrupt wake-up. | 0 |
| | | 0 | Disabled. | |
| | | 1 | Enabled. | |
| 14 | - | | Reserved. | - |
| 15 | WKT | | Self-wake-up timer interrupt wake-up. | 0 |
| | | 0 | Disabled. | |
| | | 1 | Enabled. | |
| 31:16 | - | | Reserved. | - |

### 6.6.28 Deep-sleep mode configuration register

The bits in this register (BOD_PD and LPOSC_OD) can be programmed to control aspects of deep-sleep and power-down modes. The bits are loaded into corresponding bits of the PDRUNCFG register when deep-sleep mode or power-down mode is entered.

**Remark:** Hardware forces the analog blocks to be powered down in deep-sleep and power-down modes. An exception are the BOD and low power oscillator, which can be configured to remain running through this register. The LPOSC_PD value written to the PDSLEEPCFG register is overwritten if the LOCK bit in the WWDT MOD register (see Table 238) is set. See Section 17.5.3 for details.

**Table 82. Deep-sleep configuration register (PDSLEEPCFG, address 0x4004 8230) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0b111 |
| 3 | BOD_PD | | BOD power-down control for deep-sleep and power-down mode. | 1 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |
| 5:4 | - | | Reserved. | 11 |
| 6 | LPOSC_PD | | Low power oscillator power-down control for deep-sleep and power-down mode. Changing this bit to powered-down has no effect when the LOCK bit in the WWDT MOD register is set. In this case, the low power oscillator is always running. | 1 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |
| 15:7 | - | - | Reserved. | 0b111111111 |
| 31:16 | - | - | Reserved. | 0 |

### 6.6.29 Wake-up configuration register

This register controls the power configuration of the device when waking up from deep-sleep or power-down mode.

**Table 83. Wake-up configuration register (PDAWAKECFG, address 0x4004 8234) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | FROOUT_PD | | FRO oscillator output wake-up configuration. | 0 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |
| 1 | FRO_PD | | FRO oscillator power-down wake-up configuration. | 0 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |
| 2 | FLASH_PD | | Flash wake-up configuration. | 0 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |
| 3 | BOD_PD | | BOD wake-up configuration. | 0 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |
| 4 | ADC_PD | | ADC wake-up configuration. | 1 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |
| 5 | - | | Reserved. | - |
| 6 | LPOSC_PD | | Low power oscillator wake-up configuration. Changing this bit to powered-down has no effect when the LOCK bit in the WWDT MOD register is set. In this case, the low power oscillator is always running. | 1 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |

**Table 83. Wake-up configuration register (PDAWAKECFG, address 0x4004 8234) bit description** *...continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 12:7 | - | | Reserved. Always write these bits as 0x00. | 0x00 |
| 13 | DAC0 | | DAC0 wake-up configuration. | 1 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |
| 14 | - | | Reserved. | 0 |
| 15 | ACMP | | Analog comparator wake-up configuration. | 1 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |
| 31:16 | - | - | Reserved. | 0 |

### 6.6.30 Power configuration register

The PDRUNCFG register controls the power to the various analog blocks. This register can be written to at any time while the chip is running, and a write will take effect immediately with the exception of the power-down signal to the FRO.

To avoid glitches when powering down the FRO, the FRO clock is automatically switched off at a clean point. Therefore, for the FRO a delay is possible before the power-down state takes effect.

**Table 84. Power configuration register (PDRUNCFG, address 0x4004 8238) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | FROOUT_PD | | FRO oscillator output power. | 0 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |
| 1 | FRO_PD | | FRO oscillator power down. | 0 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |
| 2 | FLASH_PD | | Flash power down. | 0 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |
| 3 | BOD_PD | | BOD power down. | 0 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |
| 4 | ADC_PD | | ADC wake-up configuration. | 1 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |
| 5 | - | - | Reserved. | 1 |
| 6 | LPOSC_PD | | Low power oscillator power down. Changing this bit to powered-down has no effect when the LOCK bit in the WWDT MOD register is set. In this case, the low power oscillator is always running. | 1 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |

**Table 84. Power configuration register (PDRUNCFG, address 0x4004 8238) bit description** *...continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 12:7 | - | | Reserved. Always write these bits as 0x00. | 0x00 |
| 13 | DAC0 | | DAC0 wake-up configuration. | 1 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |
| 14 | - | | Reserved. | 0 |
| 15 | ACMP | | Analog comparator power down. | 1 |
| | | 0 | Powered. | |
| | | 1 | Powered down. | |
| 31:16 | - | - | Reserved. | 0 |

### 6.6.31 Device ID register

The device ID register is a read-only register and contains the part ID for each part. This register is also read by the ISP/IAP commands (see Table 21).

**Table 85. Device ID register (DEVICE_ID, address 0x4004 83F8) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 31:0 | DEVICEID | Used to read the part identification number. | part-dependent |

**Table 86. LPC804 device identification numbers**

| Part number | Part ID |
|---|---|
| LPC804M101JBD64 | 0x00008040 |
| LPC804M101JDH20 | 0x00008041 |
| LPC804M101JDH24 | 0x00008042 |
| LPC804M111JDH24 | 0x00008043 |
| LPC804M101JHI33 | 0x00008044 |

## 6.7 Functional description

### 6.7.1 Reset

Reset has the following sources: the $\overline{\text{RESET}}$ pin, Watchdog Reset, Power-On Reset (POR), and Brown Out Detect (BOD). In addition, there is an Arm software reset.

The $\overline{\text{RESET}}$ pin is a Schmitt trigger input pin. Assertion of chip Reset by any source, once the operating voltage attains a usable level, starts the FRO causing reset to remain asserted until the external Reset is de-asserted and the oscillator is running.

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.

### 6.7.2 Brown-out detection

The brown-out detection circuit includes up to three 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 75).

The interrupt signal can be enabled for interrupt in the Interrupt Enable Register in the NVIC (see Table 39) 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 STARTERP1 register (see Table 81) and in the NVIC, the BOD interrupt can wake up the chip from deep-sleep and power-down mode.

If the BOD reset is enabled, the forced BOD reset can wake up the chip from deep-sleep or power-down mode.

## 7.1 How to read this chapter

The ROM-based set fro_oscout API call is available on all parts.

## 7.2 Features

- Select desired on-chip fro_oscout.

## 7.3 General description

Control of FRO output frequency can be configured through a simple call to the ROM.

The set_fro_frequency API call must be used to the select desired fro_oscout (30 MHz/ 24 MHz/18 MHz) and this is divided by two to provide FRO frequencies of 15 MHz, 12 MHz, and 9 MHz.This is performed by executing a function, which is pointed by a pointer within the ROM Driver Table. Figure 10 shows the pointer structure used to call the set FRO frequency API.

**Remark:** Disable all interrupts before making calls to the FRO API. The interrupts can be re-enabled after the FRO API call is completed.

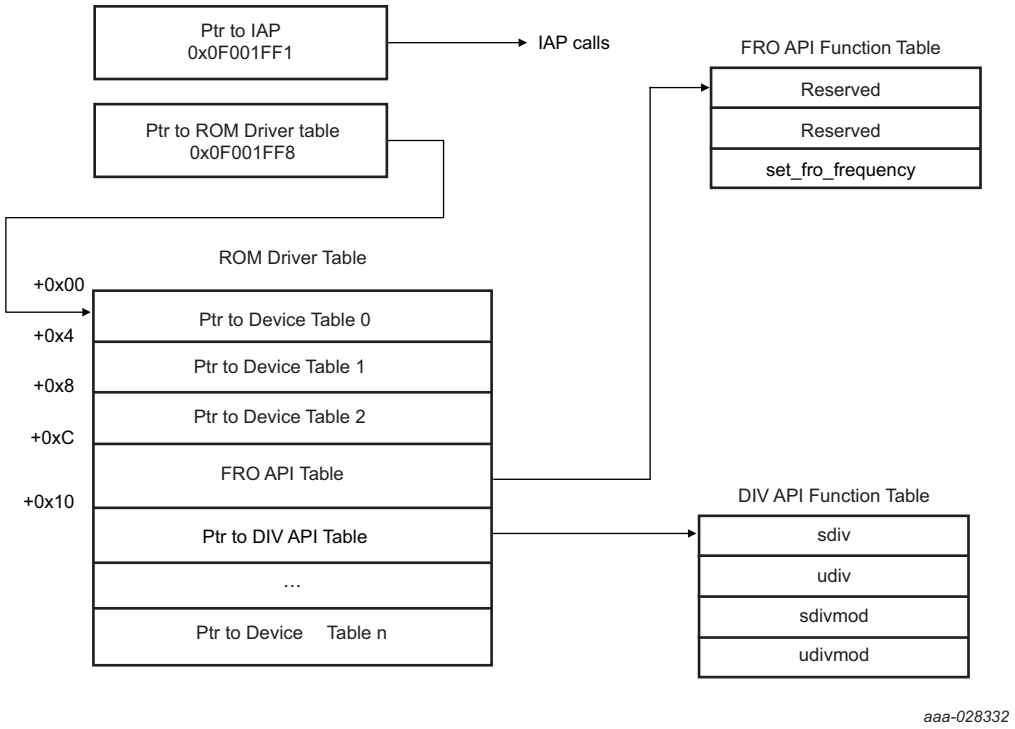



**Fig 10. ROM pointer structure**

## 7.4 API description

The FRO API provides a function to configure the fro_oscout. The FRO API can be called in the application code through a simple API call. An example is provided with the code bundle software package on nxp.com.

The following function prototypes are used:

**Table 87. FRO API call**

| Function prototype | API description | Reference |
|---|---|---|
| `void set_fro_frequency(uint32_t iFreq);` | Setup the fro_oscout to either 30 MHz, 24 MHz or 18 MHz. | 7.4.1 |

### 7.4.1 set_fro_frequency

This routine sets up the fro_oscout (30 MHz/24 MHz/18 MHz) to provide frequencies of 15 MHz, 12 MHz, and 9 MHz. The requested frequency is set up and the appropriate factory trim value is used.

See Figure 9 “LPC804 FRO subsystem” for more details to select FRO frequency.

Table 88 shows the set_fro_frequency routine.

**Table 88. set_fro_frequency routine**

| Routine | sidiv |
|---|---|
| Prototype | void set_fro_frequency(uint32_t iFreq); |
| Input parameter | **Param0 —** Required frequency (in kHz). Example: parameter 1: 18000 => 18MHz, 24000=>24Mhz, 30000=>30Mhz. |
| Return | None. |
| Description | Setup the fro_oscout to either 30 MHz, 24 MHz or 18 MHz. |

#### 7.4.1.1 Param0: frequency

The frequency is the required fro_oscout, 30 MHz, 24 MHz or 18 MHz. The fro_oscout is internally divided by two to provide frequencies of 15 MHz, 12 MHz, and 9 MHz.

## 8.1 How to read this chapter

The switch matrix is identical for all LPC804 parts.

## 8.2 Features

- Flexible assignment of digital peripheral functions to pins.
- Enable/disable of analog functions.
- Provides level shifter functionality to allow up to two selected signals to be routed from user-selected pins in one voltage domain to selected pins in the alternate domain. This feature can also be used on a single supply device if voltage level shifting is not required.

## 8.3 Basic configuration

Once configured, no clocks are needed for the switch matrix to function. The system clock is needed only to write to or read from the pin assignment registers. After the switch matrix is configured, disable the clock to the switch matrix block in the SYSAHBCLKCTRL register.

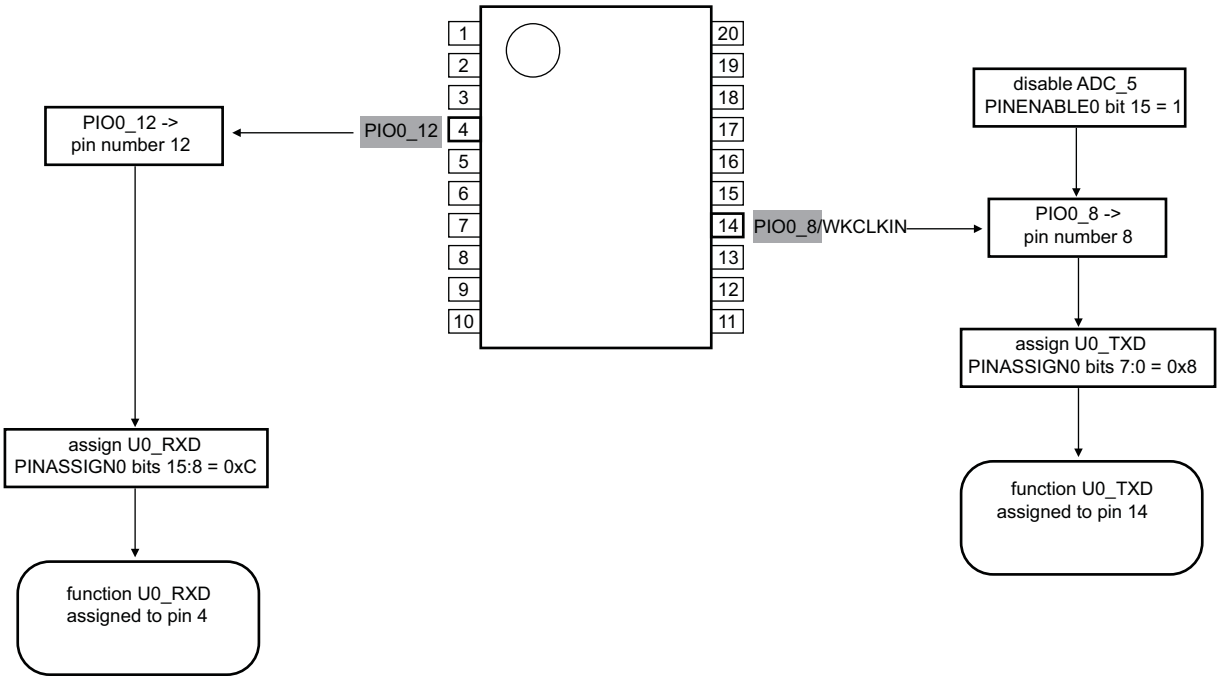
Before activating a peripheral or enabling its interrupt, use the switch matrix to connect the peripheral to external pins.

The serial wire debug pins SWDIO and SWCLK are enabled by default on pins PIO0_2 and PIO0_3.

**Remark:** For the purpose of programming the pin functions through the switch matrix, every pin except the power and ground pins is identified in a package-independent way by its GPIO port pin number.

**Remark:** The switch matrix is reset by a system reset from the $\overline{\text{RESET}}$ pin as well as all other resets.

### 8.3.1 Connect an internal signal to a package pin




A pin is identified for the purpose of programming the switch matrix by its default GPIO port pin number.

**Fig 11. Example: Connect function U0_RXD and U0_TXD to pins 4 and 14**

The switch matrix connects all internal signals listed in the table of movable functions through the pin assignment registers to external pins on the package. External pins are identified by their default GPIO pin number PIO0_n. Follow these steps to connect an internal signal FUNC to an external pin. An example of a movable function is the UART transmit signal TXD:

1. Find the pin function in the list of movable functions in Table 89 or in the data sheet.
2. Use the LPC804 data sheet to decide which pin x on the LPC804 package to connect the pin function to.
3. Use the pin description table to find the default GPIO function PIO0_n assigned to package pin x. m is the pin number.
4. Locate the pin assignment register for the function FUNC in the switch matrix register description.
5. Disable any special functions on pin PIO0_n in the PINENABLE0.
6. Program the pin number n into the bits assigned to the pin function.

The pin function is now connected to pin x on the package.

### 8.3.2 Enable an analog input or other special function

The switch matrix enables functions that can only be assigned to one pin. Examples are analog inputs, all GPIO pins, and the debug SWD pins.

- If you want to assign a GPIO pin to a pin on any LPC804 package, disable any special function available on this pin in the PINENABLE0 register and do not assign any movable function to it.

  By default, all pins except pins PIO0_2, PIO0_3, and PIO0_5 are assigned to GPIO.
- For all other functions that are not in the table of movable functions, do the following:
  a. Locate the function in the pin description table in the data sheet. This shows the package pin for this function.
  b. Enable the function in the PINENABLE0 register. All other possible functions on this pins are now disabled.

### 8.3.3 Changing the pin function assignment

Pin function assignments can be changed “on-the-fly” from one peripheral to another while the part is running. To disconnect a peripheral from the pins and change the pin function assignment, follow these steps:

1. Enable the clock to the switch matrix.
2. Find the pin assign register for the current pin function. For example, register PINASSIGN0 for pin function U0_RXD.
3. Set the corresponding bits in the PINASSIGN register to their default value 0xFF.
4. Clear all pending interrupts for the disconnected peripheral and ensure that the peripheral is in a defined state.
5. In the pin assign register for the new pin function, program the pin number.
6. Disable the clock to the switch matrix.

## 8.4 General description

The switch matrix connects internal signals (functions) to external pins. Functions are signals coming from or going to a single pin on the package and coming from or going to an on-chip peripheral block. Examples of functions are the GPIOs, the UART transmit output (TXD), or the clock output CLKOUT. Many peripherals have several functions that must be connected to external pins.

The switch matrix also enables the output driver for digital functions that are outputs. The electrical pin characteristics for both inputs and outputs (internal pull-up/down resistors, inverter, open-drain mode) are configured by the IOCON block for each pin.

Most functions can be assigned through the switch matrix to any external pin that is not a power or ground pin. These functions are called movable functions.

A few functions like the analog comparator inputs can only be assigned to one particular external pin with the appropriate electrical characteristics. These functions are called fixed-pin functions. If a fixed-pin function is not used, it can be replaced by any other movable function.

For fixed-pin analog functions, the switch matrix enables the analog input or output and disables the digital pad.

GPIOs are special fixed-pin functions. Each GPIO is assigned to one and only one external pin by default. External pins are therefore identified by their fixed-pin GPIO function. The level on a digital input is always reflected in the GPIO port register and in the pin interrupt/pattern match state, if selected, regardless of which (digital) function is assigned to the pin through the switch matrix.

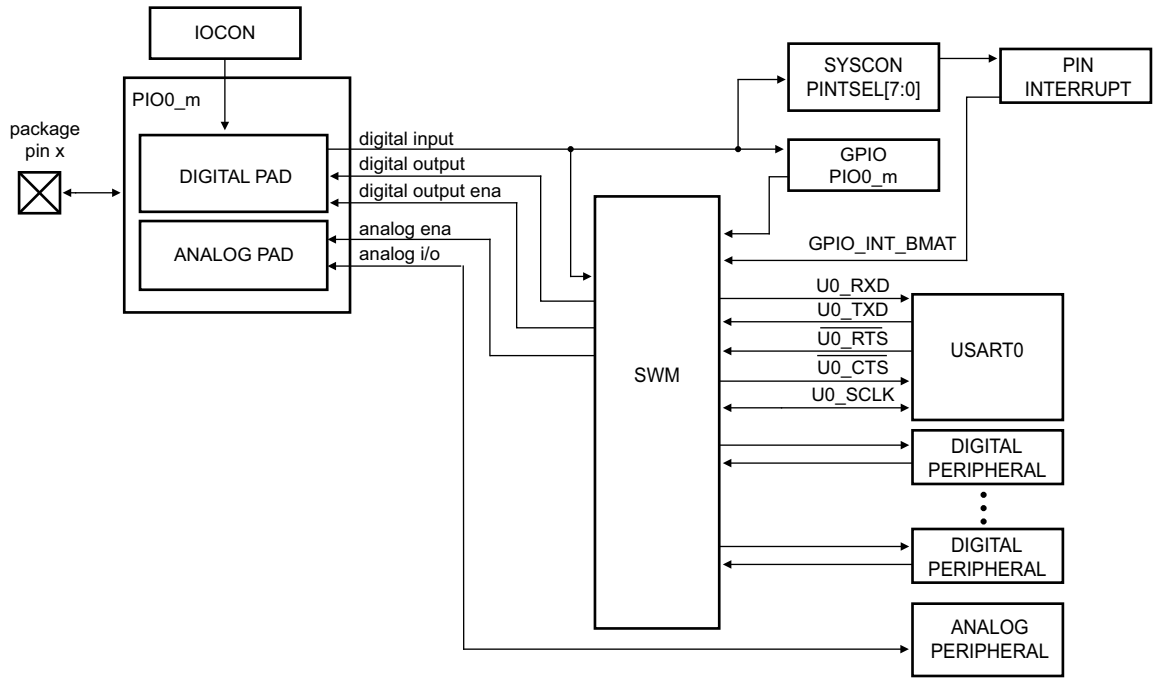



**Fig 12. Functional diagram of the switch matrix**

**Remark:** From all movable and fixed-pin functions, you can assign multiple functions to the same pin but no more than one output or bidirectional function (see Figure 12). Use the following guidelines when assigning pins:

- It is allowed to connect one input signal on a pin to multiple internal inputs by programming the same pin number in more than one PINASSIGN register or programming the corresponding bits in PINASSIGN_FIXED0 register.

  Example:

  You can enable the CLKIN input in the PINENABLE0 register on pin PIO0_1 and also assign other block inputs to pin PIO0_1 through the PINASSIGN registers to feed the CLKIN into the other block.
- It is allowed to let one digital output function control one or more digital inputs by programming the same pin number in the PINASSIGN register bit fields or programming the corresponding bits in PINASSIGN_FIXED0 register for the output and inputs.

  Example:

  You can loop back the USART transmit output to the receive input by assigning the same pin number to Un_RXD and Un_TXD.
- It is not allowed to connect more than one output or bidirectional function to a pin.
- When you assign any function to a pin through the switch matrix, the GPIO output becomes disabled.
- Enabling any analog fixed-pin function disables all digital functions on the same pin.

### 8.4.1 Movable functions

**Table 89. Movable functions (assign to pins PIO0_0 to PIO0_5 and PIO0_7 to PIO0_30 through switch matrix)**

| Function name | Type | Description | SWM Pin assign register | Reference |
|---|---|---|---|---|
| U0_TXD | O | Transmitter output for USART0. | PINASSIGN0 | Table 91 |
| U0_RXD | I | Receiver input for USART0. | PINASSIGN0 | Table 91 |
| $\overline{\text{U0\_RTS}}$ | O | Request To Send output for USART0. | PINASSIGN0 | Table 91 |
| $\overline{\text{U0\_CTS}}$ | I | Clear To Send input for USART0. | PINASSIGN0 | Table 91 |
| U0_SCLK | I/O | Serial clock input/output for USART0 in synchronous mode. | PINASSIGN1 | Table 92 |
| U1_TXD | O | Transmitter output for USART1. | PINASSIGN1 | Table 92 |
| U1_RXD | I | Receiver input for USART1. | PINASSIGN1 | Table 92 |
| U1_SCLK | I/O | Serial clock input/output for USART1 in synchronous mode. | PINASSIGN2 | Table 93 |
| SPI0_SCK | I/O | Serial clock for SPI0. | PINASSIGN2 | Table 93 |
| SPI0_MOSI | I/O | Master Out Slave In for SPI0. | PINASSIGN2 | Table 93 |
| SPI0_MISO | I/O | Master In Slave Out for SPI0. | PINASSIGN2 | Table 93 |
| SPI0_SSEL0 | I/O | Slave select 0 for SPI0. | PINASSIGN2 | Table 93 |
| SPI0_SSEL1 | I/O | Slave select 1 for SPI0. | PINASSIGN3 | Table 94 |
| TO_CAP0 | I | Time Capture channel 0. | PINASSIGN3 | Table 94 |
| TO_CAP1 | I | Time Capture channel 1. | PINASSIGN3 | Table 94 |
| TO_CAP2 | I | Time Capture channel 2. | PINASSIGN3 | Table 94 |

**Table 89. Movable functions (assign to pins PIO0_0 to PIO0_5 and PIO0_7 to PIO0_30 through switch matrix)**

| Function name | Type | Description | SWM Pin assign register | Reference |
|---|---|---|---|---|
| TO_MAT0 | O | Timer Match channel 0. | PINASSIGN4 | Table 95 |
| TO_MAT1 | O | Timer Match channel 1. | PINASSIGN4 | Table 95 |
| TO_MAT2 | O | Timer Match channel 2. | PINASSIGN4 | Table 95 |
| TO_MAT3 | O | Timer Match channel 3. | PINASSIGN4 | Table 95 |
| I2C0_SDA | I/O | $I^2C$0-bus data input/output. | PINASSIGN5 | Table 96 |
| I2C0_SCL | I/O | $I^2C$0-bus clock input/output. | PINASSIGN5 | Table 96 |
| ACMP_O | O | Analog comparator output. | PINASSIGN5 | Table 96 |
| CLKOUT | O | Clock output. | PINASSIGN5 | Table 96 |
| GPIO_INT_BMAT | O | Output of the pattern match engine. | PINASSIGN6 | Table 97 |
| LVLSHFT_IN0 | I | Level shift input 0. | PINASSIGN6 | Table 97 |
| LVLSHFT_IN1 | I | Level shift input 1. | PINASSIGN6 | Table 97 |
| LVLSHFT_OUT0 | O | Level shift output 0. | PINASSIGN6 | Table 97 |
| LVLSHFT_OUT1 | O | Level shift output 1. | PINASSIGN7 | Table 98 |
| I2C1_SDA | I/O | $I^2C$1-bus data input/output. | PINASSIGN7 | Table 98 |
| I2C1_SCL | I/O | $I^2C$1-bus clock input/output. | PINASSIGN7 | Table 98 |
| PLU_CLKIN | I/O | PLU clock input. | PINASSIGN7 | Table 98 |
| CAPT_X0 | O | CAPT_X0 function | PINASSIGN8 | Table 99 |
| CAPT_X1 | O | CAPT_X1 function | PINASSIGN8 | Table 99 |
| CAPT_X2 | O | CAPT_X2 function | PINASSIGN8 | Table 99 |
| CAPT_X3 | O | CAPT_X3 function | PINASSIGN8 | Table 99 |
| CAPT_X4 | O | CAPT_X4 function | PINASSIGN9 | Table 100 |
| CAPT_YL | O | CAPT_YL function | PINASSIGN9 | Table 100 |
| CAPT_YH | I/O | CAPT_YH function | PINASSIGN9 | Table 100 |
| PLU_INPUT0 | I | PLU_INPUT0 function | PINASSIGN_FIXED0 | Table 101 |
| PLU_INPUT1 | I | PLU_INPUT1 function | PINASSIGN_FIXED0 | Table 101 |
| PLU_INPUT2 | I | PLU_INPUT2 function | PINASSIGN_FIXED0 | Table 101 |
| PLU_INPUT3 | I | PLU_INPUT3 function | PINASSIGN_FIXED0 | Table 101 |
| PLU_INPUT4 | I | PLU_INPUT4 function | PINASSIGN_FIXED0 | Table 101 |
| PLU_INPUT5 | I | PLU_INPUT5 function | PINASSIGN_FIXED0 | Table 101 |
| PLU_OUT0 | O | PLU_OUT0 function | PINASSIGN_FIXED0 | Table 101 |
| PLU_OUT1 | O | PLU_OUT1 function | PINASSIGN_FIXED0 | Table 101 |
| PLU_OUT2 | O | PLU_OUT2 function | PINASSIGN_FIXED0 | Table 101 |
| PLU_OUT3 | O | PLU_OUT3 function | PINASSIGN_FIXED0 | Table 101 |
| PLU_OUT4 | O | PLU_OUT4 function | PINASSIGN_FIXED0 | Table 101 |
| PLU_OUT5 | O | PLU_OUT5 function | PINASSIGN_FIXED0 | Table 101 |
| PLU_OUT6 | O | PLU_OUT6 function | PINASSIGN_FIXED0 | Table 101 |
| PLU_OUT7 | O | PLU_OUT7 function | PINASSIGN_FIXED0 | Table 101 |

### 8.4.2 Switch matrix register interface

The switch matrix consists of three blocks of pin-assignment registers PINASSIGN, PINASSIGN_FIXED, and PINENABLE. Every function has an assigned field (1-bit, 2-bit, or 8-bit wide) within this bank of registers where you can program the external pin - identified by its GPIO function - you want the function to connect to.

GPIO0 range from PIO0_0 to PIO0_5 and PIO0_7 to PIO0_30, for assignment through the pin-assignment registers, are numbered 0 to 5 and 7 to 30.

There are two types of functions which must be assigned to port pins in different ways:

1. **Movable functions** (PINASSIGN0 to 9 and PINASSIGN_FIXED0):

   All movable functions are digital functions. Assign movable functions to pin numbers through the 8 bits of the PINASSIGN register or 2 bits of the PINASSIGN_FIXED0 register associated with this function. Once the function is assigned a pin PIO0_n, it is connected through this pin to a physical pin on the package.

   **Remark:** You can assign only one digital output function to an external pin at any given time.

   **Remark:** You can assign more than one digital input function to one external pin.

2. **Fixed-pin functions** (PINENABLE0):

   Some functions require pins with special characteristics and cannot be moved to other physical pins. Hence these functions are mapped to a fixed port pin. Examples of fixed-pin functions are the oscillator pins or comparator inputs.

   Each fixed-pin function is associated with one bit in the PINENABLE0 or register which selects or deselects the function.

   - If a fixed-pin function is deselected, any movable function can be assigned to its port and pin.
   - If a fixed-pin function is deselected and no movable function is assigned to this pin, the pin is assigned its GPIO function.
   - On reset, all fixed-pin functions are deselected, except RESETN. The SWCLK and SWDIO fixed pin functions are enabled by the boot ROM if the CRP setting allows it.
   - If a fixed-pin analog function is selected, its assigned pin cannot be used for any other function.

### 8.4.3 Level Shifter function

The LPC804 provides a dual voltage I/O feature. The pins on one side of the package are supplied by VDDIO and the pins on the other side are supplied by $V_{DD}$. Each of these two supplies can be connected to different voltages within the allowed $V_{DD}$ range. This feature allows the device to level-shift signals from one off-chip voltage domain to another. The switch matrix provides level shifter functionality to allow up to two selected signals to be routed from user-selected pins in one voltage domain to selected pins in the alternate domain. This feature can also be used on a single supply device if voltage level shifting is not required. See Table 97 “Pin assign register 6 (PINASSIGN6, address 0x4000 C018) bit description” and Table 98 “Pin assign register 7 (PINASSIGN7, address 0x4000 C01C) bit description” for details.

## 8.5 Register description

**Table 90. Register overview: Switch matrix (base address 0x4000 C000)**

| Name | Access | Offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| PINASSIGN0 | R/W | 0x000 | Pin assign register 0. Assign movable functions U0_TXD, U0_RXD, U0_RTS, U0_CTS. | 0xFFFF FFFF | Table 91 |
| PINASSIGN1 | R/W | 0x004 | Pin assign register 1. Assign movable functions U0_SCLK, U1_TXD, U1_RXD, U1_SCLK. | 0xFFFF FFFF | Table 92 |
| PINASSIGN2 | R/W | 0x008 | Pin assign register 2. Assign movable functions SPI0_SCK, SPI0_MOSI, SPI0_MISO, SPI0_SSEL0. | 0xFFFF FFFF | Table 93 |
| PINASSIGN3 | R/W | 0x00C | Pin assign register 3. Assign movable function SPI0_SSEL1, T0_CAP0, T0_CAP1, T0_CAP2. | 0xFFFF FFFF | Table 94 |
| PINASSIGN4 | R/W | 0x010 | Pin assign register 4. Assign movable functions T0_MAT0, T0_MAT1, T0_MAT2,T0_MAT3. | 0xFFFF FFFF | Table 95 |
| PINASSIGN5 | R/W | 0x014 | Pin assign register 5. Assign movable functions I2C0_SDA, I2C0_SCL, COMP0_OUT, CLKOUT. | 0xFFFF FFFF | Table 96 |
| PINASSIGN6 | R/W | 0x018 | Pin assign register 6. Assign movable functions GPIO_INT_BMAT, LVLSHFT_IN0, LVLSHFT_IN1, LVLSHFT_OUT0. | 0xFFFF FFFF | Table 97 |
| PINASSIGN7 | R/W | 0x01C | Pin assign register 7. Assign movable functions LVLSHFT_OUT1, I2C1_SDA, I2C1_SCL, PLU_CLKIN. | 0xFFFF FFFF | Table 98 |
| PINASSIGN8 | R/W | 0x020 | Assign movable functions CAPT_X0, CAPT_X1, CAPT_X2, CAPT_X3. | 0xFFFF FFFF | Table 99 |
| PINASSIGN9 | R/W | 0x024 | Assign movable functions CAPT_X4, CAPT_YL, CAPT_YH. | 0xFFFF FFFF | Table 100 |
| PINASSIGN_FIXED0 | R/W | 0x180 | Pin assign to fixed bits register 0. Assign movable functions PLU_INPUT0 to PLU_INPUT5 and PLU_OUT0 to PLU_INPUT7. | 0xFFFF FFFF | Table 101 |
| PINENABLE0 | R/W | 0x1C0 | Pin enable register 0. Enables fixed-pin functions ACMP_I1, ACMP_I2, ACMP_I3, ACMP_I4, SWCLK, SWDIO, RESET, CLKIN, WKCLK_IN, VDDCMP, ADC_0 - ADC_11, DACOUT0. | 0x3FFF8F | Table 102 |

### 8.5.1 Pin assign register 0

**Table 91. Pin assign register 0 (PINASSIGN0, address 0x4000 C000) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 7:0 | U0_TXD_O | U0_TXD function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 15:8 | U0_RXD_I | U0_RXD function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 23:16 | U0_RTS_O | U0_RTS function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_17 (= 0x11). | 0xFF |
| 31:24 | U0_CTS_I | U0_CTS function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |

### 8.5.2 Pin assign register 1

**Table 92. Pin assign register 1 (PINASSIGN1, address 0x4000 C004) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 7:0 | U0_SCLK_IO | U0_SCLK function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E).. | 0xFF |
| 15:8 | U1_TXD_O | U1_TXD function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 23:16 | U1_RXD_I | U1_RXD function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 31:24 | U1_SCLK_IO | U1_SCLK function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |

### 8.5.3 Pin assign register 2

**Table 93. Pin assign register 2 (PINASSIGN2, address 0x4000 C008) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 7:0 | SPI0_SCK_IO | SPI0_SCK function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |

**Table 93. Pin assign register 2 (PINASSIGN2, address 0x4000 C008) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 15:8 | SPI0_MOSI_IO | SPI0_MOSI function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 23:16 | SPI0_MISO_IO | SPI0_MISO function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 31:24 | SPI0_SSEL0_IO | SPI0_SSEL0 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |

### 8.5.4 Pin assign register 3

**Table 94. Pin assign register 3 (PINASSIGN3, address 0x4000 C00C) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 7:0 | SPI0_SSEL1_IO | SPI0_SSEL1 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 15:8 | T0_CAP0 | T0_CAP0 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 23:16 | T0_CAP1 | T0_CAP1 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 31:24 | T0_CAP2 | T0_CAP2 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |

### 8.5.5 Pin assign register 4

**Table 95. Pin assign register 4 (PINASSIGN4, address 0x4000 C010) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 7:0 | T0_MAT0 | T0_MAT0 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 15:8 | T0_MAT1 | T0_MAT1 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 23:16 | T0_MAT2 | T0_MAT2 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 31:24 | T0_MAT3 | T0_MAT3 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |

### 8.5.6 Pin assign register 5

**Table 96. Pin assign register 5 (PINASSIGN5, address 0x4000 C014) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 7:0 | I2C0_SDA_IO | I2C0_SDA function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 15:8 | I2C0_SCL_IO | I2C0_SCL function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 23:16 | COMP0_OUT_O | COMP0_OUT function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 31:24 | CLKOUT_O | CLKOUT function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |

### 8.5.7 Pin assign register 6

**Table 97. Pin assign register 6 (PINASSIGN6, address 0x4000 C018) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 7:0 | GPIO_INT_BMAT_O | GPIO_INT_BMAT function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 15:8 | LVLSHFT_IN0 | LVLSHFT_IN0 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 23:16 | LVLSHFT_IN1 | LVLSHFT_IN1 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 31:24 | LVLSHFT_OUT0 | LVLSHFT_OUT0 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |

### 8.5.8 Pin assign register 7

**Table 98. Pin assign register 7 (PINASSIGN7, address 0x4000 C01C) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 7:0 | LVLSHFT_OUT1 | LVLSHFT_OUT1 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |

**Table 98. Pin assign register 7 (PINASSIGN7, address 0x4000 C01C) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 15:8 | I2C1_SDA_IO | I2C1_SDA function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 23:16 | I2C1_SCL_IO | I2C1_SCL function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 31:24 | PLU_CLKIN_IN | PLU_CLKIN function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | - |

### 8.5.9 Pin assign register 8

**Table 99. Pin assign register 8 (PINASSIGN8, address 0x4000 C020) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 7:0 | CAPT_X0_O | CAPT_X0 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 15:8 | CAPT_X1_O | CAPT_X1 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 23:16 | CAPT_X2_O | CAPT_X2 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 31:24 | CAPT_X3_O | CAPT_X2 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |

### 8.5.10 Pin assign register 9

**Table 100. Pin assign register 9 (PINASSIGN9, address 0x4000 C024) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 7:0 | CAPT_X4_O | CAPT_X4 function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 15:8 | CAPT_YL_O | CAPT_YL function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 23:16 | CAPT_YH_O | CAPT_YH function assignment. The value is the pin number to be assigned to this function. The following pins are available:<br>PIO0_0 (= 0) to PIO0_5 (= 0x5) and from PIO0_7 (= 0x7) to PIO0_30 (= 0x1E). | 0xFF |
| 31:24 | - | Reserved | - |

### 8.5.11 Pin fixed assign register 0

**Table 101. Pin fixed assign register 0 (PINASSIGNFIXED0, address 0x4000 C180) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 1:0 | PLU_INPUT0 | | PLU_INPUT0 function assignment. | 0x3 |
| | | 2’b00 | Assign to PIO0_00. | |
| | | 2’b01 | Assign to PIO0_08. | |
| | | 2’b10 | Assign to PIO0_17. | |
| | | 2’b11 | None. | |
| 3:2 | PLU_INPUT1 | | PLU_INPUT1 function assignment. | 0x3 |
| | | 2’b00 | Assign to PIO0_01. | |
| | | 2’b01 | Assign to PIO0_09. | |
| | | 2’b10 | Assign to PIO0_18. | |
| | | 2’b11 | None. | |
| 5:4 | PLU_INPUT2 | | PLU_INPUT2 function assignment. | 0x3 |
| | | 2’b00 | Assign to PIO0_02. | |
| | | 2’b01 | Assign to PIO0_10. | |
| | | 2’b10 | Assign to PIO0_19. | |
| | | 2’b11 | None. | |
| 7:6 | PLU_INPUT3 | | PLU_INPUT3 function assignment. | 0x3 |
| | | 2’b00 | Assign to PIO0_03. | |
| | | 2’b01 | Assign to PIO0_11. | |
| | | 2’b10 | Assign to PIO0_20. | |
| | | 2’b11 | None. | |
| 9:8 | PLU_INPUT4 | | PLU_INPUT4 function assignment. | 0x3 |
| | | 2’b00 | Assign to PIO0_04. | |
| | | 2’b01 | Assign to PIO0_12. | |
| | | 2’b10 | Assign to PIO0_21. | |
| | | 2’b11 | None. | |
| 11:10 | PLU_INPUT5 | | PLU_INPUT5 function assignment. | 0x3 |
| | | 2’b00 | Assign to PIO0_05. | |
| | | 2’b01 | Assign to PIO0_13. | |
| | | 2’b10 | Assign to PIO0_22. | |
| | | 2’b11 | None. | |
| 13:12 | PLU_OUT0 | | PLU_OUT0 function assignment. | 0x3 |
| | | 2’b00 | Assign to PIO0_07. | |
| | | 2’b01 | Assign to PIO0_14. | |
| | | 2’b10 | Assign to PIO0_23. | |
| | | 2’b11 | None | |

**Table 101. Pin fixed assign register 0 (PINASSIGNFIXED0, address 0x4000 C180) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 15:14 | PLU_OUT1 | | PLU_OUT1 function assignment. | 0x3 |
| | | 2'b00 | Assign to PIO0_08. | |
| | | 2'b01 | Assign to PIO0_15. | |
| | | 2'b10 | Assign to PIO0_24. | |
| | | 2'b11 | None. | |
| 17:16 | PLU_OUT2 | | PLU_OUT2 function assignment. | 0x3 |
| | | 2'b00 | Assign to PIO0_09. | |
| | | 2'b01 | Assign to PIO0_16. | |
| | | 2'b10 | Assign to PIO0_25. | |
| | | 2'b11 | None. | |
| 19:18 | PLU_OUT3 | | PLU_OUT3 function assignment. | 0x3 |
| | | 2'b00 | Assign to PIO0_10. | |
| | | 2'b01 | Assign to PIO0_17. | |
| | | 2'b10 | Assign to PIO0_26. | |
| | | 2'b11 | None. | |
| 21:20 | PLU_OUT4 | | PLU_OUT4 function assignment. | 0x3 |
| | | 2'b00 | Assign to PIO0_11. | |
| | | 2'b01 | Assign to PIO0_18. | |
| | | 2'b10 | Assign to PIO0_27. | |
| | | 2'b11 | None. | |
| 23:22 | PLU_OUT5 | | PLU_OUT5 function assignment. | 0x3 |
| | | 2'b00 | Assign to PIO0_12. | |
| | | 2'b01 | Assign to PIO0_19. | |
| | | 2'b10 | Assign to PIO0_28. | |
| | | 2'b11 | None. | |
| 25:24 | PLU_OUT6 | | PLU_OUT6 function assignment. | 0x3 |
| | | 2'b00 | Assign to PIO0_13. | |
| | | 2'b01 | Assign to PIO0_20. | |
| | | 2'b10 | Assign to PIO0_29. | |
| | | 2'b11 | None. | |
| 27:26 | PLU_OUT7 | | PLU_OUT7 function assignment. | 0x3 |
| | | 2'b00 | Assign to PIO0_14. | |
| | | 2'b01 | Assign to PIO0_21. | |
| | | 2'b10 | Assign to PIO0_30. | |
| | | 2'b11 | None. | |
| 31:28 | - | - | Reserved. | - |

### 8.5.12 PINENABLE 0

**Table 102. Pin enable register 0 (PINENABLE0, address 0x4000 C1C0) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | ACMP_I1 | | ACMP_I1 function select. | 1 |
| | | 0 | ACMP_I1 enabled on pin PIO0_00. | |
| | | 1 | ACMP_I1 disabled. | |
| 1 | ACMP_I2 | | ACMP_I2 function select. | 1 |
| | | 0 | ACMP_I2 enabled on pin PIO0_1. | |
| | | 1 | ACMP_I2 disabled. | |
| 2 | ACMP_I3 | | ACMP_I3 function select. | 1 |
| | | 0 | ACMP_I3 enabled on pin PIO0_14. | |
| | | 1 | ACMP_I3 disabled. | |
| 3 | ACMP_I4 | | ACMP_I4 function select. | 1 |
| | | 0 | ACMP_I4 enabled on pin PIO0_16. | |
| | | 1 | ACMP_I4 disabled. | |
| 4 | SWCLK | | SWCLK function select. | 0 |
| | | 0 | SWCLK enabled on pin PIO0_3. | |
| | | 1 | SWCLK disabled. | |
| 5 | SWDIO | | SWDIO function select. | 0 |
| | | 0 | SWDIO enabled on pin PIO0_2. | |
| | | 1 | SWDIO disabled. | |
| 6 | RESETN | | RESETN function select. | 0 |
| | | 0 | RESETN enabled on pin PIO0_5. | |
| | | 1 | RESETN disabled. | |
| 7 | CLKIN | | CLKIN function select. | 1 |
| | | 0 | CLKIN enabled on pin PIO0_1. | |
| | | 1 | CLKIN disabled. | |
| 8 | WKCLKIN | | WKCLK_IN function select. | 1 |
| | | 0 | WKCLK_IN enabled on pin PIO0_11. | |
| | | 1 | WKCLK_IN disabled. | |
| 9 | VDDCMP | | VDDCMP function select. | 1 |
| | | 0 | VDDCMP enabled on pin PIO0_7. | |
| | | 1 | VDDCMP disabled. | |
| 10 | ADC_0 | | ADC_0 function select. | 1 |
| | | 0 | ADC_0 enabled on pin PIO0_1. | |
| | | 1 | ADC_0 disabled. | |
| 11 | ADC_1 | | ADC_1 function select. | 1 |
| | | 0 | ADC_1 enabled on pin PIO0_7. | |
| | | 1 | ADC_1 disabled. | |
| 12 | ADC_2 | | ADC_2 function select. | 1 |
| | | 0 | ADC2 enabled on pin PIO0_14. | |
| | | 1 | ADC2 disabled. | |

**Table 102. Pin enable register 0 (PINENABLE0, address 0x4000 C1C0) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 13 | ADC_3 | | ADC_3 function select. | 1 |
| | | 0 | ADC_3 enabled on pin PIO0_16. | |
| | | 1 | ADC_3 disabled. | |
| 14 | ADC_4 | | ADC_4 function select. | 1 |
| | | 0 | ADC_4 enabled on pin PIO0_9. | |
| | | 1 | ADC_4 disabled. | |
| 15 | ADC_5 | | ADC_5 function select. | 1 |
| | | 0 | ADC_5 enabled on pin PIO0_8. | |
| | | 1 | ADC_5 disabled. | |
| 16 | ADC_6 | | ADC_6 function select. | 1 |
| | | 0 | ADC_6 enabled on pin PIO0_11. | |
| | | 1 | ADC_6 disabled. | |
| 17 | ADC_7 | | ADC_7 function select. | 1 |
| | | 0 | ADC_7 enabled on pin PIO0_10. | |
| | | 1 | ADC_7 disabled. | |
| 18 | ADC_8 | | ADC_8 function select. | 1 |
| | | 0 | ADC_8 enabled on pin PIO0_15. | |
| | | 1 | ADC_8 disabled. | |
| 19 | ADC_9 | | ADC_9 function select. | 1 |
| | | 0 | ADC_9 enabled on pin PIO0_17. | |
| | | 1 | ADC_9 disabled. | |
| 20 | ADC_10 | | ADC_10 function select. | 1 |
| | | 0 | ADC_10 enabled on pin PIO0_13. | |
| | | 1 | ADC_10 disabled. | |
| 21 | ADC_11 | | ADC_11 function select. | 1 |
| | | 0 | ADC_11 enabled on pin PIO0_4. | |
| | | 1 | ADC_11 disabled. | |
| 22 | ACMP_15 | | ACMP_15 function select. | 1 |
| | | 0 | ACMP_I5 enabled on pin PIO0_21. | |
| | | 1 | ACMP_I5 disabled. | |
| 23 | DACOUT0 | | DACOUT0 function select. | 1 |
| | | 0 | DACOUT0 enabled on pin PIO0_19. | |
| | | 1 | DACOUT0 disabled. | |
| 31:24 | - | - | Reserved. | - |

**Remark:** In analog mode, the internal pull-up must be disabled via the IOCON register.

## 9.1 How to read this chapter

The IOCON block is identical for all LPC804 parts. Registers for pins that are not available on a specific package are reserved.

**Table 103. Pinout summary**

| Package | Pins/configuration registers available |
|---|---|
| TSSOP20 | PIO0_0 to PIO0_5, PIO0_7 to PIO0_17. |
| TSSOP24 (Single Supply) | PIO0_0 to PIO0_5, PIO0_7 to PIO0_21. |
| TSSOP24 (Dual Supply) | PIO0_0 to PIO0_5, PIO0_7 to PIO0_16, PIO0_18 to PIO0_21. |
| HVQFN33 | PIO0_0 to PIO0_5, PIO0_7 to PIO0_30. |

## 9.2 Features

The following electrical properties are configurable for each pin:

- Pull-up/pull-down resistor.
- Open-drain mode.
- Hysteresis.
- Analog mode (for a subset of pins, see the LPC804 data sheet).

Two I2Cs support data rates up to 400 kbit/s on standard digital pins.

## 9.3 Basic configuration

Enable the clock to the IOCON in the SYSAHBCLKCTRL register (Table 64, bit 18). Once the pins are configured, you can disable the IOCON clock to conserve power.

## 9.4 General description

### 9.4.1 Pin configuration

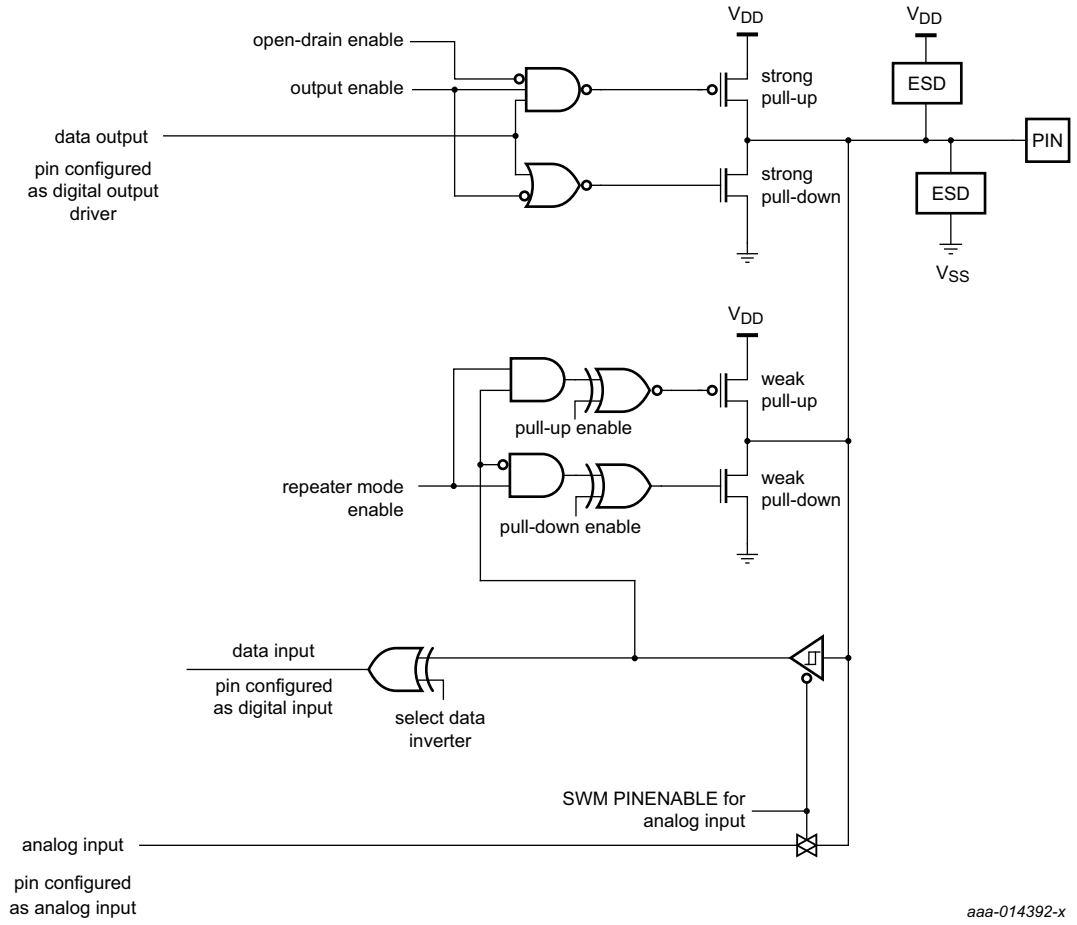



**Fig 13. Pin configuration**

### 9.4.2 Pin function

The pin function is determined entirely through the switch matrix. By default one of the GPIO functions is assigned to each pin. The switch matrix can assign all functions from the movable function table to any pin in the IOCON block or enable a special function like an analog input on a specific pin.

Related links:

Table 89 “Movable functions (assign to pins PIO0_0 to PIO0_5 and PIO0_7 to PIO0_30 through switch matrix)”

### 9.4.3 Pin mode

The MODE bit in the IOCON register allows enabling or disabling an on-chip pull-up resistor for each pin.

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. 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.

### 9.4.4 Open-drain mode

An open-drain mode can be enabled for all digital I/O pins. This mode is not a true open-drain mode. The input cannot be pulled up above $V_{DD}$.

**Remark:** As opposed to the true open-drain I2C-bus pins, digital pins with configurable open-drain mode are **not** 5 V tolerant when $V_{DD}$ = 0.

### 9.4.5 Analog mode

The switch matrix automatically configures the pin in analog mode whenever an analog input or output is selected as the pin's function. In analog mode, the internal pull-up must be disabled via the IOCON register.

## 9.5 Register description

Each port pin PIO0_m has one IOCON register assigned to control the pin's function and electrical characteristics.

**Remark:** See Table 105 for the IOCON register map ordered by pin name.

**Table 104. Register overview: I/O configuration (base address 0x4004 4000)**

| Name | Access | Address offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| PIO0_17 | R/W | 0x000 | I/O configuration for pin PIO0_17/ADC_9. | 0x0000 00B0 | Table 106 |
| PIO0_13 | R/W | 0x004 | I/O configuration for pin PIO0_13/ADC_10. | 0x0000 00B0 | Table 107 |
| PIO0_12 | R/W | 0x008 | I/O configuration for pin PIO0_12. | 0x0000 00B0 | Table 108 |
| PIO0_5 | R/W | 0x00C | I/O configuration for pin PIO0_5/$\overline{\text{RESET}}$ | 0x0000 00B0 | Table 109 |
| PIO0_4 | R/W | 0x010 | I/O configuration for pin PIO0_4/ADC_11/TRSTN | 0x0000 00B0 | Table 110 |
| PIO0_3 | R/W | 0x014 | I/O configuration for pin PIO0_3/SWCLK | 0x0000 00B0 | Table 111 |
| PIO0_2 | R/W | 0x018 | I/O configuration for pin PIO0_2/SWDIO | 0x0000 00B0 | Table 112 |
| PIO0_11 | R/W | 0x01C | I/O configuration for pin PIO0_11/ADC_6/WKTCLKIN. | 0x0000 00B0 | Table 113 |
| PIO0_10 | R/W | 0x020 | I/O configuration for pin PIO0_10/ADC_7 | 0x0000 00B0 | Table 114 |
| PIO0_16 | R/W | 0x024 | I/O configuration for pin PIO0_16/ADC_3/ACMP_14 | 0x0000 00B0 | Table 115 |
| PIO0_15 | R/W | 0x028 | I/O configuration for pin PIO0_15/ADC_8 | 0x0000 00B0 | Table 116 |
| PIO0_1 | R/W | 0x02C | I/O configuration for pin PIO0_1/ADC_0/ACMP_I2/CLKIN | 0x0000 00B0 | Table 117 |
| PIO0_21 | R/W | 0x030 | I/O configuration for pin PIO0_21/ACMP15. | 0x000 00B0 | Table 130 |
| PIO0_9 | R/W | 0x034 | I/O configuration for pin PIO0_9/ADC_4 | 0x0000 00B0 | Table 118 |
| PIO0_8 | R/W | 0x038 | I/O configuration for pin PIO0_8/ADC_5 | 0x0000 00B0 | Table 119 |

**Table 104. Register overview: I/O configuration (base address 0x4004 4000)**

| Name | Access | Address offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| PIO0_7 | R/W | 0x03C | I/O configuration for pin PIO0_7/ADC_0/VDDCMP. | 0x0000 00B0 | Table 120 |
| PIO0_29 | R/W | 0x040 | I/O configuration for pin PIO0_29. | 0x000 00B0 | Table 134 |
| PIO0_0 | R/W | 0x044 | I/O configuration for pin PIO0_0/ACMP_I1. | 0x0000 00B0 | Table 121 |
| PIO0_14 | R/W | 0x048 | I/O configuration for pin PIO0_14/ ACMP_I3/ADC_2 | 0x0000 00B0 | Table 122 |
| PIO0_28 | R/W | 0x04C | I/O configuration for pin PIO0_28. | 0x000 00B0 | Table 123 |
| PIO0_27 | R/W | 0x050 | I/O configuration for pin PIO0_27. | 0x000 00B0 | Table 124 |
| PIO0_26 | R/W | 0x054 | I/O configuration for pin PIO0_26. | 0x000 00B0 | Table 125 |
| PIO0_20 | R/W | 0x058 | I/O configuration for pin PIO0_20. | 0x000 00B0 | Table 131 |
| PIO0_30 | R/W | 0x05C | I/O configuration for pin PIO0_30. | 0x000 00B0 | Table 135 |
| PIO0_19 | R/W | 0x060 | I/O configuration for pin PIO0_19/DACOUT. | 0x000 00B0 | Table 132 |
| PIO0_25 | R/W | 0x064 | I/O configuration for pin PIO0_25. | 0x000 00B0 | Table 126 |
| PIO0_24 | R/W | 0x068 | I/O configuration for pin PIO0_24. | 0x000 00B0 | Table 127 |
| PIO0_23 | R/W | 0x06C | I/O configuration for pin PIO0_23. | 0x000 00B0 | Table 128 |
| PIO0_22 | R/W | 0x070 | I/O configuration for pin PIO0_22. | 0x000 00B0 | Table 129 |
| PIO0_18 | R/W | 0x074 | I/O configuration for pin PIO0_18. | 0x000 00B0 | Table 133 |

**Table 105. I/O configuration registers ordered by pin name**

| Name | Address offset | True open-drain | Analog[1] | High-drive output | Reference |
|---|---|---|---|---|---|
| PIO0_0 | 0x044 | no | yes | no | Table 121 |
| PIO0_1 | 0x02C | no | yes | no | Table 117 |
| PIO0_2 | 0x018 | no | no | yes | Table 112 |
| PIO0_3 | 0x014 | no | no | yes | Table 111 |
| PIO0_4 | 0x010 | no | yes | no | Table 110 |
| PIO0_5 | 0x00C | no | no | no | Table 109 |
| PIO0_7 | 0x03C | no | yes | no | Table 120 |
| PIO0_8 | 0x038 | no | yes | no | Table 119 |
| PIO0_9 | 0x034 | no | yes | no | Table 118 |
| PIO0_10 | 0x020 | no | yes | no | Table 114 |
| PIO0_11 | 0x01C | no | yes | no | Table 113 |
| PIO0_12 | 0x008 | no | no | yes | Table 108 |
| PIO0_13 | 0x004 | no | yes | no | Table 107 |
| PIO0_14 | 0x048 | no | yes | no | Table 122 |
| PIO0_15 | 0x028 | no | yes | no | Table 116 |
| PIO0_16 | 0x024 | no | yes | no | Table 115 |
| PIO0_17 | 0x000 | no | yes | no | Table 106 |
| PIO0_18 | 0x074 | no | no | yes | Table 133 |
| PIO0_19 | 0x060 | no | yes | no | Table 132 |
| PIO0_20 | 0x058 | no | no | yes | Table 131 |
| PIO0_21 | 0x030 | no | yes | no | Table 130 |

**Table 105. I/O configuration registers ordered by pin name**

| Name | Address offset | True open-drain | Analog[1] | High-drive output | Reference |
|---|---|---|---|---|---|
| PIO0_22 | 0x070 | no | no | no | Table 129 |
| PIO0_23 | 0x06C | no | no | no | Table 128 |
| PIO0_24 | 0x068 | no | no | no | Table 127 |
| PIO0_25 | 0x064 | no | no | no | Table 126 |
| PIO0_26 | 0x054 | no | no | no | Table 125 |
| PIO0_27 | 0x050 | no | no | no | Table 124 |
| PIO0_28 | 0x04C | no | no | no | Table 123 |
| PIO0_29 | 0x040 | no | no | no | Table 134 |
| PIO0_30 | 0x05C | no | no | no | Table 135 |

[1] The analog pad is enabled when the analog function is selected in the switch matrix through the PINENABLE register.

Table 106 applies to pins PIO0[17 to 7] and PIO0[5 to 0].

### 9.5.1 PIO0_17 register

**Table 106. PIO0_17 register (PIO0_17, address 0x4004 4000) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.2 PIO0_13 register

**Table 107. PIO0_13 register (PIO0_13, address 0x4004 4004) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.3 PIO0_12 register

**Table 108. PIO0_12 register (PIO0_12, address 0x4004 4008) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |

**Table 108. PIO0_12 register (PIO0_12, address 0x4004 4008) bit description** *...continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.4 PIO0_5 register

**Table 109. PIO0_5 register (PIO0_5, address 0x4004 400C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.5 PIO0_4 register

**Table 110. PIO0_4 register (PIO0_4, address 0x4004 4010) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |

**Table 110. PIO0_4 register (PIO0_4, address 0x4004 4010) bit description** *...continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.6 PIO0_3 register

**Table 111. PIO0_3 register (PIO0_3, address 0x4004 4014) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input. | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.7 PIO0_2 register

**Table 112. PIO0_2 register (PIO0_2, address 0x4004 4018) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input. | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.8 PIO0_11 register

**Table 113. PIO0_11 register (PIO0_11, address 0x4004 401C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |

**Table 113. PIO0_11 register (PIO0_11, address 0x4004 401C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 10 | OD | - | Open-drain mode | 0 |
| | | 0 | Disable | |
| | | 1 | Open-drain mode enabled. | |
| 31:11 | - | - | Reserved. | - |

### 9.5.9 PIO0_10 register

**Table 114. PIO0_10 register (PIO0_10, address 0x4004 4020) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | - | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | - | Open-drain mode | 0 |
| | | 0 | Disable | |
| | | 1 | Open-drain mode enabled. | |
| 31:11 | - | - | Reserved. | - |

### 9.5.10 PIO0_16 register

**Table 115. PIO0_16 register (PIO0_16, address 0x4004 4024) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |

**Table 115. PIO0_16 register (PIO0_16, address 0x4004 4024) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.11 PIO0_15 register

**Table 116. PIO0_15 register (PIO0_15, address 0x4004 4028) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 31:7 | - | - | Reserved. | - |

### 9.5.12 PIO0_1 register

**Table 117. PIO0_1 register (PIO0_1, address 0x4004 402C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |

**Table 117. PIO0_1 register (PIO0_1, address 0x4004 402C) bit description** *...continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.13 PIO0_9 register

**Table 118. PIO0_9 register (PIO0_9, address 0x4004 4034) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.14 PIO0_8 register

**Table 119. PIO0_8 register (PIO0_8, address 0x4004 4038) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |

**Table 119. PIO0_8 register (PIO0_8, address 0x4004 4038) bit description** *...continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.15 PIO0_7 register

**Table 120. PIO0_7 register (PIO0_7, address 0x4004 403C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.16 PIO0_0 register

**Table 121. PIO0_0 register (PIO0_0, address 0x4004 4044) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.17 PIO0_14 register

**Table 122. PIO0_14 register (PIO0_14, address 0x4004 4048) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |

**Table 122. PIO0_14 register (PIO0_14, address 0x4004 4048) bit description** *...continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.18 PIO0_28 register

**Table 123. PIO0_28 register (PIO0_28, address 0x4004 404C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.19 PIO0_27 register

**Table 124. PIO0_27 register (PIO0_27, address 0x4004 4050) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |

**Table 124. PIO0_27 register (PIO0_27, address 0x4004 4050) bit description** *...continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.20 PIO0_26 register

**Table 125. PIO0_26 register (PIO0_26, address 0x4004 4054) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.21 PIO0_25 register

**Table 126. PIO0_25 register (PIO0_25, address 0x4004 4064) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |

**Table 126. PIO0_25 register (PIO0_25, address 0x4004 4064) bit description** *...continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.22 PIO0_24 register

**Table 127. PIO0_24 register (PIO0_24, address 0x4004 4068) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.23 PIO0_23 register

**Table 128. PIO0_23 register (PIO0_23, address 0x4004 406C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.24 PIO0_22 register

**Table 129. PIO0_22 register (PIO0_22, address 0x4004 4070) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |

**Table 129. PIO0_22 register (PIO0_22, address 0x4004 4070) bit description** *...continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.25 PIO0_21 register

**Table 130. PIO0_21 register (PIO0_21, address 0x4004 4030) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.26 PIO0_20 register

**Table 131. PIO0_20 register (PIO0_20, address 0x4004 4058) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |

**Table 131. PIO0_20 register (PIO0_20, address 0x4004 4058) bit description** *...continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.27 PIO0_19 register

**Table 132. PIO0_19 register (PIO0_19, address 0x4004 4060) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 15:11 | - | - | Reserved. | 0 |
| 16 | DACMODE | | DAC mode enable | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 31:17 | - | - | Reserved. | 0 |

### 9.5.28 PIO0_18 register

**Table 133. PIO0_18 register (PIO0_18, address 0x4004 4074) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.29 PIO0_29 register

**Table 134. PIO0_29 register (PIO0_29, address 0x4004 4040) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |

**Table 134. PIO0_29 register (PIO0_29, address 0x4004 4040) bit description** *...continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

### 9.5.30 PIO0_30 register

**Table 135. PIO0_30 register (PIO0_30, address 0x4004 405C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. | 0 |
| 4:3 | MODE | | Selects function mode (on-chip pull-up/pull-down resistor control). | 0b10 |
| | | 0x0 | Inactive (no pull-down/pull-up resistor enabled). | |
| | | 0x1 | Pull-down resistor enabled. | |
| | | 0x2 | Pull-up resistor enabled. | |
| | | 0x3 | Repeater mode. | |
| 5 | HYS | | Hysteresis. | 1 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 6 | INV | | Invert input | 0 |
| | | 0 | Input not inverted (HIGH on pin reads as 1; LOW on pin reads as 0). | |
| | | 1 | Input inverted (HIGH on pin reads as 0, LOW on pin reads as 1). | |
| 9:7 | - | - | Reserved. | 0b001 |
| 10 | OD | | Open-drain mode. | 0 |
| | | 0 | Disable. | |
| | | 1 | Open-drain mode enabled.<br>**Remark:** This is not a true open-drain mode. | |
| 31:11 | - | - | Reserved. | 0 |

## 10.1 How to read this chapter

GPIO pins are placed in logical groups by port. A port may contain up to 32 pins. Typically, these pins have other functions. See Chapter 10 “LPC804 General Purpose I/O (GPIO)” for more details. Table 136 shows the GPIO pins that are available for each package type.

**Table 136. GPIO pins available**

| Package | GPIO Port 0 |
|---|---|
| TSSOP20 | PIO0_0 to PIO0_5, PIO0_7 to PIO0_17. |
| TSSOP24 (Single Supply) | PIO0_0 to PIO0_5, PIO0_7 to PIO0_21. |
| TSSOP24 (Dual Supply) | PIO0_0 to PIO0_5, PIO0_7 to PIO0_16, PIO0_18 to PIO0_21. |
| HVQFN33 | PIO0_0 to PIO0_5, PIO0_7 to PIO0_30. |

## 10.2 Basic configuration

For the GPIO port registers, enable the clock to the GPIO port in the SYSAHBCLKCTRL register (Table 64).

## 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.
- Direction (input/output) can be set and cleared 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 block.

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

**Remark:** The reset value of external in Table 137 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 137. Register overview: GPIO port (base address 0xA000 0000)**

| Name | Access | Address offset | Description | Reset value | Width | Reference |
|---|---|---|---|---|---|---|
| B0 to B5 | R/W | 0x0000 to 0x0005 | Byte pin registers port 0: pins PIO0_0 to PIO0_5. | external | byte (8 bit) | Table 138 |
| B7 to B30 | R/W | 0x0007 to 0x001E | Byte pin registers port 0: pins PIO0_7 to PIO0_30[1] | external | byte (8 bit) | Table 138 |
| - | - | 0x001F to 0x003F | Reserved | - | - | - |
| W0 to W5 | R/W | 0x1000 to 0x1014 | Word pin registers port 0 | external | word (32 bit) | Table 139 |
| W7 to W30 | R/W | 0x101C to 0x1078 | Word pin registers port 0 | external | word (32 bit) | Table 139 |
| DIR0 | R/W | 0x2000 | Direction registers port 0 | 0 | word (32 bit) | Table 140 |
| MASK0 | R/W | 0x2080 | Mask register port 0 | 0 | word (32 bit) | Table 141 |
| PIN0 | R/W | 0x2100 | Port pin register port 0 | external | word (32 bit) | Table 142 |
| MPIN0 | R/W | 0x2180 | Masked port register port 0 | external | word (32 bit) | Table 143 |
| SET0 | R/W | 0x2200 | Write: Set register for port 0<br>Read: output bits for port 0 | 0 | word (32 bit) | Table 144 |
| CLR0 | WO | 0x2280 | Clear port 0 | - | word (32 bit) | Table 145 |
| NOT0 | WO | 0x2300 | Toggle port 0 | - | word (32 bit) | Table 146 |
| DIRSET0 | WO | 0x2380 | Set pin direction bits for port 0. | 0 | word (32 bit) | Table 147 |
| DIRCLR0 | WO | 0x2400 | Clear pin direction bits for port 0. | - | word (32 bit) | Table 148 |
| DIRNOT0 | WO | 0x2480 | Toggle pin direction bits for port 0. | - | word (32 bit) | Table 149 |

[1] PIO0_6 is not available for all registers. PIO0_0 to PIO0_5 and PIO0_7 to PIO0_17 are available for all registers. For LPC802M011JDH20, PIO0_0 to PIO0_5 and PIO0_7 to PIO0_16 are available for all registers.

### 10.5.1 GPIO port byte pin registers

Each GPIO pin has a byte register in this address range. Bytes read in this range will be 0 if the pin is LOW or 0x01 if the pin is HIGH, regardless of direction, masking, or alternate function, except that pins configured as analog I/O always read as zeros. Writes will set or clear the pins output bit based on bit 0 of the byte written. 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 138. GPIO port byte pin registers (B[0:30], addresses 0xA000 0000 (B0) to 0xA000 001E (B30)) bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| 0 | PBYTE | Read: state of the pin PIO0_n, at byte offset n, regardless of direction, masking, or alternate function, except that pins configured as analog I/O always read as 0.<br>Write: loads the output bit of the pin. | external | 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 output bit of the pin if the value written is all zeros, otherwise, it will set the else it will set the output bit of the pin.

**Table 139. GPIO port word pin registers (W[0:30], addresses 0xA000 1000 (W0) to 0xA000 1078 (W30)) bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| | | | | |
| 31:0 | PWORD | Word for pin PIO0_n is access at offset n. Read 0: pin PIO0_n is LOW.<br>Write 0: clear output bit.<br>Read 0xFFFF FFFF: pin PIO0_n is HIGH.<br>Write any value 0x0000 0001 to 0xFFFF FFFF: set output bit.<br>**Remark:** Only 0 or 0xFFFF FFFF can be read. Writing any value other than 0 will set the output bit. | external | 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 140. GPIO direction port register (DIR0), address 0xA000 2000 bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| 30:0 | DIRP | Selects pin direction for pin PIO0_n (bit 0 = PIO0_0, bit 1 = PIO0_1, ..., bit 30 = PIO0_30).<br>0 = input.<br>1 = output. | 0 | R/W |
| 31 | - | Reserved | 0 | - |

### 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. GPIO mask port register (MASK0), address 0xA000 2080 bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| 30:0 | MASKP | Controls which bits corresponding to PIO0_n are active in the MPORT register (bit 0 = PIO0_0, bit 1 = PIO_1, ..., bit 30 = PIO0_30).<br>0 = Read MPORT: pin state; write MPORT: load output bit.<br>1 = Read MPORT: 0; write MPORT: output bit not affected. | 0 | R/W |
| 31 | - | Reserved | 0 | - |

### 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 142. GPIO port pin register (PIN0), address 0xA000 2100 bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| 30:0 | PORT | Reads pin states or loads output bits (bit 0 = PIO0_0, bit 1 = PIO0_1, ..., bit 30 = PIO0_30).<br>0 = Read: pin is low; write: clear output bit.<br>1 = Read: pin is high; write: set output bit. | external | R/W |
| 31 | - | Reserved | 0 | - |

### 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 143. GPIO masked port pin register (MPIN0), address 0xA000 2180 bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| 30:0 | MPORTP | Masked port register (bit 0 = PIO0_0, bit 1 =PIO0_1, ..., bit 30 = PIO0_30).<br>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.<br>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. | external | R/W |
| 31 | - | Reserved | 0 | - |

### 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 144. GPIO port set register (SET0), address 0xA000 2200 bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| 30:0 | SETP | Read or set output bits (bit 0 = PIO0_0,<br>bit 1 = PIO0_1, ..., bit 30 = PIO0_30).<br>0 = Read: output bit: write: no operation.<br>1 = Read: output bit; write: set output bit. | 0 | R/W |
| 31 | - | Reserved | 0 | - |

### 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 145. GPIO port clear register (CLR0), address 0xA000 2280 bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| 30:0 | CLRP | Clear output bits (bit 0 = PIO0_0, bit 1 = PIO0_1, ..., bit 30 = PIO0_30).<br>0 = No operation.<br>1 = Clear output bit. | NA | WO |
| 31 | -- | Reserved | 0 | - |

### 10.5.9 GPIO port toggle registers

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

**Table 146. GPIO port toggle register (NOT0), address 0xA000 2300 bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| 30:0 | NOTP | Toggle output bits (bit 0 = PIO0_0, bit 1 = PIO0_1, ..., bit 30 = PIO0_30).<br>0 = no operation.<br>1 = Toggle output bit. | NA | WO |
| 31 | - | Reserved. | 0 | - |

### 10.5.10 GPIO port direction set registers

Direction bits can be set by writing ones to these registers.

**Table 147. GPIO port direction set register (DIRSET0), address 0xA000 2380 bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| 30:0 | DIRSETP | Set direction bits (bit 0 = PIO0_0, bit 1 = PIO0_1, ..., bit 30 = PIO0_30).<br>0 = No operation.<br>1 = Set direction bit. | 0 | WO |
| 31 | - | Reserved. | 0 | - |

### 10.5.11 GPIO port direction clear registers

Direction bits can be cleared by writing ones to these write-only registers.

**Table 148. GPIO port direction clear register (DIRCLR0), 0xA000 2400 bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| 30:0 | DIRCLRP | Clear direction bits (bit 0 = PIO0_0,<br>bit 1 = PIO0_1, ..., bit 30 = PIO0_30).<br>0 = No operation.<br>1 = Clear direction bit. | NA | WO |
| 31 | - | Reserved. | 0 | - |

### 10.5.12 GPIO port direction toggle registers

Direction bits can be set by writing ones to these write-only registers.

**Table 149. GPIO port direction toggle register (DIRNOT0), address 0xA000 2480 bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| 30:0 | DIRNOTP | Toggle direction bits (bit 0 = PIO0_0, bit 1 = PIO0_1, ...,<br>bit 30 = PIO0_30).<br>0 = no operation.<br>1 = Toggle direction bit. | NA | WO |
| 31 | - | Reserved | 0 | - |

## 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 from bit 0 of a Byte Pin register (bits 7:1 are always zero).
- 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 in the switch matrix (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 multiple 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 Section 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 GPIO direction

Each pin in a GPIO port can be configured as input or output using the DIR registers. The direction of individual pins can be set, cleared, or toggled using the DIRSET, DIRCLR, and DIRNOT registers.

### 10.6.5 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 LPC804 parts.

## 11.2 Features

- Pin interrupts
  - Up to eight pins can be selected from all GPIO pins 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 eight pins can be selected from all GPIO pins 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 Arm 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 GPIO port pins in the SYSCON block (Table 79). 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 SYSAHBCLKCTRL register (Table 64, bit 28).
  - If you want 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 STARTERP0 register (Table 80).
  - Each selected pin interrupt is assigned to one interrupt in the NVIC (interrupts #24 to #31 for pin interrupts 0 to 7).
- Pattern match engine:
  - Select up to eight external pins from all GPIO port pins in the SYSCON block (Table 79). 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 SYSAHBCLKCTRL register (Table 64, bit 28).
- Each bit slice of the pattern match engine is assigned to one interrupt in the NVIC (interrupts #24 to #31 for slices 0 to 7).
- The combined interrupt from all slices or slice combinations can be connected to the Arm RXEV request and to pin function GPIO_INT_BMAT through the switch matrix movable function register (PINASSIGN6, Table 97).

### 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 LPC804 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 into one of the eight PINTSEL registers in the SYSCON 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 through the switch matrix.
3. Enable each pin interrupt in the NVIC.

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

See Section 6.6.25 “Pin interrupt select registers” in the SYSCON block for the PINTSEL registers.

## 11.4 Pin description

The inputs to the pin interrupt and pattern match engine are determined by the pin interrupt select registers in the SYSCON block. See Section 6.6.25 “Pin interrupt select registers”.

The pattern match engine output is assigned to an external pin through the switch matrix.

See Section 8.3.1 “Connect an internal signal to a package pin” for the steps that you need to follow to assign the GPIO pattern match function to a pin.

**Table 150. Pin interrupt/pattern match engine pin description**

| Function | Direction | Pin | Description | SWM register | Reference |
|---|---|---|---|---|---|
| GPIO_INT_BMAT | O | any | GPIO pattern match output | PINASSIGN6 | Table 97 |

## 11.5 General description

Pins with configurable functions can serve as external interrupts or inputs to the pattern match engine. You can configure up to eight pins total using the PINTSEL registers in the SYSCON 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 79). 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.

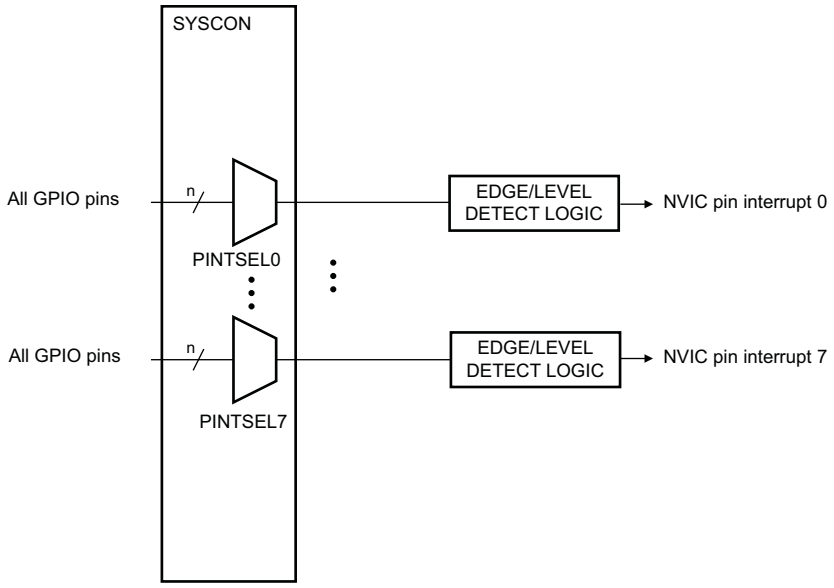
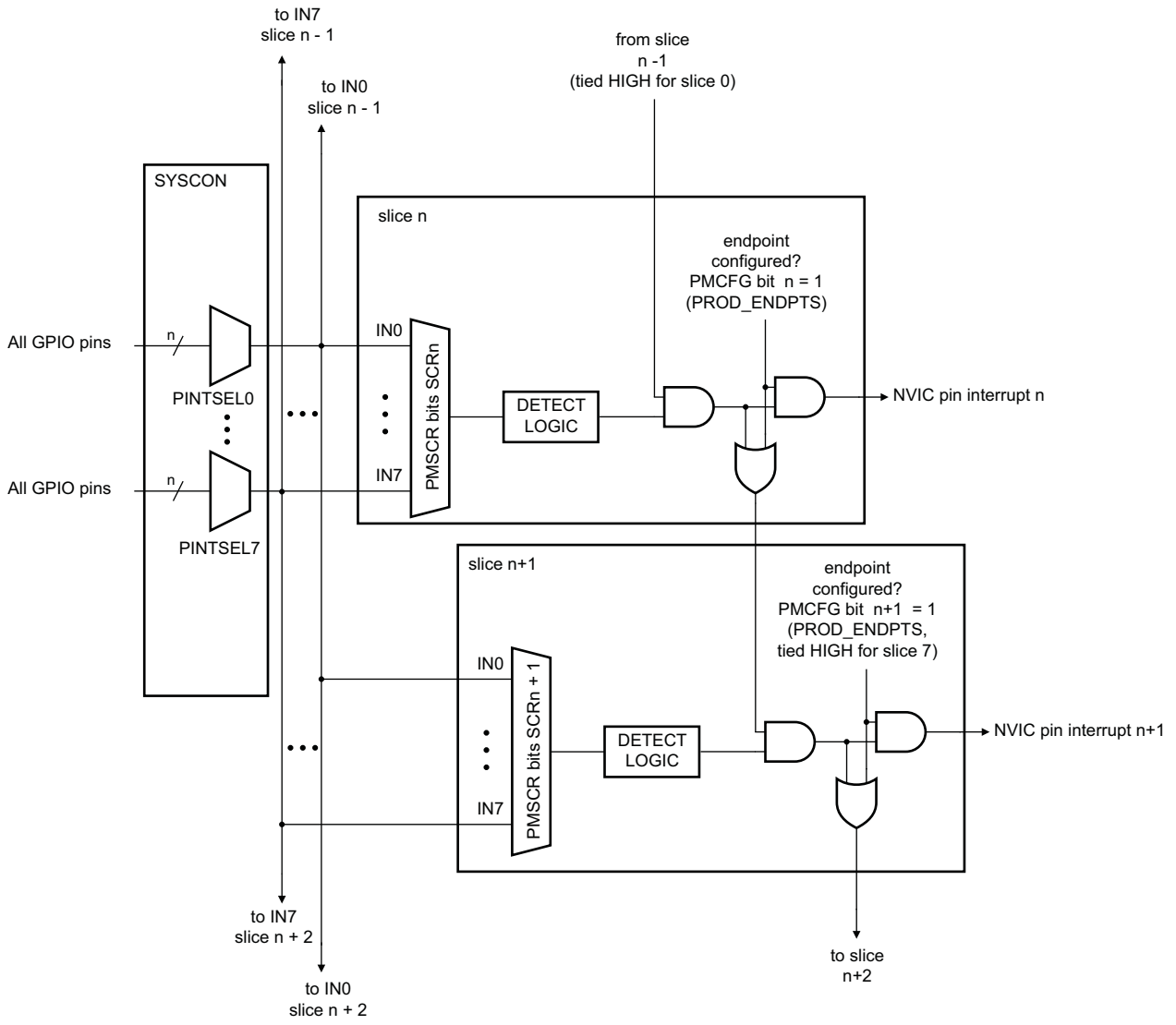



**Fig 14. Pin interrupt connections**

### 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. The slice input selector selects one input from the available eight inputs with each input connected to a pin by the input's PINTSEL register.

The detect logic monitors the selected input continuously and creates a HIGH output if the input qualifies as detected. Several terms can be combined to a minterm by designating a slice as an endpoint of the expression. A pin interrupt for this slice is asserted when the minterm evaluates as true.




See Figure 16 for the detect logic block.

**Fig 15. Pattern match engine connections**

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 16 shows the details of the edge detection logic for each slice.

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

You can create a time window 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.

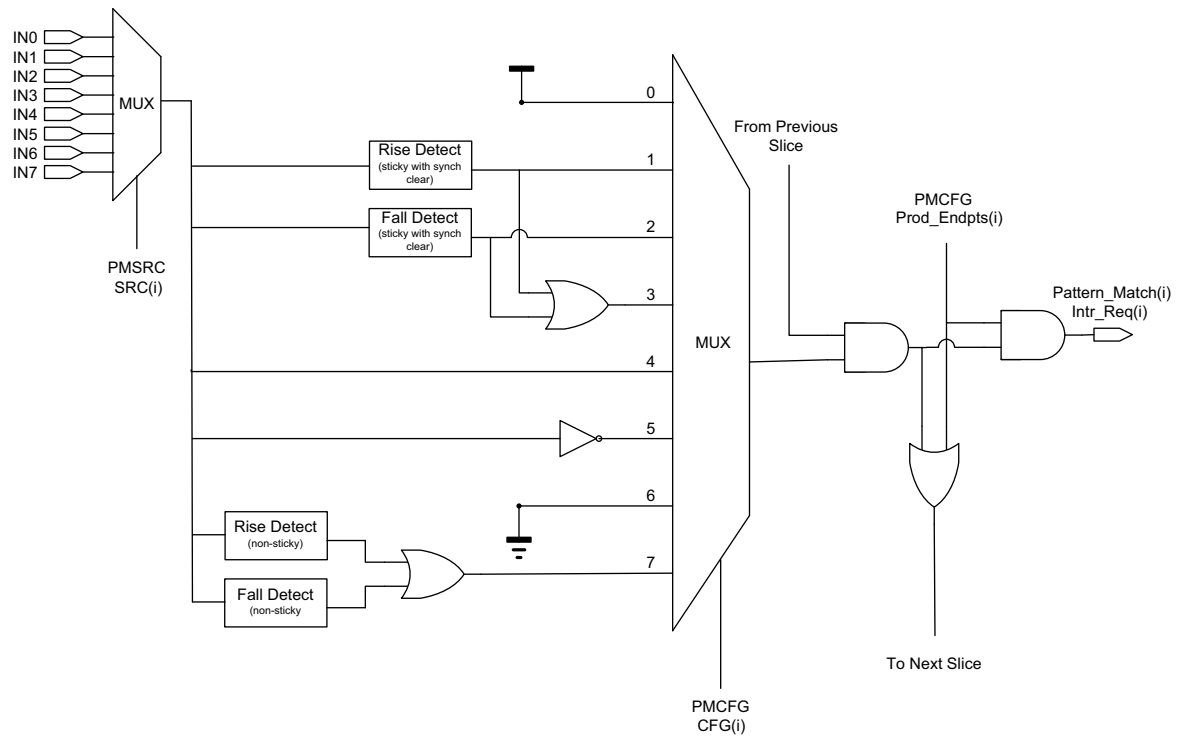



**Fig 16. Pattern match bit slice with detect logic**

### 11.5.2.1 Inputs and outputs of the pattern match engine

The connections between the pins and the pattern match engine are shown in Figure 15. All inputs to the pattern match engine are selected in the SYSCON block and can be GPIO port pins or another pin function depending on the switch matrix configuration.

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 a boolean expression is true (i.e. when any minterm is matched).

The RXEV output is also be routed to GPIO_INT_BMAT pin. This allows the GPIO module to provide a rudimentary programmable logic capability employing up to eight inputs and one output.

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 deep-sleep or power-down mode. Pin interrupts must be selected to use the pins for wake up.

#### 11.5.2.2 Boolean expressions

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 16).

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.

Example:

Assume the expression: (IN0)~(IN1)(IN3)^ + (IN1)(IN2) + (IN0)~(IN3)~(IN4) is specified through the registers PMSRC (Table 163) and PMCFG (Table 164). Each term in the boolean expression, (IN0), ~(IN1), (IN3)^, etc., represents one bit slice of the pattern match engine.

- In the first minterm (IN0)~(IN1)(IN3)^, 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 (PININT2_IRQ) 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 (PININT4_IRQ) will be asserted.
- In the third minterm (IN0)~(IN3)~(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(PININT7_IRQ) 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 minterms are true, the output is asserted.

Related links:

Section 11.7.2

## 11.6 Register description

**Table 151. Register overview: Pin interrupts and pattern match engine (base address: 0xA000 4000)**

| Name | Access | Address offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| ISEL | R/W | 0x000 | Pin Interrupt Mode register | 0 | Table 152 |
| IENR | R/W | 0x004 | Pin interrupt level or rising edge interrupt enable register | 0 | Table 153 |
| SIENR | WO | 0x008 | Pin interrupt level or rising edge interrupt set register | NA | Table 154 |
| CIENR | WO | 0x00C | Pin interrupt level (rising edge interrupt) clear register | NA | Table 155 |
| IENF | R/W | 0x010 | Pin interrupt active level or falling edge interrupt enable register | 0 | Table 156 |
| SIENF | WO | 0x014 | Pin interrupt active level or falling edge interrupt set register | NA | Table 157 |
| CIENF | WO | 0x018 | Pin interrupt active level or falling edge interrupt clear register | NA | Table 158 |
| RISE | R/W | 0x01C | Pin interrupt rising edge register | 0 | Table 159 |
| FALL | R/W | 0x020 | Pin interrupt falling edge register | 0 | Table 160 |
| IST | R/W | 0x024 | Pin interrupt status register | 0 | Table 161 |
| PMCTRL | R/W | 0x028 | Pattern match interrupt control register | 0 | Table 162 |
| PMSRC | R/W | 0x02C | Pattern match interrupt bit-slice source register | 0 | Table 163 |
| PMCFG | R/W | 0x030 | Pattern match interrupt bit slice configuration register | 0 | Table 164 |

### 11.6.1 Pin interrupt mode register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 6.6.25), one bit in the ISEL register determines whether the interrupt is edge or level sensitive.

**Table 152. Pin interrupt mode register (ISEL, address 0xA000 4000) 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.<br>0 = Edge sensitive<br>1 = Level sensitive | 0 | R/W |
| 31:8 | - | Reserved. | - | - |

### 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 Section 6.6.25), 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 153. Pin interrupt level or rising edge interrupt enable register (IENR, address 0xA000 4004) 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.<br>0 = Disable rising edge or level interrupt.<br>1 = Enable rising edge or level interrupt. | 0 | R/W |
| 31:8 | - | Reserved. | - | - |

### 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 Section 6.6.25), 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 154. Pin interrupt level or rising edge interrupt set register (SIENR, address 0xA000 4008) 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.<br>0 = No operation.<br>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 Section 6.6.25), 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 155. Pin interrupt level or rising edge interrupt clear register (CIENR, address 0xA000 400C) 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.<br>0 = No operation.<br>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 Section 6.6.25), 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 156. Pin interrupt active level or falling edge interrupt enable register (IENF, address 0xA000 4010) 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.<br>0 = Disable falling edge interrupt or set active interrupt level LOW.<br>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 Section 6.6.25), 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 157. Pin interrupt active level or falling edge interrupt set register (SIENF, address 0xA000 4014) 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.<br>0 = No operation.<br>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 Section 6.6.25), 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 158. Pin interrupt active level or falling edge interrupt clear register (CIENF, address 0xA000 4018) 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.<br>0 = No operation.<br>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 Section 6.6.25) 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 159. Pin interrupt rising edge register (RISE, address 0xA000 401C) 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.<br>Read 0: No rising edge has been detected on this pin since Reset or the last time a one was written to this bit.<br>Write 0: no operation.<br>Read 1: a rising edge has been detected since Reset or the last time a one was written to this bit.<br>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 Section 6.6.25) 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 160. Pin interrupt falling edge register (FALL, address 0xA000 4020) 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.<br>Read 0: No falling edge has been detected on this pin since Reset or the last time a one was written to this bit.<br>Write 0: no operation.<br>Read 1: a falling edge has been detected since Reset or the last time a one was written to this bit.<br>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 161. Pin interrupt status register (IST, address 0xA000 4024) 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.<br>Read 0: interrupt is not being requested for this interrupt pin.<br>Write 0: no operation.<br>Read 1: interrupt is being requested for this interrupt pin.<br>Write 1 (edge-sensitive): clear rising- and falling-edge detection for this pin.<br>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.

**Table 162. Pattern match interrupt control register (PMCTRL, address 0xA000 4028) 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 Arm 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 |
| 31:24 | 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 6.6.25. 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 163. Pattern match bit-slice source register (PMSRC, address 0xA000 402C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 7:0 | Reserved | | Software should not write 1s to unused bits. | 0 |

**Table 163. Pattern match bit-slice source register (PMSRC, address 0xA000 402C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 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. | |
| 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. | |

**Table 163. Pattern match bit-slice source register (PMSRC, address 0xA000 402C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 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. | |

**Table 163. Pattern match bit-slice source register (PMSRC, address 0xA000 402C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 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. | |
| 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. | |

**Table 163. Pattern match bit-slice source register (PMSRC, address 0xA000 402C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 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_ENPTSn 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 164. Pattern match bit slice configuration register (PMCFG, address 0xA000 4030) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | PROD_ENDPTS0 | | 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_ENDPTS1 | | 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_ENDPTS2 | | 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_ENDPTS3 | | 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_ENDPTS4 | | 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. | |

**Table 164. Pattern match bit slice configuration register (PMCFG, address 0xA000 4030) bit description** *…continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 5 | PROD_ENDPTS5 | | 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_ENDPTS6 | | 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. | |
| 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. | |

**Table 164. Pattern match bit slice configuration register (PMCFG, address 0xA000 4030) bit description** *…continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 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. | |
| 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. | |

**Table 164. Pattern match bit slice configuration register (PMCFG, address 0xA000 4030) bit description** *…continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 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. | |
| 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. | |

**Table 164. Pattern match bit slice configuration register (PMCFG, address 0xA000 4030) bit description** *…continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 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. | |
| 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). | |
| | | 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 164. Pattern match bit slice configuration register (PMCFG, address 0xA000 4030) bit description** *…continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 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 165.

**Table 165. 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) + (~IN2 * ~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 163):
  - Since bit slice 5 will be used to detect a sticky event on input 6, you can write a 1 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 164):
  - 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 162):

– 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 Arm 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

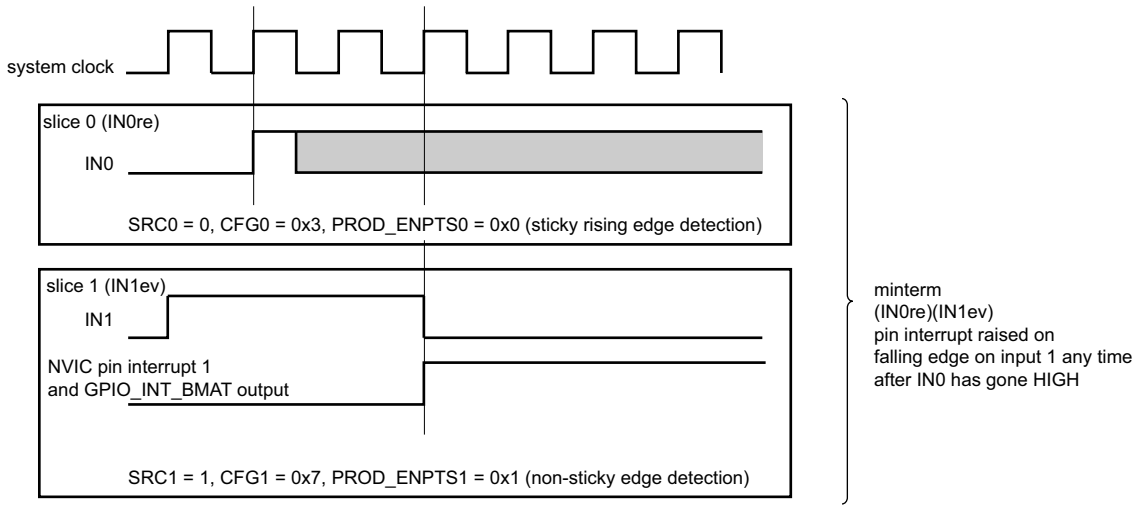



Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

**Fig 17. Pattern match engine examples: sticky edge detect**

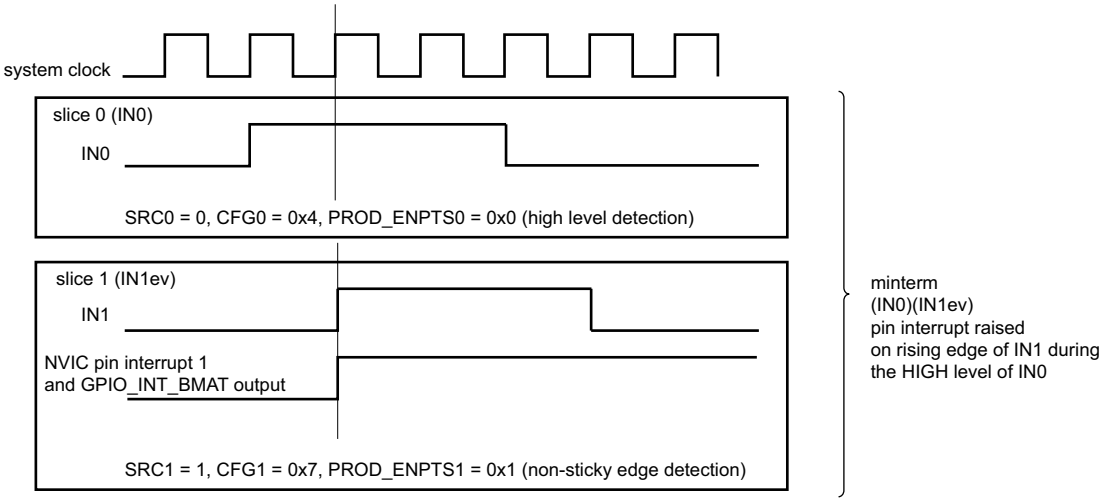



Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

**Fig 18. Pattern match engine examples: Windowed non-sticky edge detect evaluates as true**

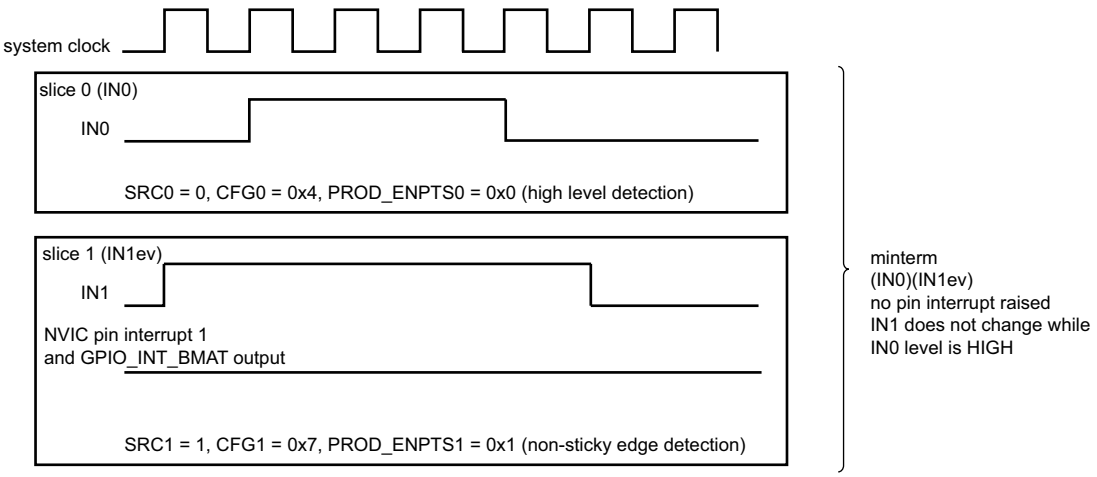



Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

**Fig 19. Pattern match engine examples: Windowed non-sticky edge detect evaluates as false**

## 12.1 How to read this chapter

This chapter provides an overview of power related information about LPC804 devices and allows the user to configure the reduced power modes deep-sleep mode, power-down mode, and deep power-down mode.

To turn analog components on or off in active and sleep modes, use the PDRUNCFG register (Section 6.6.30 “Power configuration register”). PDSLEEPCFG register can be used to turn analog peripherals on or off for deep-sleep mode and power-down modes.

## 12.2 Features

- Reduced power modes control
- Five general purpose backup registers to retain data in deep power-down mode

## 12.3 Basic configuration

### 12.3.1 Low power modes in the Arm Cortex-M0+ core

Entering and exiting the low power modes is always controlled by the Arm Cortex-M0+ core. The SCR register is the software interface for controlling the core’s actions when entering a low power mode. The SCR register is located on the Arm private peripheral bus. For details, see Ref. 3.

#### 12.3.1.1 System control register

The System control register (SCR) controls entry to and exit from a low power state. This register is located on the private peripheral bus and is a R/W register with reset value of 0x0000 0000. The SCR register, in combination with the PMU PCON register (Table 169), selects the sleep/power-down mode for the Arm core and the entire system. The SLEEPDEEP bit selects between: 0 = sleep mode, that stops the Arm core clock; 1 = deep-sleep mode, that stops the system clock.

**Table 166. System control register (SCR, address 0xE000 ED10) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 0 | - | Reserved. | 0 |
| 1 | SLEEPONEXIT | Indicates sleep-on-exit when returning from Handler mode to Thread mode:<br>0 = do not sleep when returning to Thread mode.<br>1 = enter sleep, or deep sleep, on return from an ISR to Thread mode.<br>Setting this bit to 1 enables an interrupt driven application to avoid returning to an empty main application. | 0 |
| 2 | SLEEPDEEP | Controls whether the processor uses sleep or deep-sleep as its low power mode:<br>0 = sleep<br>1 = deep sleep. | 0 |
| 3 | - | Reserved. | 0 |
| 4 | SEVONPEND | Send Event on Pending bit:<br>0 = only enabled interrupts or events can wake-up the processor, disabled interrupts are excluded<br>1 = enabled events and all interrupts, including disabled interrupts, can wake up the processor.<br>When an event or interrupt enters pending state, the event signal wakes up the processor from WFE. If the processor is not waiting for an event, the event is registered and affects the next WFE.<br>The processor also wakes up on execution of an `SEV` instruction. | 0 |
| 31:5 | - | Reserved. | 0 |

## 12.4 Pin description

A few $\overline{\text{WAKEUP}}$ pins can be used to wake up only from deep power-down mode, if enabled. PIO0_15, PIO0_9, PIO0_8, PIO0_17, PIO0_13, PIO0_4, PIO0_11, PIO0_10 are pins that have $\overline{\text{WAKEUP}}$ functions. Select the $\overline{\text{WAKEUP}}$ function in the WUENAREG register.

**Remark:** When entering deep power-down mode, an external pull-up resistor is required on the $\overline{\text{WAKEUP}}$ pin to hold it HIGH if this pin is enabled for the $\overline{\text{WAKEUP}}$ function.

## 12.5 General description

Power on the LPC804 is controlled by the PMU, the SYSCON block, and the Arm Cortex-M0+ core. The following reduced power modes are supported in order from highest to lowest power consumption:

1. Sleep mode:

   The sleep mode affects the Arm Cortex-M0+ core only. Peripherals and memories are active.

2. Deep-sleep and power-down modes:

   The deep-sleep and power-down modes affect the core and the entire system with memories and peripherals. Before entering deep-sleep or power-down, you must switch the main clock to the FRO to provide a clock signal that can be shut down cleanly. The peripherals receive no internal clocks. The internal SRAM memory and all peripheral registers as well as the processor maintain their internal states. The WWDT, WKT, and BOD can remain active to wake up the system on an interrupt.

   a. In deep-sleep mode, the flash is in standby mode.

   b. In power-down mode, the flash memory is powered down.

3. Deep power-down mode:

   For maximal power savings, the entire system is shut down except for the general purpose registers in the PMU. Only the general purpose registers in the PMU maintain their internal states. The part can wake up on a pulse on one of the WAKEUP pins. On wake-up, the part reboots.

**Remark:** The part is in active mode when it is fully powered and operational after booting.

### 12.5.1 Wake-up process

If the part receives a wake-up signal in any of the reduced power modes, it wakes up to the active mode.

See these links for related registers and wake-up instructions:

- To configure the system after wake-up: Table 83 “Wake-up configuration register (PDAWAKECFG, address 0x4004 8234) bit description”.
- To use external interrupts for wake-up: Table 80 “Start logic 0 pin wake-up enable register 0 (STARTERP0, address 0x4004 8204) bit description” and Table 79 “Pin interrupt select registers (PINTSEL[0:7], address 0x4004 8178 (PINTSEL0) to 0x4004 8194 (PINTSEL7)) bit description”
- To enable external or internal signals to wake up the part from deep-sleep or power-down modes: Table 81 “Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description”
- To configure the USART to wake up the part: Section 13.3.2 “Configure the USART for wake-up”
- For configuring the self-wake-up timer: Section 18.5
- For a list of all wake-up sources: Table 167 “Wake-up sources for reduced power modes”

**Table 167. 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 STARTERP0 registers. |
| | BOD interrupt | • Enable interrupt in NVIC and STARTERP1 registers.<br>• Enable interrupt in BODCTRL register.<br>• BOD powered in PDSLEEPCFG register. |
| | BOD reset | • Enable reset in BODCTRL register.<br>• BOD powered in PDSLEEPCFG register. |
| | WWDT interrupt | • Enable interrupt in NVIC and STARTERP1 registers.<br>• WWDT running. Enable WWDT in WWDT MOD register and feed.<br>• Enable interrupt in WWDT MOD register.<br>• LPOSC powered in PDSLEEPCFG register. |
| | WWDT reset | • WWDT running.<br>• Enable reset in WWDT MOD register.<br>• LPOSC powered in PDSLEEPCFG register. |
| | Self-Wake-up Timer (WKT) time-out | • Enable interrupt in NVIC and STARTERP1 registers.<br>• Enable low-power oscillator in the LPOSCCLKEN register in the SYSCON block.<br>• Select low-power clock for WKT clock in the WKT CTRL register.<br>• Start the WKT by writing a time-out value to the WKT COUNT register. |
| | Interrupt from USART/SPI/I2C peripheral | • Enable interrupt in NVIC and STARTERP1 registers.<br>• Enable USART/I2C/SPI interrupts.<br>• Provide an external clock signal to the peripheral.<br>• Configure the USART in synchronous slave mode and I2C and SPI in slave mode. |
| Deep power-down | $\overline{\text{WAKEUP}}$ pins | Enable the $\overline{\text{WAKEUP}}$ function in the WUENAREG register in the PMU. |

## 12.6 Register description

**Table 168. Register overview: PMU (base address 0x4002 0000)**

| Name | Access | Address offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| PCON | R/W | 0x000 | Power control register | 0x0 | Table 169 |
| GPREG0 | R/W | 0x004 | General purpose register 0 | 0x0 | Table 170 |
| GPREG1 | R/W | 0x008 | General purpose register 1 | 0x0 | Table 170 |
| GPREG2 | R/W | 0x00C | General purpose register 2 | 0x0 | Table 170 |
| GPREG3 | R/W | 0x010 | General purpose register 3 | 0x0 | Table 170 |
| GPREG4 | R/W | 0x014 | General purpose register 4 | 0x0 | Table 170 |
| Deep power-down wake-up source status register | R/W | 0x020 | Pin wake-up status register | 0x0 | Table 171 |
| Deep power-down wake-up enable register | R/W | 0x024 | Pin wake-up enable register | 0x0 | Table 172 |

### 12.6.1 Power control register

The power control register selects whether one of the Arm Cortex-M0+ controlled power-down modes (sleep mode or deep-sleep/power-down mode) or the deep power-down mode is entered and provides the flags for sleep or deep-sleep/power-down modes and deep power-down modes respectively.

**Table 169. Power control register (PCON, address 0x4002 0000) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | PM | | Power mode | 0 |
| | | 0x0 | Default. The part is in active or sleep mode. | |
| | | 0x1 | Deep-sleep mode. Arm WFI will enter deep-sleep mode. | |
| | | 0x2 | Power-down mode. Arm WFI will enter power-down mode. | |
| | | 0x3 | Deep power-down mode. Directly enter deep power-down mode. | |
| 3 | NODPD | | A 1 in this bit prevents entry to deep power-down mode when 0x3 is written to the PM field above, the SLEEPDEEP bit is set, and a WFI is executed.<br>This bit is cleared only by power-on reset, so writing a one to this bit locks the part in a mode in which deep power-down mode is blocked. | 0 |
| 7:4 | - | - | Reserved. Do not write ones to this bit. | 0 |
| 8 | SLEEPFLAG | | Sleep mode flag | 0 |
| | | 0 | Active mode. Read: No power-down mode entered. Part is in Active mode.<br>Write: No effect. | |
| | | 1 | Low power mode. Read: sleep, deep-sleep or power-down mode entered.<br>Write: Writing a 1 clears the SLEEPFLAG bit to 0. | |
| 10:9 | - | - | Reserved. Do not write ones to this bit. | 0 |
| 11 | DPDFLAG | | Deep power-down flag | 0 |
| | | 0 | Not deep power-down. Read: deep power-down mode **not** entered.<br>Write: No effect. | 0 |
| | | 1 | Deep power-down. Read: deep power-down mode entered.<br>Write: Clear the deep power-down flag. | |
| 31:12 | - | - | Reserved. Do not write ones to this bit. | 0 |

### 12.6.2 General purpose registers 0 to 4

The general purpose registers retain data through the deep power-down mode when power is still applied to the $V_{DD}$ pin but the chip has entered deep power-down mode. Only a cold boot - when all power has been completely removed from the chip - will reset the general purpose registers.

**Table 170. General purpose registers 0 to 4 (GPREG[0:4], address 0x4002 0004 (GPREG0) to 0x4002 0014 (GPREG4)) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 31:0 | GPDATA | Data retained during deep power-down mode. | 0x0 |

### 12.6.3 Deep power-down wake-up source status register

The WUSRCREG register provides a status on the wake-up pin. The associated bit is set when the corresponding wake-up pin changes to LOW. When the wake-up pin changes to HIGH, writing a 1 to the corresponding bit clears the previous status. Bits should be cleared (writing 1) before deep power-down mode.

**Table 171. WUSRCREG, address 0x4002 0020 bit description**

| Bit | Symbol | Bit | Description | Reset value |
|---|---|---|---|---|
| 7:0 | WUSRCREG | | Pin wake-up status register. | 0 |
| | WUSRCREG0 | 0 | PIO0_15 | 0 |
| | WUSRCREG1 | 1 | PIO0_9 | 0 |
| | WUSRCREG2 | 2 | PIO0_8 | 0 |
| | WUSRCREG3 | 3 | PIO0_17 | 0 |
| | WUSRCREG4 | 4 | PIO0_13 | 0 |
| | WUSRCREG5 | 5 | PIO0_4 | 0 |
| | WUSRCREG6 | 6 | PIO0_11 | 0 |
| | WUSRCREG7 | 7 | PIO0_10 | 0 |
| 31:8 | - | - | Reserved | - |

### 12.6.4 Deep power-down wake-up enable register

The WUENAREG register is used to enable the wake-up pins for deep power-down mode.

**Table 172. WUENAREG, address 0x4002 0024 bit description**

| Bit | Symbol | Bit | Description | Reset value |
|---|---|---|---|---|
| 7:0 | WUENAREG | | Pin wake-up enable register. | 0 |
| | WUENAREG0 | 0 | PIO0_15 | 0 |
| | WUENAREG1 | 1 | PIO0_9 | 0 |
| | WUENAREG2 | 2 | PIO0_8 | 0 |
| | WUENAREG3 | 3 | PIO0_17 | 0 |
| | WUENAREG4 | 4 | PIO0_13 | 0 |
| | WUENAREG5 | 5 | PIO0_4 | 0 |
| | WUENAREG6 | 6 | PIO0_11 | 0 |
| | WUENAREG7 | 7 | PIO0_10 | 0 |
| 31:8 | - | - | Reserved | - |

## 12.7 Functional description

### 12.7.1 Power management

The part supports 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.

**Table 173. Peripheral configuration in reduced power modes**

| Peripheral | Sleep mode | Deep-sleep mode | Power-down mode | Deep power-down mode |
|---|---|---|---|---|
| FRO | software configurable | on | off | off |
| FRO output | software configurable | off | off | off |
| Flash | software configurable | standby | off | off |
| BOD | software configurable | software configurable | software configurable | off |
| LPOsc/WWDT | software configurable | software configurable | software configurable | off |
| Digital peripherals | software configurable | off | off | off |
| Wake-up buffers | software configurable (cannot be used as wake-up source) | software configurable (cannot be used as wake-up source) | software configurable (cannot be used as wake-up source) | software configurable |
| ADC | software configurable | off | off | off |
| DAC | software configurable | off | off | off |
| Capacitive Touch | software configurable | software configurable | software configurable | off |
| WKT/low-power oscillator | software configurable | software configurable | software configurable | off |
| Comparator | software configurable | off | off | off |
| PLU | off | off | off | off |

**Remark:** The Debug mode is not supported in sleep, deep-sleep, power-down, or deep power-down modes.

### 12.7.2 Reduced power modes and WWDT lock features

The WWDT lock feature influences the power consumption in any of the power modes because locking the WWDT clock source forces the low power oscillator to be on independently of the deep-sleep and power-down mode software configuration through the PDSLEEPCFG register. For details see Section 17.5.3 “Using the WWDT lock features”.

### 12.7.3 Active mode

In Active mode, the Arm Cortex-M0+ core, memories, and peripherals are clocked by the system clock or main clock.

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

#### 12.7.3.1 Power configuration in Active mode

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

- The SYSAHBCLKCTRL register controls which memories and peripherals are running (Table 64).
- The power to various analog blocks (oscillators, the BOD circuit, and the flash block) can be controlled at any time individually through the PDRUNCFG register (Table 84 “Power configuration register (PDRUNCFG, address 0x4004 8238) bit description”).
- The clock source for the system clock can be selected from the FRO (default), the external clock, or the low power oscillator (see Figure 6 and related registers).
- The system clock frequency can be selected by the MAINCLKSEL (Table 57) and the SYSAHBCLKDIV register (Table 59).
- The ADC clock and CLKOUT use their own clock dividers. The clocks can be shut down through the corresponding clock divider registers. Select the "none" option in the clock select register to shut down the clock to the peripheral and its divider.
- The two USARTs, $I^2C$ and SPI share the same Fractional Rate Divider and its clock select multiplexer. Select the "none" option in the FRG0CLKSEL register to shut down the clock.

### 12.7.4 Sleep mode

In sleep mode, the system clock to the Arm Cortex-M0+ core 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 SYSAHBCLKCTRL register, 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.

#### 12.7.4.1 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 selected as in Active mode.

#### 12.7.4.2 Programming sleep mode

The following steps must be performed to enter sleep mode:

1. The PM bits in the PCON register must be set to the default value 0x0.
2. The SLEEPDEEP bit in the Arm Cortex-M0+ SCR register must be set to zero (Table 166).
3. Use the Arm Cortex-M0+ Wait-For-Interrupt (WFI) instruction.

#### 12.7.4.3 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 due to an interrupt, the microcontroller returns to its original power configuration defined by the contents of the PDRUNCFG and the SYSAHBCLKDIV registers. If a reset occurs, the microcontroller enters the default configuration in Active mode.

### 12.7.5 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, except for the BOD circuit and the low power oscillator, which can be selected or deselected during deep-sleep mode in the PDSLEEPCFG register. The main clock, and therefore all peripheral clocks, are disabled except for the clock to the watchdog timer if the low power oscillator is enabled. The FRO is running, but its output is disabled. The flash is in standby 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.

#### 12.7.5.1 Power configuration in deep-sleep mode

Power consumption in deep-sleep mode is determined by the deep-sleep power configuration setting in the PDSLEEPCFG (Table 82) register:

- The low power oscillator can be left running in deep-sleep mode if required for the WWDT.
- The BOD circuit can be left running in deep-sleep mode if required by the application.

#### 12.7.5.2 Programming deep-sleep mode

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

1. The PM bits in the PCON register must be set to 0x1 (Table 169).
2. Select the power configuration in deep-sleep mode in the PDSLEEPCFG (Table 82) register.
3. Select the power configuration after wake-up in the PDAWAKECFG (Table 83) register.
4. If any of the available wake-up interrupts are needed for wake-up, enable the interrupts in the interrupt wake-up registers (Table 80, Table 81) and in the NVIC.
5. Select the FRO as the main clock.
6. Write one to the SLEEPDEEP bit in the Arm Cortex-M0+ SCR register (Table 166).
7. Use the Arm WFI instruction.

#### 12.7.5.3 Wake-up from deep-sleep mode

The microcontroller can wake up from deep-sleep mode in the following ways:

- Signal on one of the eight pin interrupts selected in Table 79. Each pin interrupt must also be enabled in the STARTERP0 register (Table 80) and in the NVIC.
- BOD signal, if the BOD is enabled in the PDSLEEPCFG register:

- BOD interrupt using the deep-sleep interrupt wake-up register 1 (Table 81). The BOD interrupt must be enabled in the NVIC. The BOD interrupt must be selected in the BODCTRL register.
- Reset from the BOD circuit. In this case, the BOD circuit must be enabled in the PDSLEEPCFG register, and the BOD reset must be enabled in the BODCTRL register (Table 75).

- WWDT signal, if the low power oscillator is enabled in the PDSLEEPCFG register:
  - WWDT interrupt using the interrupt wake-up register 1 (Table 81). The WWDT interrupt must be enabled in the NVIC. The WWDT interrupt must be set in the WWDT MOD register, and the WWDT must be enabled in the SYSAHBCLKCTRL register.
  - Reset from the watchdog timer. The WWDT reset must be set in the WWDT MOD register. In this case, the low power oscillator must be running in deep-sleep mode (see PDSLEEPCFG register), and the WDT must be enabled in the SYSAHBCLKCTRL register.
- Via any of the USART blocks if the USART is configured in synchronous mode. See Section 13.3.2 “Configure the USART for wake-up”.
- Via the $I^2C$. See Section 15.3.3.
- Via the SPI. See Section 14.3.1.

**Remark:** To disable the BOD in deep sleep mode if enabled in active mode, disable BOD reset (bit 4 in the BODCTRL register) before entering power-down mode. After wake-up, enable BOD reset (bit 4 in the BODCTRL register).

### 12.7.6 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, except for the BOD circuit and the low power oscillator, which must be selected or deselected during power-down mode in the PDSLEEPCFG register. The main clock and therefore all peripheral clocks are disabled except for the clock to the watchdog timer if the low power oscillator is enabled. The FRO itself and the flash are powered down, decreasing power consumption compared to deep-sleep mode.

Power-down 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. Wake-up times are longer compared to the deep-sleep mode.

#### 12.7.6.1 Power configuration in power-down mode

Power consumption in power-down mode can be configured by the power configuration setting in the PDSLEEPCFG (Table 82) register in the same way as for deep-sleep mode (see Section 12.7.5.1):

- The low power oscillator can be left running in power-down mode if required for the WWDT.
- The BOD circuit can be left running in power-down mode if required by the application.

#### 12.7.6.2 Programming power-down mode

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

1. The PM bits in the PCON register must be set to 0x2 (Table 169).
2. Select the power configuration in power-down mode in the PDSLEEPCFG (Table 82) register.
3. Select the power configuration after wake-up in the PDAWAKECFG (Table 83) register.
4. If any of the available wake-up interrupts are used for wake-up, enable the interrupts in the interrupt wake-up registers (Table 80, Table 81) and in the NVIC.
5. Select the FRO as the main clock.
6. Write one to the SLEEPDEEP bit in the Arm Cortex-M0+ SCR register (Table 166).
7. Use the Arm WFI instruction.

#### 12.7.6.3 Wake-up from power-down mode

The microcontroller can wake up from power-down mode in the same way as from deep-sleep mode:

- Signal on one of the eight pin interrupts selected in Table 79. Each pin interrupt must also be enabled in the STARTERP0 register (Table 80) and in the NVIC.
- BOD signal, if the BOD is enabled in the PDSLEEPCFG register:
  - BOD interrupt using the interrupt wake-up register 1 (Table 81). The BOD interrupt must be enabled in the NVIC. The BOD interrupt must be selected in the BODCTRL register.
  - Reset from the BOD circuit. In this case, the BOD reset must be enabled in the BODCTRL register (Table 75).
- WWDT signal, if the low power oscillator is enabled in the PDSLEEPCFG register:
  - WWDT interrupt using the interrupt wake-up register 1 (Table 81). The WWDT interrupt must be enabled in the NVIC. The WWDT interrupt must be set in the WWDT MOD register.
  - Reset from the watchdog timer. The WWDT reset enable must be set in the WWDT MOD register.
  - Via any of the USART blocks. See Section 13.3.2 “Configure the USART for wake-up”.
  - Via the $I^2C$. See Section 15.3.3.
  - Via the SPI. See Section 14.3.1.

  **Remark:** If the BOD is enabled in active mode and the user needs to disable the BOD in power-down mode, disable the BOD reset (bit 4 in the BODCTRL register) before entering power-down mode. After wake-up, enable the BOD reset (bit 4 in the BODCTRL register).

### 12.7.7 Deep power-down mode

In deep power-down mode, power and clocks are shut off to the entire chip with the exception of the $\overline{\text{WAKEUP}}$ pins and general purpose registers. Five general-purpose registers are available to store information during deep power-down mode.

During deep power-down mode, the contents of the SRAM and registers are not retained except for a small amount of data which can be stored in the general purpose registers of the PMU block.

All functional pins are tri-stated in deep power-down mode except for the $\overline{\text{WAKEUP}}$ pin.

**Remark:** Setting bit 3 in the PCON register (Table 169) prevents the part from entering deep-power down mode.

#### 12.7.7.1 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 $\overline{\text{WAKEUP}}$ pins are powered.

#### 12.7.7.2 Programming deep power-down mode using the $\overline{\text{WAKEUP}}$ pin:

The following steps must be performed to enter deep power-down mode when using the $\overline{\text{WAKEUP}}$ pin for waking up:

1. Pull the $\overline{\text{WAKEUP}}$ pin externally HIGH.
2. Ensure that bit 3 in the PCON register (Table 169) is cleared.
3. Write 0x3 to the PM bits in the PCON register (see Table 169).
4. Store data to be retained in the general purpose registers (Section 12.6.2).
5. Write one to the SLEEPDEEP bit in the Arm Cortex-M0+ SCR register (Table 166).
6. Use the Arm WFI instruction.

#### 12.7.7.3 Wake-up from deep power-down mode using the $\overline{\text{WAKEUP}}$ pin:

Pulling the $\overline{\text{WAKEUP}}$ pins LOW wakes up the LPC804 from deep power-down, and the part goes through the entire reset process.

1. On the $\overline{\text{WAKEUP}}$ pins, transition from HIGH to LOW.
   - 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 re-boots.
   - All registers except the GPREG0 to GPREG4 registers and PCON will be in their reset state.
2. Once the chip has booted, read the deep power-down flag in the PCON register (Table 169) to verify that the reset was caused by a wake-up event from deep power-down and was not a cold reset.
3. Clear the deep power-down flag in the PCON register (Table 169).
4. (Optional) Read the stored data in the general purpose registers (Section 12.6.2).
5. Set up the PMU for the next deep power-down cycle.

## 13.1 How to read this chapter

Two USARTs are available on all parts depending on the switch matrix configuration. UART1 does not support RTS/CTS hardware signaling for automatic flow control.

## 13.2 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.
- 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 autobaud 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.

## 13.3 Basic configuration

Configure the USARTs for receiving and transmitting data:

- In the SYSAHBCLKCTRL register, set bit 14 and 15 (Table 64) to enable the clock to the register interface.
- Clear the USART peripheral resets using the PRESETCTRL register. See Table 66.
- Enable or disable the USART interrupts in slots #3 and #4 in the NVIC. See Table 38.
- Configure the USART pin functions through the switch matrix.
- Configure the USART clock and baud rate. See Section 13.3.1.

For wake-up from deep-sleep and power-down modes the USART must be configured in synchronous mode. See Section 13.3.2 for details.

### 13.3.1 Configure the USART clock and baud rate

Two USARTs have a shared clock selection that include a shared fractional divider. The fractional divider for the baud rate calculation is set up in the SYSCON block as follows (see Figure 20):

1. If a fractional value is needed to obtain a particular baud rate, program the fractional divider. 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.

   FCLK

   = (FRGINPUTCLK)/(1+(MULT/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.

2. In asynchronous mode: Configure the baud rate divider BRGVAL in the USARTn BRG register. The baud rate divider divides the FCLK clock to create the clock needed to produce the required baud rate.

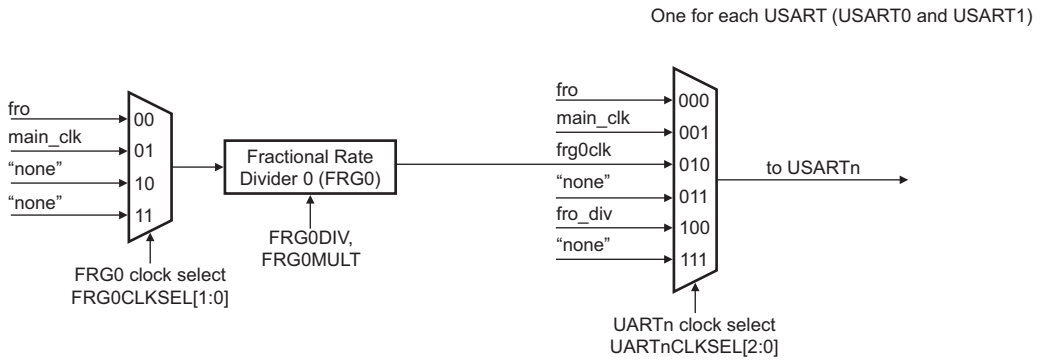
   baud rate = FCLK/((OSRVAL+1) x (BRGVAL+1)).

   BRGVAL = FCLK/((OSRVAL + 1) x baud rate) – 1

   (assumes FCLK ≥ oversample rate x baud rate)

   Section 13.6.9 “USART Baud Rate Generator register”

3. In synchronous mode: The serial clock is Un_SCLK = FCLK/(BRGVAL+1).




See Figure 7 “LPC804 clock generation (continued) aaa-get number”.

**Fig 20. USART clocking**

For details on the clock configuration see:

Section 13.7.1 “Clocking and baud rates”

### 13.3.2 Configure the USART for wake-up

The 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, you have 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 clocks from the Arm core, 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.

#### 13.3.2.1 Wake-up from sleep mode

- Configure the USART in either asynchronous mode or synchronous mode. See Table 176.
- 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 179).

#### 13.3.2.2 Wake-up from deep-sleep or power-down mode

- Configure the USART in synchronous slave mode. See Table 176. You must connect the SCLK function to a pin and connect the pin to the master.
- Enable the USART wake-up in the STARTERP1 register. See Table 81 “Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description”.
- Enable the USART interrupt in the NVIC.
- In the PDAWAKE register, configure all peripherals that need to be running when the part wakes up.
- The USART wakes up the part from deep-sleep or power-down mode on all events that cause an interrupt and are also enabled in the INTENSET register. Typical wake-up events are:
  - A start bit has been received.
  - The RXDAT buffer has received a byte.
  - 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 and the DELTACTS interrupt is enabled. This event wakes up the part without the synchronous UART clock running.

  **Remark:** By enabling or disabling the interrupt in the INTENSET register (Table 179), you can customize when the wake-up occurs in the USART receive/transmit protocol.

## 13.4 Pin description

**Table 174. USART pin description**

| Function | I/O | Type | Connect to | Use register | Reference | Description |
|---|---|---|---|---|---|---|
| U0_TXD | O | external to pin | any pin | PINASSIGN0 | Table 91 | Transmitter output for USART0. Serial transmit data. |
| U0_RXD | I | external to pin | any pin | PINASSIGN0 | Table 91 | Receiver input for USART0. |
| $\overline{\text{U0\_RTS}}$ | O | external to pin | any pin | PINASSIGN0 | Table 91 | Request To Send output for USART0. This signal supports inter-processor communication through the use of hardware flow control. This feature is active when the USART RTS signal is configured to appear on a device pin. |
| $\overline{\text{U0\_CTS}}$ | I | external to pin | any pin | PINASSIGN0 | Table 91 | 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 de-asserted (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 | external to pin | any pin | PINASSIGN1 | Table 92 | Serial clock input/output for USART0 n synchronous mode. |
| U1_TXD | O | external to pin | any pin | PINASSIGN1 | Table 92 | Transmitter output for USART1. Serial transmit data. |
| U1_RXD | I | external to pin | any pin | PINASSIGN1 | Table 92 | Receiver input for USART1. |
| U1_SCLK | I/O | external to pin | any pin | PINASSIGN1 | Table 92 | Serial clock input/output for USART1 in synchronous mode. |

## 13.5 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.

When RTS signal is configured as an RS-485 output enable, it is asserted at the beginning of an transmitted character, and de-asserted either at the end of the character, or after a one character delay (selected by software).

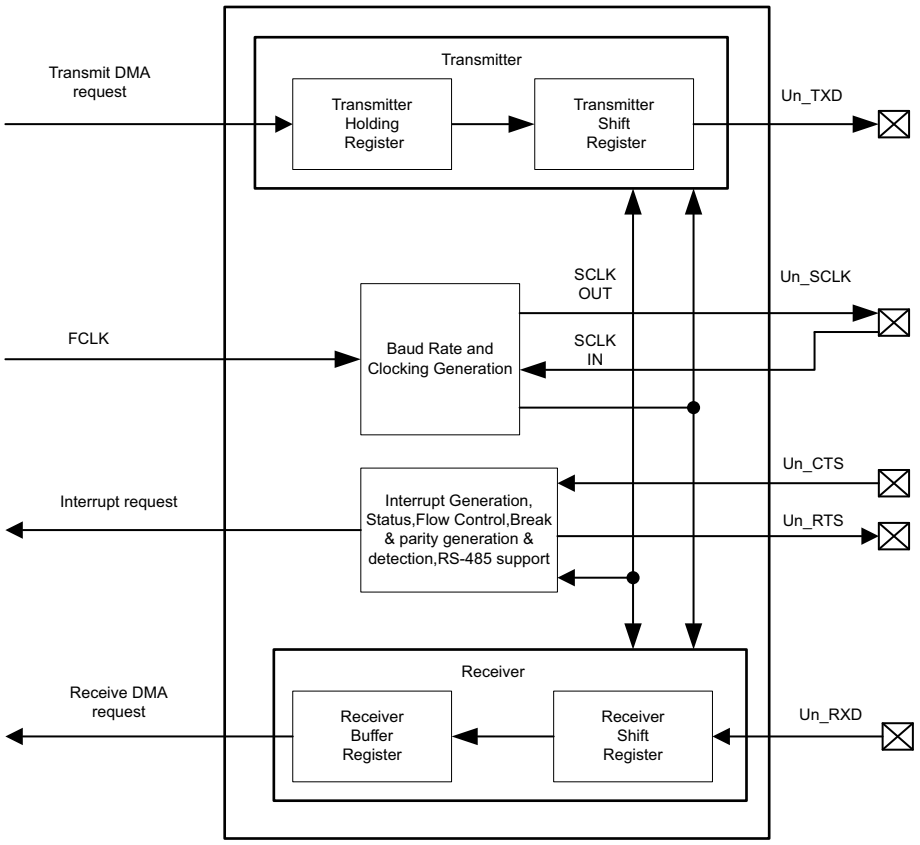
The USART transmitter block accepts data written by the CPU and buffers the data in the transmit holding register. When the transmitter is available, the transmit shift register takes that data, formats it, and serializes it to the serial output, Un_TXD.

The Baud Rate Generator block divides the incoming clock to create a 16x baud rate clock in the standard asynchronous operating mode. The BRG clock input source is the shared Fractional Rate Generator that runs from the common USART peripheral clock U_PCLK).

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.

**Remark:** The fractional value and the USART peripheral clock are shared between all USARTs.




USART block diagram

U_PCLK = FRGCLKDIV/(1+MULT/DIV)

**Fig 21. USART block diagram**

## 13.6 Register description

The reset value reflects the data stored in used bits only. It does not include the content of reserved bits.

**Table 175. Register overview: USART (base address 0x4006 4000 (USART0) and 0x4006 8000 (USART1))**

| Name | Access | Offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| CFG | R/W | 0x000 | USART Configuration register. Basic USART configuration settings that typically are not changed during operation. | 0 | Table 176 |
| CTL | R/W | 0x004 | USART Control register. USART control settings that are more likely to change during operation. | 0 | Table 177 |
| STAT | R/W | 0x008 | 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. | 0x000E | Table 178 |
| INTENSET | R/W | 0x00C | 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 179 |
| INTENCLR | W | 0x010 | 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 180 |
| RXDAT | R | 0x014 | Receiver Data register. Contains the last character received. | - | Table 181 |
| RXDATSTAT | R | 0x018 | Receiver Data with Status register. Combines the last character received with the current USART receive status. Allows software to recover incoming data and status together. | - | Table 182 |
| TXDAT | R/W | 0x01C | Transmit Data register. Data to be transmitted is written here. | 0 | Table 183 |
| BRG | R/W | 0x020 | Baud Rate Generator register. 16-bit integer baud rate divisor value. | 0 | Table 184 |
| INTSTAT | R | 0x024 | Interrupt status register. Reflects interrupts that are currently enabled. | 0x0005 | Table 185 |
| OSR | R/W | 0x028 | Oversample selection register for asynchronous communication. | 0xF | Table 186 |
| ADDR | R/W | 0x02C | Address register for automatic address matching. | 0 | Table 187 |

### 13.6.1 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:** 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 176. USART Configuration register (CFG, address 0x4006 4000 (USART0), 0x4006 8000 (USART1)) 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 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) 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. | |
| 8:7 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

**Table 176. USART Configuration register (CFG, address 0x4006 4000 (USART0), 0x4006 8000 (USART1)) bit description** *…continued*

| Bit | Symbol | Value | Description | Reset Value |
|---|---|---|---|---|
| 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 is selected. | |
| | | 1 | Synchronous mode is selected. | |
| 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. | |
| 17:16 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 18 | OETA | | Output Enable Turnaround time enable for RS-485 operation. | 0 |
| | | 0 | De-asserted. If selected by OESEL, the Output Enable signal de-asserted at the end of the last stop bit of a transmission. | |
| | | 1 | Asserted. If selected by OESEL, the Output Enable signal remains asserted for 1 character time after then end the last stop bit of a transmission. OE will also remain asserted if another transmit begins before it is de-asserted. | |
| 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 176. USART Configuration register (CFG, address 0x4006 4000 (USART0), 0x4006 8000 (USART1)) bit description** *...continued*

| Bit | Symbol | Value | Description | Reset Value |
|---|---|---|---|---|
| 20 | OESEL | | Output Enable Select. | 0 |
| | | 0 | Flow control. The RTS signal is used as the standard flow control function. | |
| | | 1 | Output enable. The RTS signal is taken over in order 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 | Not changed. 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 UART. 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 | Not changed. 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 UART 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 |

### 13.6.2 USART Control register

The CTL register controls aspects of USART operation that are more likely to change during operation.

**Table 177. USART Control register (CTL, address 0x4006 4004 (USART0), 0x4006 8004 (USART1)) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 1 | TXBRKEN | | Break Enable. | 0 |
| | | 0 | Normal operation. | |
| | | 1 | Continuous break is sent immediately when this bit is set, and remains until this bit is cleared.<br>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. | |

**Table 177. USART Control register (CTL, address 0x4006 4004 (USART0), 0x4006 8004 (USART1)) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2 | ADDRDET | | Enable address detect mode. | 0 |
| | | 0 | Disabled. The USART presents all incoming data. | |
| | | 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 | | Transmit Disable. | 0 |
| | | 0 | Not disabled. USART transmitter is not disabled. | |
| | | 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 | | Continuous Clock generation. By default, SCLK is only output while data is being transmitted in synchronous mode. | 0 |
| | | 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. | |
| | | 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 | | Clear Continuous Clock. | 0 |
| | | 0 | No effect on the CC bit. | |
| | | 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 | | Autobaud enable. | 0 |
| | | 0 | Disabled. UART is in normal operating mode. | |
| | | 1 | Enabled. UART is in autobaud mode. This bit should only be set when the UART is enabled in the CFG register and the UART receiver is idle. The first start bit of RX is measured and used the update the BRG register to match the received data rate. AUTOBAUD is cleared once this process is complete, or if there is an ABERR. This bit can be cleared by software when set, but only when the UART receiver is idle. Disabling the UART in the CFG register also clears the AUTOBAUD bit. | |
| 31:17 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

### 13.6.3 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 180).

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 178. USART Status register (STAT, address 0x4006 4008 (USART0), 0x4006 8008 (USART1)) 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 Interrupt flag. When 1, this bit indicates that the USART transmitter is fully idle after being disabled via the TXDIS in the CTL 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 |
| 14 | PARITYERRINT | Parity Error interrupt flag. This flag is set when a parity error is detected in a received character.. | 0 | W1 |

**Table 178. USART Status register (STAT, address 0x4006 4008 (USART0), 0x4006 8008 (USART1)) bit description**

| Bit | Symbol | Description | Reset value | Access [1] |
|---|---|---|---|---|
| 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 | Autobaud Error. An autobaud error can occur if the BRG counts to its limit before the end of the start bit that is being measured, essentially an autobaud 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.

### 13.6.4 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 179. USART Interrupt Enable read and set register (INTENSET, address 0x4006 400C (USART0), 0x4006 800C (USART1)) 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 179. USART Interrupt Enable read and set register (INTENSET, address 0x4006 400C (USART0), 0x4006 800C (USART1)) bit description** *…continued*

| Bit | Symbol | Description | Reset Value |
|---|---|---|---|
| 15 | RXNOISEEN | When 1, enables an interrupt when noise is detected. | 0 |
| 16 | ABERREN | When 1, enables an interrupt when an autobaud error occurs. | 0 |
| 31:17 | - | Reserved. Read value is undefined, only zero should be written. | NA |

### 13.6.5 USART Interrupt Enable Clear register

The INTENCLR register is used to clear bits in the INTENSET register.

**Table 180. USART Interrupt Enable clear register (INTENCLR, address 0x4006 4010 (USART0), 0x4006 8010 (USART1)) 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 | TXDISINTCLR | 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 |

### 13.6.6 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.

**Table 181. USART Receiver Data register (RXDAT, address 0x4006 4014 (USART0), 0x4006 8014 (USART1)) bit description**

| Bit | Symbol | Description | Reset Value |
|---|---|---|---|
| 8:0 | RXDAT | 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 |

### 13.6.7 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.

**Remark:** Reading this register changes the status flags.

**Table 182. USART Receiver Data with Status register (RXDATSTAT, address 0x4006 4018 (USART0), 0x4006 8018 (USART1)) bit description**

| Bit | Symbol | Description | Reset Value |
|---|---|---|---|
| 8:0 | RXDAT | 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 178. | 0 |
| 31:16 | - | Reserved, the value read from a reserved bit is not defined. | NA |

### 13.6.8 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.

**Table 183. USART Transmitter Data Register (TXDAT, address 0x4006 401C (USART0), 0x4006 801C (USART1)) bit description**

| Bit | Symbol | Description | Reset Value |
|---|---|---|---|
| 8:0 | TXDAT | 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 |

### 13.6.9 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.

Details on how to select the right values for BRG can be found in Section 13.7.1.

**Remark:** If software needs to change the baud rate, 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 registers). 3) Write the new BRGVAL. 4) Write to the CFG register to set the Enable bit to 1.

**Table 184. USART Baud Rate Generator register (BRG, address 0x4006 4020 (USART0), 0x4006 8020 (USART1)) 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.<br>0 = The FRG clock is used directly by the USART function.<br>1 = The FRG clock is divided by 2 before use by the USART function.<br>2 = The FRG clock is divided by 3 before use by the USART function.<br>...<br>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 |

### 13.6.10 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 178 for detailed descriptions of the interrupt flags.

**Table 185. USART Interrupt Status register (INTSTAT, address 0x4006 4024 (USART0), 0x4006 8024 (USART1)) 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. | 1 |
| 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 | ABERR | Autobaud Error flag. | 0 |
| 31:17 | - | Reserved. Read value is undefined, only zero should be written. | NA |

### 13.6.11 USART 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 UART 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 186. USART Oversample selection register (OSR, address 0x4006 4028 (USART0), 0x4006 4028 (USART1)) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 3:0 | OSRVAL | Oversample Selection Value.<br>0 to 3 = not supported<br>0x4 = 5 peripheral clocks are used to transmit and receive each data bit.<br>0x5 = 6 peripheral clocks are used to transmit and receive each data bit.<br>...<br>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 |

### 13.6.12 USART Address register

The ADDR register holds the address for hardware address matching in address detect mode with automatic address matching enabled.

**Table 187. USART Address register (ADDR, address 0x4006 402C (USART0), 0x4006 802C (USART1)) 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 |

## 13.7 Functional description

### 13.7.1 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 20.

#### 13.7.1.1 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 inputs clock divided by 1 + (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 details see Section 13.3.1 “Configure the USART clock and baud rate”.

#### 13.7.1.2 Baud Rate Generator (BRG)

The Baud Rate Generator (see Section 13.6.9) 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.

#### 13.7.1.3 Baud rate calculations

Base clock rates are 16x for asynchronous mode and 1x for synchronous mode.

### 13.7.2 Synchronous mode

**Remark:** Synchronous mode transmit and receive operate at the incoming clock rate in slave mode and the BRG selected rate (not divided by 16) in master mode.

### 13.7.3 Flow control

The USART supports both hardware and software flow control.

#### 13.7.3.1 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.

If connected to a pin, and if enabled to do so, the CTS input can allow an external device to throttle the USART transmitter.

Figure 22 shows an overview of RTS and CTS within the USART.

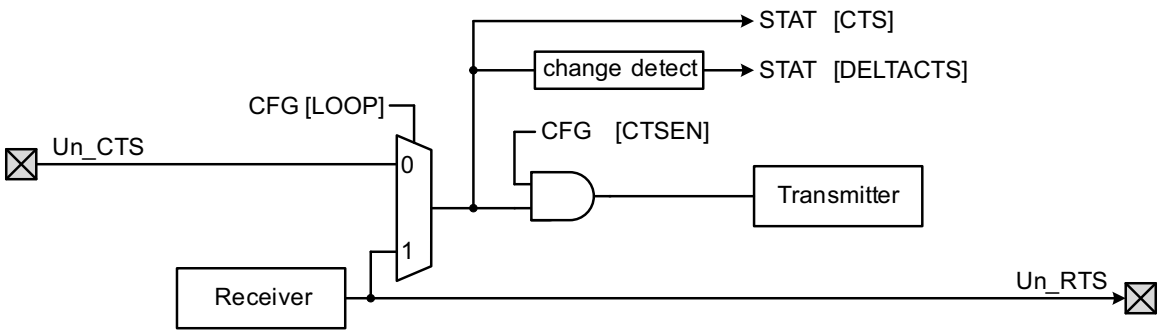



**Fig 22. Hardware flow control using RTS and CTS**

#### 13.7.3.2 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).

### 13.7.4 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. The assertion of RXRDY clears the AUTOBAUD bit automatically. The assertion of ABERR clears the AUTOBAUD bit once the receive line goes inactive.

Autobaud has no meaning, and should not be enabled, if the USART is in synchronous mode.

**Remark:** Before using autobaud, set the BRG register to 0x0 (this is the default). This setting allows the autobaud function to handle all baud rates.

### 13.7.5 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 13.6.1 and the ADDR register in Section 13.6.12), as well as software address recognition (see the ADDRDET bit in the CTL register in Section 13.6.2).

Automatic data direction control with the RTS pin can be set up using the OESEL, OEPOL, and OETA bits in the CFG register (Section 13.6.1). Data direction control can also be implemented in software using a GPIO pin.

### 13.7.6 Oversampling

Typical industry standard UARTs 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 UART 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 UART 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.

## 14.1 How to read this chapter

The SPI interface is available on all parts.

## 14.2 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 two Slave Select input/outputs with selectable polarity and flexible usage.

**Remark:** Texas Instruments SSI and National Microwire modes are not supported.

## 14.3 Basic configuration

Configure SPI using the following registers:

- In the SYSAHBCLKCTRL register, set bit 11 (Table 64) to enable the clock to the register interface.
- Clear the SPI peripheral resets using the PRESETCTRL register (Table 66).
- Enable/disable the SPI interrupts in interrupt slot #0 in the NVIC.
- Configure the SPI pin functions through the switch matrix. See Section 14.4.
- Using the SPI0CLKSEL register, the peripheral clock sources for the SPI can be FRO, main_clk, frg0clk, and fro_div. See Figure 6 “LPC804 clock generation aaa-get number”.

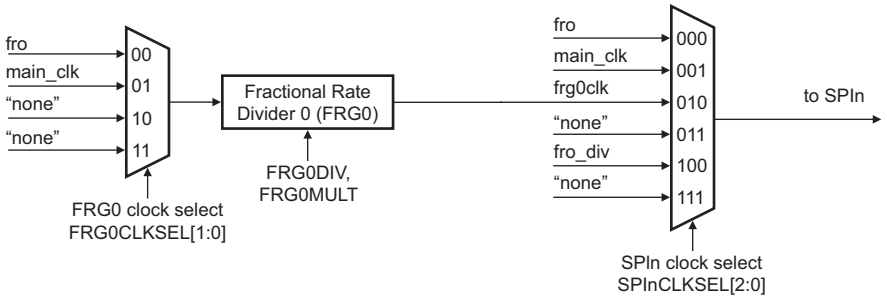



**Fig 23. SPI clocking**

### 14.3.1 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.

#### 14.3.1.1 Wake-up from sleep mode

- Configure the SPI in either master or slave mode. See Table 190.
- 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 193).

#### 14.3.1.2 Wake up from deep-sleep or power-down mode

- Configure the SPI in slave mode. See Table 190. You must connect the SCK function to a pin and connect the pin to the master.
- Enable the SPI interrupt in the STARTERP1 register. See Table 81 "Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description".
- In the PDAWAKE register, configure all peripherals that need to be running when the part wakes up.
- Enable the SPI interrupt in the NVIC.
- Enable the interrupt in the INTENSET register which configures the interrupt as wake-up event (Table 193). Examples are the following wake-up events:
  - A change in the state of the SSEL pins.
  - Data available to be received.
  - Receiver overrun.

## 14.4 Pin description

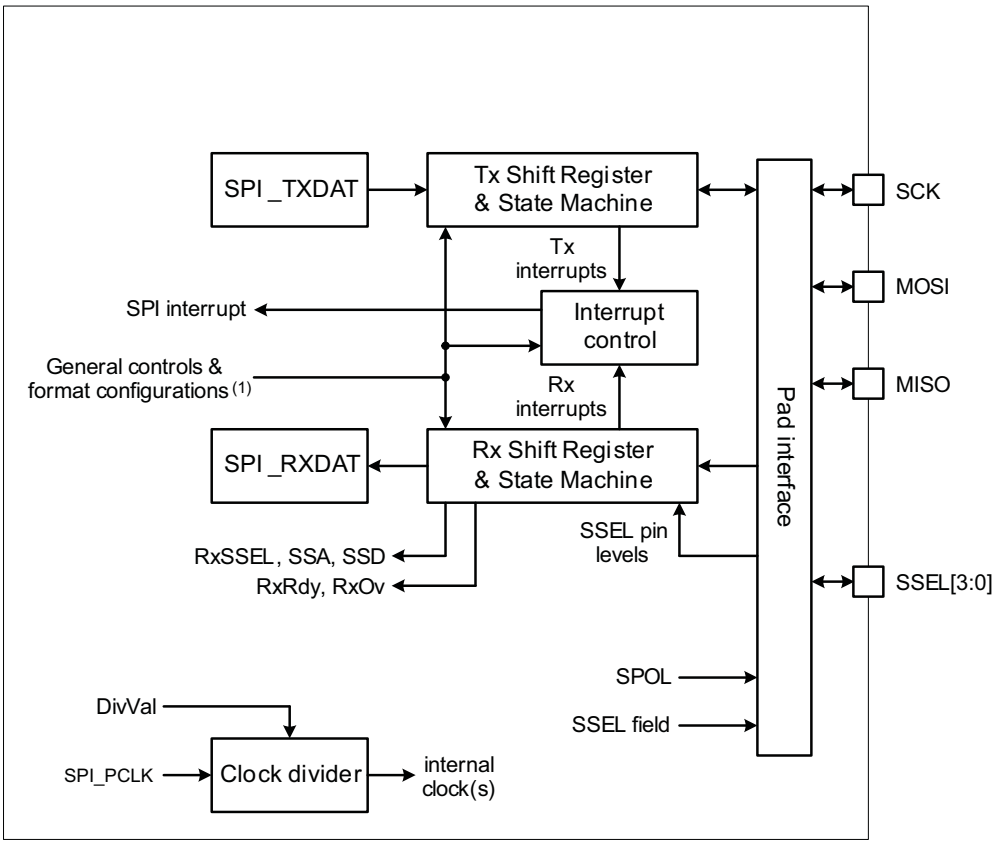
The SPI signals are movable functions and are assigned to external pins through the switch matrix.

See Section 8.3.1 "Connect an internal signal to a package pin" to assign the SPI functions to pins on the part.

**Table 188. SPI Pin Description**

| Function | I/O | Type | Connect to | Use register | Reference | Description |
|---|---|---|---|---|---|---|
| SPI0_SCK | I/O | external to pin | any pin | PINASSIGN2 | Table 93 | 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 | external to pin | any pin | PINASSIGN2 | Table 93 | 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 | external to pin | any pin | PINASSIGN2 | Table 93 | 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 | external to pin | any pin | PINASSIGN2 | Table 93 | 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 | external to pin | any pin | PINASSIGN3 | Table 94 | Slave Select 1. |

## 14.5 General description




(1) Includes CPOL, CPHA, LSBF, FLEN, master, enable, transfer_delay, frame_delay, pre_delay, post_delay, SOT, EOT, EOF, RXIGNORE, individual interrupt enables.

**Fig 24. SPI block diagram**

## 14.6 Register description

The reset value reflects the data stored in used bits only. It does not include reserved bits content.

**Table 189. Register overview: SPI (base address 0x4005 8000 (SPI)**

| Name | Access | Offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| CFG | R/W | 0x000 | SPI Configuration register | 0 | Table 190 |
| DLY | R/W | 0x004 | SPI Delay register | 0 | Table 191 |
| STAT | R/W | 0x008 | SPI Status. Some status flags can be cleared by writing a 1 to that bit position | 0x0102 | Table 192 |
| INTENSET | R/W | 0x00C | 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 193 |

**Table 189. Register overview: SPI (base address 0x4005 8000 (SPI)**

| Name | Access | Offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| INTENCLR | W | 0x010 | SPI Interrupt Enable Clear. Writing a 1 to any implemented bit position causes the corresponding bit in INTENSET to be cleared. | NA | Table 194 |
| RXDAT | R | 0x014 | SPI Receive Data | NA | Table 195 |
| TXDATCTL | R/W | 0x018 | SPI Transmit Data with Control | 0 | Table 196 |
| TXDAT | R/W | 0x01C | SPI Transmit Data | 0 | Table 197 |
| TXCTL | R/W | 0x020 | SPI Transmit Control | 0 | Table 198 |
| DIV | R/W | 0x024 | SPI clock Divider | 0 | Table 199 |
| INTSTAT | R | 0x028 | SPI Interrupt Status | 0x02 | Table 200 |

### 14.6.1 SPI Configuration register

The CFG register contains information for the general configuration of the SPI. Typically, this information is not changed during operation. Some configurations, such as CPOL, CPHA, and LSBF should not be made while the SPI is not fully idle. See the description of the master idle status (MSTIDLE in Table 192) for more information.

**Remark:** If the interface is re-configured from Master mode to Slave mode or the reverse (an unusual case), the SPI should be disabled and re-enabled with the new configuration.

**Table 190. SPI Configuration register (CFG, addresses 0x4005 8000 (SPI) 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. | |
| 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 |

**Table 190. SPI Configuration register (CFG, addresses 0x4005 8000 (SPI) bit description** *...continued*

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 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. The value in the SSEL0 fields of the RXDAT, TXDATCTL, and TXCTL registers related to SSEL0 is not inverted relative to the pins. | |
| | | 1 | High. The SSEL0 pin is active high. The value in the SSEL0 fields of the RXDAT, TXDATCTL, and TXCTL registers related to SSEL0 is inverted relative to the pins. | |
| 9 | SPOL1 | | SSEL1 Polarity select. | 0 |
| | | 0 | Low. The SSEL1 pin is active low. The value in the SSEL1 fields of the RXDAT, TXDATCTL, and TXCTL registers related to SSEL1 is not inverted relative to the pins. | |
| | | 1 | High. The SSEL1 pin is active high. The value in the SSEL1 fields of the RXDAT, TXDATCTL, and TXCTL registers related to SSEL1 is inverted relative to the pins. | |
| 31:10 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

### 14.6.2 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 14.7.2.1 “Pre_delay and Post_delay”

Section 14.7.2.2 “Frame_delay”

Section 14.7.2.3 “Transfer_delay”

**Table 191. SPI Delay register (DLY, addresses 0x4005 8004 (SPI) 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.<br>There is always one SPI clock time between SSEL assertion and the first clock edge. This is not considered part of the pre-delay.<br>0x0 = No additional time is inserted.<br>0x1 = 1 SPI clock time is inserted.<br>0x2 = 2 SPI clock times are inserted.<br>...<br>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.<br>0x0 = No additional time is inserted.<br>0x1 = 1 SPI clock time is inserted.<br>0x2 = 2 SPI clock times are inserted.<br>...<br>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).<br>0x0 = No additional time is inserted.<br>0x1 = 1 SPI clock time is inserted.<br>0x2 = 2 SPI clock times are inserted.<br>...<br>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.<br>0x0 = The minimum time that SSEL is deasserted is 1 SPI clock time. (Zero added time.)<br>0x1 = The minimum time that SSEL is deasserted is 2 SPI clock times.<br>0x2 = The minimum time that SSEL is deasserted is 3 SPI clock times.<br>...<br>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 |

### 14.6.3 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 192. SPI Status register (STAT, addresses 0x4005 8008 (SPI) 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 |

**Table 192. SPI Status register (STAT, addresses 0x4005 8008 (SPI) bit description**

| Bit | Symbol | Description | Reset value | Access [1] |
|---|---|---|---|---|
| 7 | ENDTRANSFER | 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.

### 14.6.4 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 192 for details of the interrupts.

**Table 193. SPI Interrupt Enable read and Set register (INTENSET, addresses 0x4005 800C (SPI) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | RXRDYEN | | Determines whether an interrupt occurs when receiver data is available. | 0 |
| | | 0 | No interrupt will be generated when receiver data is available. | |
| | | 1 | An interrupt will be generated when receiver data is available in the RXDAT register. | |
| 1 | TXRDYEN | | Determines whether an interrupt occurs when the transmitter holding register is available. | 0 |
| | | 0 | No interrupt will be generated when the transmitter holding register is available. | |
| | | 1 | An interrupt will be generated when data may be written to TXDAT. | |
| 2 | RXOVEN | | 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.<br>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 | No interrupt will be generated when a receiver overrun occurs. | |
| | | 1 | An interrupt will be generated if a receiver overrun occurs. | |
| 3 | TXUREN | | 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 | No interrupt will be generated when the transmitter underruns. | |
| | | 1 | An interrupt will be generated if the transmitter underruns. | |

**Table 193. SPI Interrupt Enable read and Set register (INTENSET, addresses 0x4005 800C (SPI) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 4 | SSAEN | | Determines whether an interrupt occurs when the Slave Select is asserted. | 0 |
| | | 0 | No interrupt will be generated when any Slave Select transitions from deasserted to asserted. | |
| | | 1 | An interrupt will be generated when any Slave Select transitions from deasserted to asserted. | |
| 5 | SSDEN | | Determines whether an interrupt occurs when the Slave Select is deasserted. | 0 |
| | | 0 | No interrupt will be generated when all asserted Slave Selects transition to deasserted. | |
| | | 1 | 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. | |
| 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 |

### 14.6.5 SPI Interrupt Enable Clear register

The INTENCLR register is used to clear interrupt enable bits in the INTENSET register.

**Table 194. SPI Interrupt Enable clear register (INTENCLR, addresses 0x4005 8010 (SPI) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 0 | RXRDYEN | Writing 1 clears the corresponding bits in the INTENSET register. | 0 |
| 1 | TXRDYEN | Writing 1 clears the corresponding bits in the INTENSET register. | 0 |
| 2 | RXOVEN | Writing 1 clears the corresponding bits in the INTENSET register. | 0 |
| 3 | TXUREN | Writing 1 clears the corresponding bits in the INTENSET register. | 0 |
| 4 | SSAEN | Writing 1 clears the corresponding bits in the INTENSET register. | 0 |
| 5 | SSDEN | Writing 1 clears the corresponding bits 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 |

### 14.6.6 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 14.7.4.

**Table 195. SPI Receiver Data register (RXDAT, addresses 0x4005 8014 (SPI) 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 |

### 14.6.7 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.

**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 14.6.8) instead of the TXDATCTL register. Control information can then be written separately via the TXCTL register (see Section 14.6.9). 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 14.7.4.

For details on using multiple consecutive data transmits for transfer lengths larger than 16 bit, see Section 14.7.5 “Data lengths greater than 16 bits”.

**Table 196. SPI Transmitter Data and Control register (TXDATCTL, addresses 0x4005 8018 (SPI) 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.<br>**Remark:** The active state of the SSEL0 pin is configured by bits in the CFG register. | 0 |
| | | 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.<br>**Remark:** The active state of the SSEL1 pin is configured by bits in the CFG register. | 0 |
| | | 0 | SSEL1 asserted. | |
| | | 1 | SSEL1 not asserted. | |
| 19:18 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 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. | |

**Table 196. SPI Transmitter Data and Control register (TXDATCTL, addresses 0x4005 8018 (SPI) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 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. | 0 |
| | | 0 | Read received data. Received data must be read first and then TXDATA should be written to allow transmission to progress for non DMA cases. 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.<br>0x0 = Data transfer is 1 bit in length.<br>0x1 = Data transfer is 2 bits in length.<br>0x2 = Data transfer is 3 bits in length.<br>...<br>0xF = Data transfer is 16 bits in length. | 0x0 |
| 31:28 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

### 14.6.8 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 14.6.7). That data will be sent to the transmit shift register when it is available, and another character may then be written to TXDAT.

**Table 197. SPI Transmitter Data Register (TXDAT, addresses 0x4005 801C (SPI) 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 |

### 14.6.9 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 14.6.7). 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 14.6.7) instead of TXDAT. Control information can then be written along with data.

**Table 198. SPI Transmitter Control register (TXCTL, addresses 0x4005 8020 (SPI) 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 |
| 19:18 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 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 |

### 14.6.10 SPI Divider register

The DIV register determines the clock used by the SPI in master mode.

For details on clocking, see Section 14.7.3 “Clocking and data rates”.

**Table 199. SPI Divider register (DIV, addresses 0x4005 8024 (SPI) 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.<br>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 |

### 14.6.11 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 192 for detailed descriptions of the interrupt flags.

**Table 200. SPI Interrupt Status register (INTSTAT, addresses 0x4005 8028 (SPI) 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 | Writing 1 clears the corresponding bit in the INTENSET register | 0 |
| 31:9 | - | Reserved. Read value is undefined, only zero should be written. | NA |

## 14.7 Functional description

### 14.7.1 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 201 and shown in Figure 25. CPOL and CPHA are configured by bits in the CFG register (Section 14.6.1).

**Table 201. 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 |

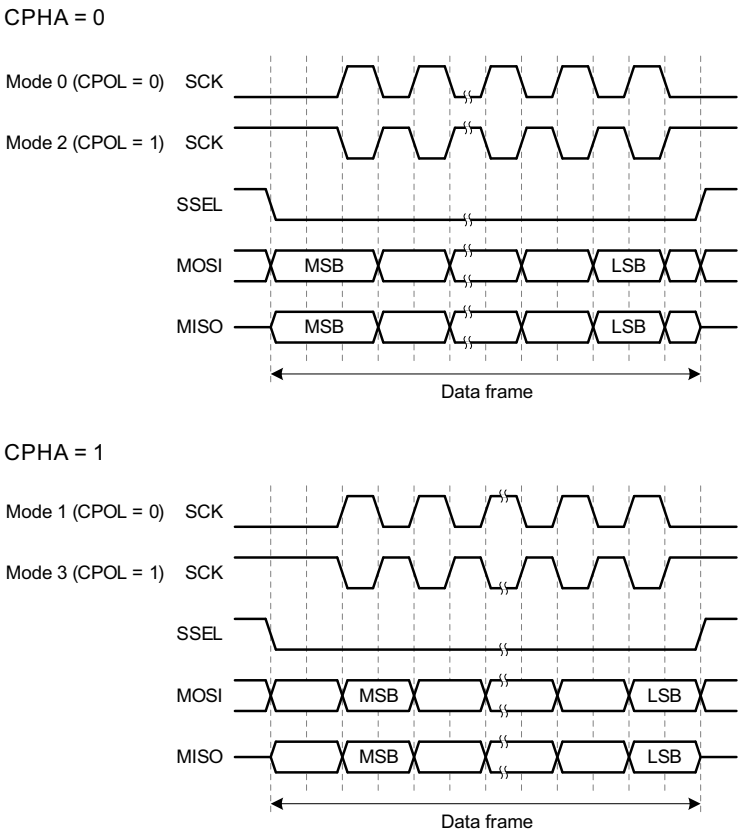



**Fig 25. Basic SPI operating modes**

### 14.7.2 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 de-asserted.
- Frame_delay: delay between data frames when SSEL is not de-asserted.
- Transfer_delay: minimum duration of SSEL in the de-asserted state between transfers.

#### 14.7.2.1 Pre_delay and Post_delay

Pre_delay and Post_delay are illustrated by the examples in Figure 26. 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 de-assertion of SSEL.

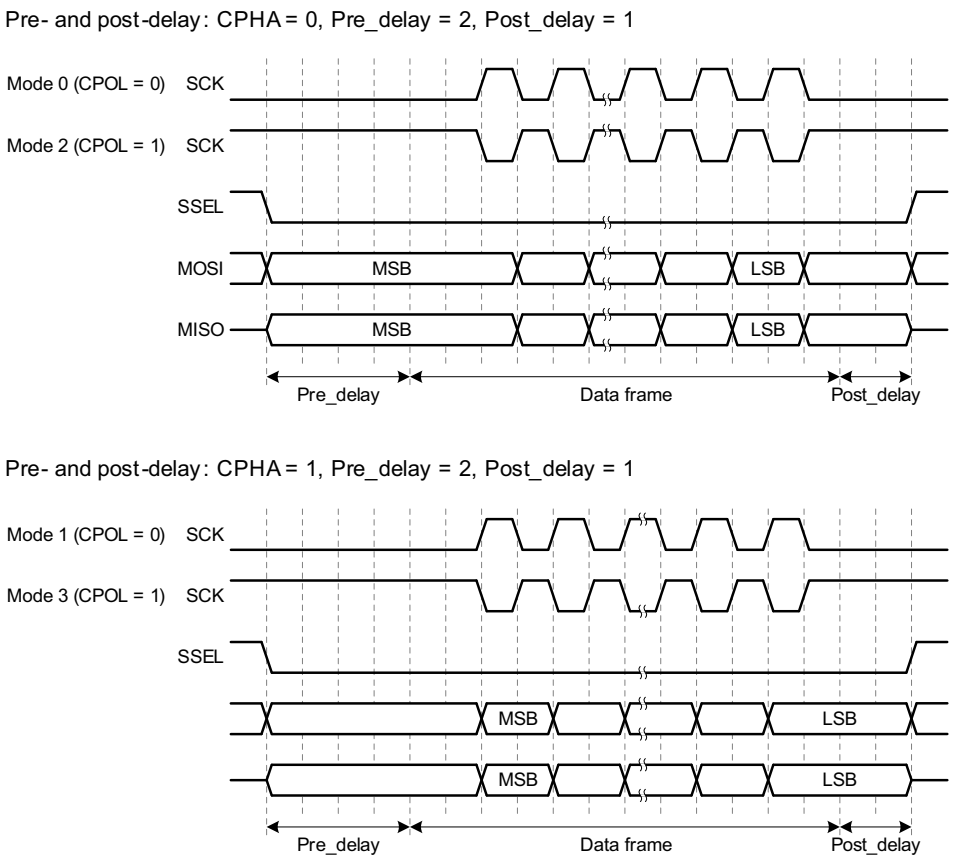



**Fig 26. Pre_delay and Post_delay**

### 14.7.2.2 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 27. Note that frame boundaries occur only where specified. This is because frame lengths can be any size, involving multiple data writes. See Section 14.7.5 for more information.

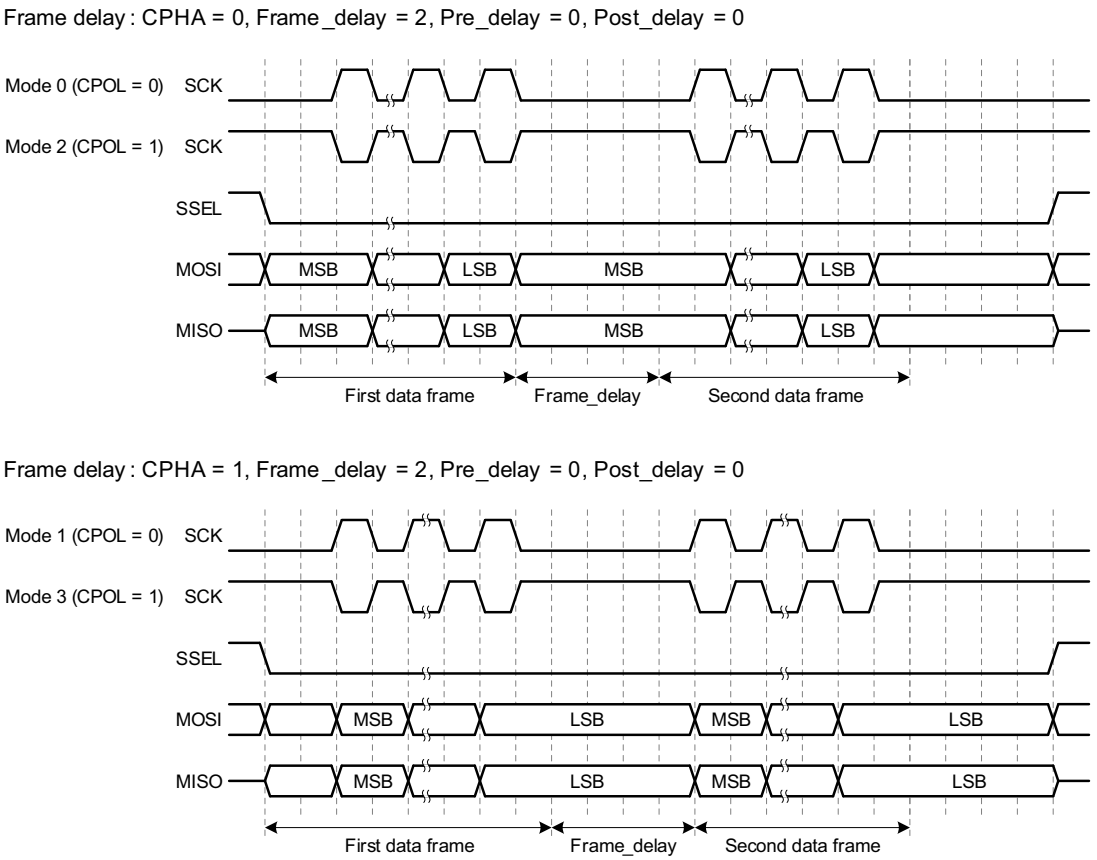



**Fig 27. Frame_delay**

### 14.7.2.3 Transfer_delay

The Transfer_delay value controls the minimum amount of time that SSEL is de-asserted between transfers, because the EOT bit = 1. When Transfer_delay = 0, SSEL may be de-asserted for a minimum of one SPI clock time. Transfer_delay is illustrated by the examples in Figure 28.

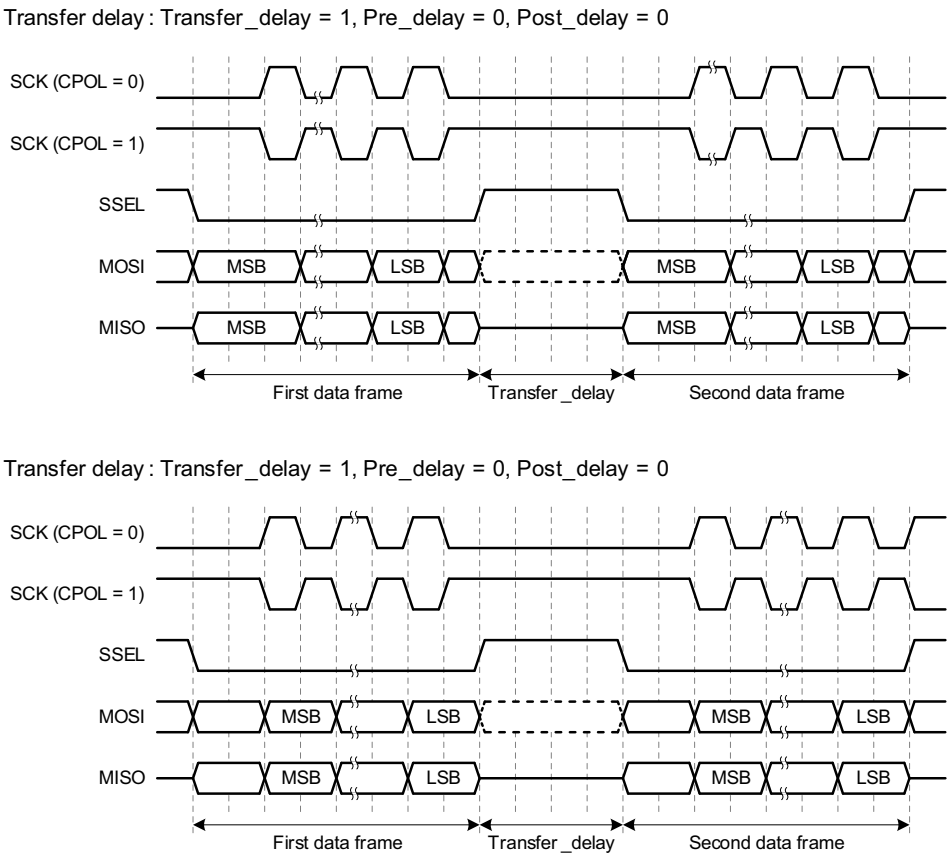



**Fig 28. Transfer_delay**

### 14.7.3 Clocking and data rates

In order to use the SPI, clocking details must be defined. This includes configuring the system clock, peripheral clock, and selection of the clock divider value in DIV. See Figure 23.

#### 14.7.3.1 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 FCLK, or at lower integer divide rates. The SPI rate will be = FCLK.

In slave mode, the clock is taken from the SCK input and the SPI clock divider is not used.

### 14.7.4 Slave select

The SPI block provides for two 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 two 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 SSELN field, which appears in both the CTL and DATCTL registers. In slave mode, the state of all two SSELs is saved along with received data in the RXSSEL_N field of the RXDAT register.

### 14.7.5 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 de-asserted 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 de-asserted between 24-bit increments, for instance, would require changing the value of the EOF bit on alternate 12-bit frames.

### 14.7.6 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.

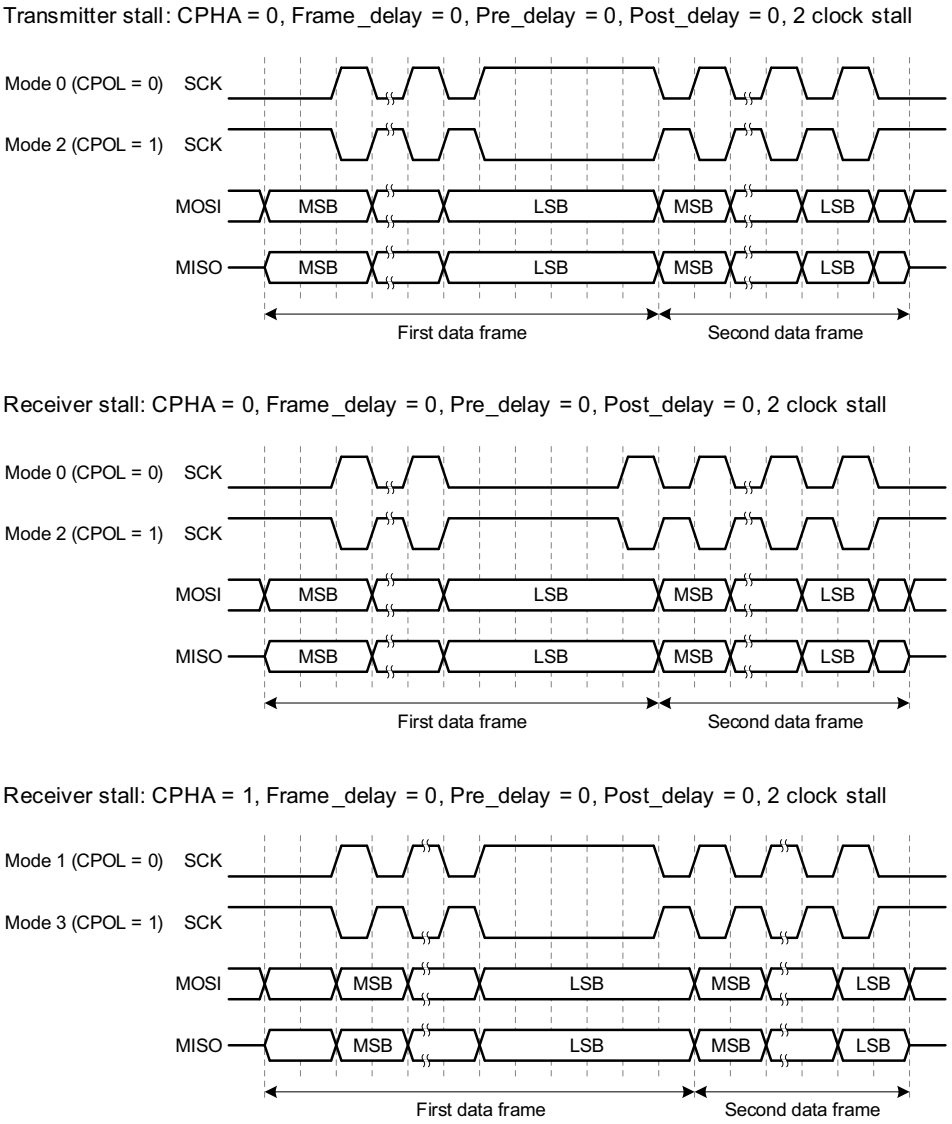



**Fig 29. Examples of data stalls**

## 15.1 How to read this chapter

Two I2C interfaces are available on all parts.

Read this chapter if you want to understand the I2C operation and the software interface and want to learn how to use the I2C for wake-up from reduced power modes.

## 15.2 Features

- Independent Master, Slave, and Monitor functions.
- Supports both Multi-master and Multi-master with Slave functions.
- Multiple $I^2C$ 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 $I^2C$ bus addresses.
- 10-bit addressing supported with software assist.
- Supports System Management (SMBus).
- Supports the $I^2C$-bus specification Fast-mode.

## 15.3 Basic configuration

Configure the $I^2C$ interfaces using the following registers:

- In the SYSAHBCLKCTRL register, set the corresponding bits to enable the clocks to the register interfaces. See Table 64.
- Clear the $I^2C$ peripheral resets using the PRESETCTRL register (Table 66).
- Enable/disable the $I^2C$ interrupt in interrupt slot #7 in the NVIC. See Table 38.
- Configure the $I^2C$ pin functions through the switch matrix. See Table 202.
- The peripheral clock for the $I^2C$ (see Figure 30).

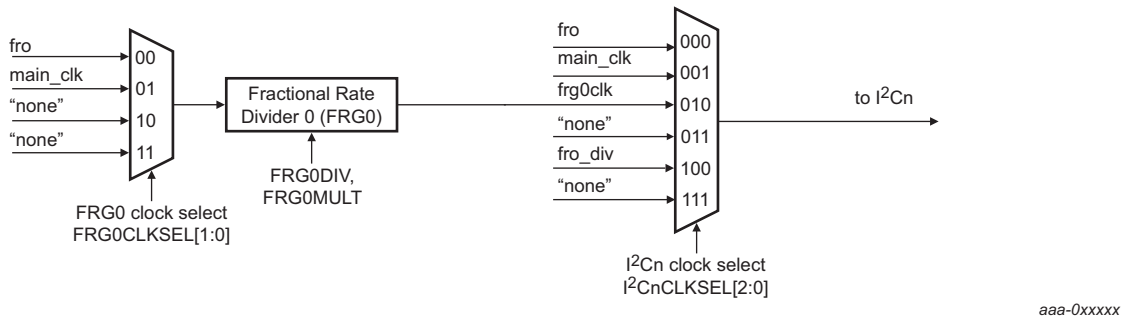



**Fig 30. I2C clocking**

### 15.3.1 I²C transmit/receive in master mode

In this example, the I²C 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 15 MHz and the bit rate is approximately 400 kHz. You can assign the SCL and SDA functions to pins through the switch matrix. See Table 202.

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:

- Select a source for the FCLK that will allow for the required I²C-bus rate. Divide the clock as needed. See Table 211 "I²C Clock Divider register (CLKDIV, address 0x4005 0014 (I2C0), 0x4005 4014 (I2C1)) bit description".
- Set the SCL high and low times to 2 clock cycles each. This is the default. See Table 214 "Master Time register (MSTTIME, address 0x4005 0024 (I²C0), 0x4005 4024 (I²C1)) bit description". The result is an SCL clock of 375 kHz.

#### 15.3.1.1 Master write to slave

Configure the I²C as master: Set the MSTEN bit to 1 in the CFG register. See Table 204.

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 215.
2. Start the transmission by setting the MSTSTART bit to 1 in the Master control register. See Table 213. 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 213. 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 213.

#### 15.3.1.2 Master read from slave

Configure the I²C as master: Set the MSTEN bit to 1 in the CFG register. See Table 204.

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 215.
2. Start the transmission by setting the MSTSTART bit to 1 in the Master control register. See Table 213. 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 213.

### 15.3.2 I2C receive/transmit in slave mode

In this example, the I2C is configured as the slave. The slave receives 8 bits from the master and sends 8 bits to the slave. The system clock is set to 15 MHz and the bit rate is approximately 400 kHz. You can assign the SCL and SDA functions to pins through the switch matrix. See Table 202.

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.

#### 15.3.2.1 Slave read from master

Configure the $I^2C$ as slave with address x:

- Set the SLVEN bit to 1 in the CFG register. See Table 204.
- Write the slave address x to the address 0 match register. See Table 218.

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 216.
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 217.
5. Acknowledge (“ack”) the data by setting SLVCONTINUE = 1 in the slave control register. See Table 216.

#### 15.3.2.2 Slave write to master

- Set the SLVEN bit to 1 in the CFG register. See Table 204.
- Write the slave address x to the address 0 match register. See Table 218.

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 216.
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 217.
5. Continue the transaction by setting SLVCONTINUE = 1 in the slave control register. See Table 216.

### 15.3.3 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 and 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 INTENCLR register of the INTENCLR register I²C block, can then wake up the core.

#### 15.3.3.1 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

#### 15.3.3.2 Wake-up from deep-sleep and power-down modes

- Enable the I²C interrupt in the NVIC.
- Enable the I²C interrupt in the STARTERP1 register in the SYSCON block to create the interrupt signal asynchronously while the core and the peripheral are not clocked. See Table 81 “Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description”.
- In the PDAWAKE register, configure all peripherals that need to be running when the part wakes up.
- 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

## 15.4 Pin description

Pins for the I²C interfaces are movable functions and can be assigned to any pin. However, the pins are not open-drain and do not support Fast-mode Plus mode. Bit rates of 400 kHz are supported on all pins.

**Table 202. I2C-bus pin description**

| Function | Direction | Type | Connect to | Use register | Reference | Description |
|---|---|---|---|---|---|---|
| I2C0_SDA | I/O | external to pin | any pin | PINASSIGN5 | Table 96 | I2C0 serial data. |
| I2C0_SCL | I/O | external to pin | any pin | PINASSIGN5 | Table 96 | I2C0 serial clock. |
| I2C1_SDA | I/O | external to pin | any pin | PINASSIGN7 | Table 98 | I2C1 serial data. |
| I2C1_SCL | I/O | external to pin | any pin | PINASSIGN7 | Table 98 | I2C1 serial clock. |

## 15.5 General description

The architecture of the I²C-bus interface is shown in Figure 31.

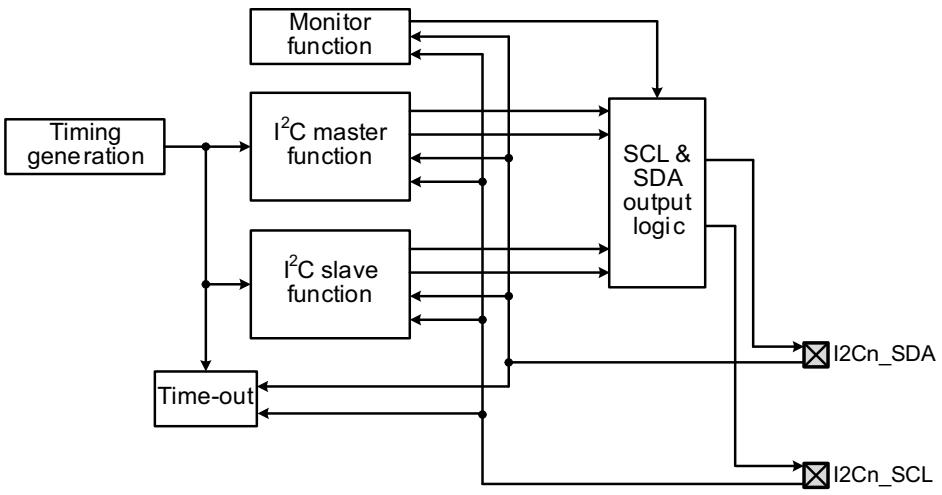



**Fig 31. I²C block diagram**

## 15.6 Register description

The register functions can be grouped as follows:

- Common registers:
  - Table 204 “I²C Configuration register (CFG, address 0x4005 0000 (I2C0), 0x4005 4000 (I2C1) bit description”
  - Table 205 “I²C Status register (STAT, address 0x4005 0004 (I2C0), 0x4005 4004 (I2C1)) bit description”

- Table 212 “I2C Interrupt Status register (INTSTAT, address 0x4005 0018 (I2C0), 0x4005 4018 (I2C1)) bit description”
- Table 208 “Interrupt Enable Set and read register (INTENSET, address 0x4005 0008 (I2C0), 0x4005 400C (I2C1)) bit description”
- Table 209 “Interrupt Enable Clear register (INTENCLR, address 0x4005 000C (I2C0), 0x4005 400C (I2C1)) bit description”
- Table 210 “Time-out value register (TIMEOUT, address 0x4005 0010 (I2C0), 0x4005 4010 (I2C1)) bit description”
- Table 211 “I2C Clock Divider register (CLKDIV, address 0x4005 0014 (I2C0), 0x4005 4014 (I2C1)) bit description”

- Master function registers:
  - Table 213 “Master Control register (MSTCTL, address 0x4005 0020 (I2C0), 0x4005 4020 (I2C1)) bit description”
  - Table 214 “Master Time register (MSTTIME, address 0x4005 0024 (I2C0), 0x4005 4024 (I2C1)) bit description”
  - Table 215 “Master Data register (MSTDAT, address 0x4005 0028 (I2C0), 0x4005 4028 (I2C1)) bit description”
- Slave function registers:
  - Table 216 “Slave Control register (SLVCTL, address 0x4005 0040 (I2C0), 0x4005 4040 (I2C1)) bit description”
  - Table 216 “Slave Control register (SLVCTL, address 0x4005 0040 (I2C0), 0x4005 4040 (I2C1)) bit description”
  - Table 218 “Slave Address registers (SLVADR[0:3], address 0x4005 0048 (SLVADR0) to 0x4005 0054 (SLVADR3) (I2C0), 0x4005 4048 (SLVADR0) to 0x4005 4054 (SLVADR3) (I2C1)) bit description”
  - Table 219 “Slave address Qualifier 0 register (SLVQUAL0, address 0x4005 0058 (I2C0), 0x4005 4058 (I2C1)) bit description”
- Monitor function register: Table 220 “Monitor data register (MONRXDAT, address 0x4005 0080 (I2C0), 0x4005 4080 (I2C1)) bit description”

**Table 203. Register overview: I²C (base address 0x4005 0000 (I2C0), 0x4005 4000 (I2C1)**

| Name | Access | Offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| CFG | R/W | 0x00 | Configuration for shared functions. | 0 | Table 204 |
| STAT | R/W | 0x04 | Status register for Master, Slave, and Monitor functions. | 0x00080 1 | Table 205 |
| INTENSET | R/W | 0x08 | Interrupt Enable Set and read register. | 0 | Table 208 |
| INTENCLR | W | 0x0C | Interrupt Enable Clear register. | NA | Table 209 |
| TIMEOUT | R/W | 0x10 | Time-out value register. | 0xFFFF | Table 210 |
| CLKDIV | R/W | 0x14 | Clock pre-divider for the entire I²C block. This determines what time increments are used for the MSTTIME register. | 0 | Table 211 |
| INTSTAT | R | 0x18 | Interrupt Status register for Master, Slave, and Monitor functions. | 0 | Table 212 |
| MSTCTL | R/W | 0x20 | Master control register. | 0 | Table 213 |
| MSTTIME | R/W | 0x24 | Master timing configuration. | 0x77 | Table 214 |
| MSTDAT | R/W | 0x28 | Combined Master receiver and transmitter data register. | NA | Table 215 |
| SLVCTL | R/W | 0x40 | Slave control register. | 0 | Table 216 |
| SLVDAT | R/W | 0x44 | Combined Slave receiver and transmitter data register. | NA | Table 217 |
| SLVADR0 | R/W | 0x48 | Slave address 0. | 0x01 | Table 218 |
| SLVADR1 | R/W | 0x4C | Slave address 1. | 0x01 | Table 218 |
| SLVADR2 | R/W | 0x50 | Slave address 2. | 0x01 | Table 218 |
| SLVADR3 | R/W | 0x54 | Slave address 3. | 0x01 | Table 218 |
| SLVQUAL0 | R/W | 0x58 | Slave Qualification for address 0. | 0 | Table 219 |
| MONRXDAT | RO | 0x80 | Monitor receiver data register. | 0 | Table 220 |

### 15.6.1 I²C Configuration register

The CFG register contains mode settings that apply to Master, Slave, and Monitor functions.

**Table 204. I²C Configuration register (CFG, address 0x4005 0000 (I2C0), 0x4005 4000 (I2C1) 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 I²C Master function is disabled. | |
| | | 1 | Enabled. The I²C 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 I²C slave function is disabled. | |
| | | 1 | Enabled. The I²C 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 I²C monitor function is disabled. | |
| | | 1 | Enabled. The I²C monitor function is enabled. | |

**Table 204. I²C Configuration register (CFG, address 0x4005 0000 (I2C0), 0x4005 4000 (I2C1) bit description**

| Bit | Symbol | Value | Description | Reset Value |
|---|---|---|---|---|
| 3 | TIMEOUTEN | | I²C 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 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 to ensure that software can read all incoming data supplied by the monitor function. | |
| 31:5 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

### 15.6.2 I²C Status register

The STAT register provides status flags and state information about all of the functions of the I²C block. Some information in this register is read-only and some flags can be cleared by writing a 1 to them.

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 206 and Table 207.

**Table 205. I²C Status register (STAT, address 0x4005 0004 (I2C0), 0x4005 4004 (I2C1)) bit description**

| Bit | Symbol | Value | Description | Reset value | Access |
|---|---|---|---|---|---|
| 0 | MSTPENDING | | Master Pending. Indicates that the Master is waiting to continue communication on the I2C-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. 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. | 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 | MSTARBLOSS | | 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 loss. No Arbitration Loss has occurred. | | |
| | | 1 | Arbitration loss. The Master function has experienced an Arbitration Loss.<br>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 205. I²C Status register (STAT, address 0x4005 0004 (I2C0), 0x4005 4004 (I2C1)) bit description** *…continued*

| Bit | Symbol | Value | Description | Reset value | Access |
|---|---|---|---|---|---|
| 6 | MSTSTSTPERR | | 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 | Start/stop error has occurred. The Master function has experienced a Start/Stop Error.<br>A Start or Stop was detected at a time when it is not allowed by the I²C 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 | SLVPENDING | | Slave Pending. Indicates that the Slave function is waiting to continue communication on the I2C-bus and needs software service. This flag will cause an interrupt when set if enabled via INTENSET. The SLVPENDING flag is read-only and is automatically cleared when a 1 is written to the SLVCONTINUE bit in the MSTCTL register. | 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. | 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). | | |
| | | 0x3 | Reserved. | | |
| 11 | SLVNOTSTR | | Slave Not Stretching. Indicates when the slave function is stretching the I²C 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 I²C 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 I²C bus clock. Deep-sleep or power-down mode could be entered at this time. | | |
| 13:12 | SLVIDX | | Slave address match Index. This field is valid when the I²C 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 | Slave address 0 was matched. | | |
| | | 0x1 | Slave address 1 was matched. | | |
| | | 0x2 | Slave address 2 was matched. | | |
| | | 0x3 | Slave address 3 was matched. | | |

**Table 205. I²C Status register (STAT, address 0x4005 0004 (I2C0), 0x4005 4004 (I2C1)) bit description** *…continued*

| Bit | Symbol | Value | Description | Reset value | Access |
|---|---|---|---|---|---|
| 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 | MONACTIVE | | Monitor Active flag. This flag indicates when the Monitor function considers the I²C 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 I²C bus to be inactive. | | |
| | | 1 | Active. The Monitor function considers the I²C bus to be active. | | |
| 19 | MONIDLE | | Monitor Idle flag. This flag is set when the Monitor function sees the I²C 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 I²C bus is not idle, or this flag has been cleared by software. | | |
| | | 1 | Idle. The I²C bus has gone idle at least once since the last time this flag was cleared by software. | | |
| 23:20 | - | | Reserved. Read value is undefined, only zero should be written. | NA | NA |

**Table 205. I²C Status register (STAT, address 0x4005 0004 (I2C0), 0x4005 4004 (I2C1)) bit description** *…continued*

| Bit | Symbol | Value | Description | Reset value | Access |
|---|---|---|---|---|---|
| 24 | EVENTTIMEOUT | | 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 I2C-bus is idle. | 0 | W1 |
| | | 0 | No time-out. I²C bus events have not caused a time-out. | | |
| | | 1 | Event time-out. The time between I²C bus events has been longer than the time specified by the I2C TIMEOUT register. | | |
| 25 | SCLTIMEOUT | | 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 206. Master function state codes (MSTSTATE)**

| MSTSTATE | Description | Actions |
|---|---|---|
| 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. |
| 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. |
| 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. |
| 0x3 | **Slave NACKed address.** | Send a Stop or Repeated Start. |
| 0x4 | **Slave NACKed transmitted data.** | Send a Stop or Repeated Start. |

**Table 207. Slave function state codes (SLVSTATE)**

| SLVSTATE | | Description | Actions |
|---|---|---|---|
| 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 15.7.3 regarding 10-bit addressing. |
| 1 | SLVST_RX | **Received data is available (Slave Receiver mode).** | Read data reply with an ACK or a NACK. |
| 2 | SLVST_TX | **Data can be transmitted (Slave Transmitter mode).** | Send data. |
| 3 | - | Reserved. | - |

### 15.6.3 Interrupt Enable Set and read register

The INTENSET register controls which $I^2C$ 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, if an interrupt is supported there. Reading INTENSET indicates which interrupts are currently enabled.

**Table 208. Interrupt Enable Set and read register (INTENSET, address 0x4005 0008 (I2C0), 0x4005 400C (I2C1)) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | MSTPENDINGEN | | Master Pending interrupt Enable. | 0 |
| | | 0 | The MstPending interrupt is disabled. | |
| | | 1 | 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 | The MstArbLoss interrupt is disabled. | |
| | | 1 | 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 | The MstStStpErr interrupt is disabled. | |
| | | 1 | 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 | The SlvPending interrupt is disabled. | |
| | | 1 | 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 | The SlvNotStr interrupt is disabled. | |
| | | 1 | 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 | The SlvDeSel interrupt is disabled. | |
| | | 1 | The SlvDeSel interrupt is enabled. | |
| 16 | MONRDYEN | | Monitor data Ready interrupt Enable. | 0 |
| | | 0 | The MonRdy interrupt is disabled. | |
| | | 1 | The MonRdy interrupt is enabled. | |
| 17 | MONOVEN | | Monitor Overrun interrupt Enable. | 0 |
| | | 0 | The MonOv interrupt is disabled. | |
| | | 1 | The MonOv interrupt is enabled. | |
| 18 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 19 | MONIDLEEN | | Monitor Idle interrupt Enable. | 0 |
| | | 0 | The MonIdle interrupt is disabled. | |
| | | 1 | The MonIdle interrupt is enabled. | |
| 23:20 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

**Table 208. Interrupt Enable Set and read register (INTENSET, address 0x4005 0008 (I2C0), 0x4005 400C (I2C1)) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 24 | EVENTTIMEOUTEN | | Event time-out interrupt Enable. | 0 |
| | | 0 | The Event time-out interrupt is disabled. | |
| | | 1 | The Event time-out interrupt is enabled. | |
| 25 | SCLTIMEOUTEN | | SCL time-out interrupt Enable. | 0 |
| | | 0 | The SCL time-out interrupt is disabled. | |
| | | 1 | The SCL time-out interrupt is enabled. | |
| 31:26 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

### 15.6.4 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 209. Interrupt Enable Clear register (INTENCLR, address 0x4005 000C (I2C0), 0x4005 400C (I2C1)) 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 | MSTARBLOSSCLR | Master Arbitration Loss interrupt clear. | 0 |
| 5 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 6 | MSTSTSTPERRCLR | 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 |
| 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 |

### 15.6.5 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 15.7.2 “Time-out”.

**Table 210. Time-out value register (TIMEOUT, address 0x4005 0010 (I2C0), 0x4005 4010 (I2C1)) 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 I2C function clocks and also a time-out resolution of 16 I2C function clocks. | 0xF |
| 15:4 | TO | Time-out time value. Specifies the time-out interval value in increments of 16 I2C function clocks, as defined by the CLKDIV register. To change this value while I2C is in operation, disable all time-outs, write a new value to TIMEOUT, then re-enable time-outs.<br>0x000 = A time-out will occur after 16 counts of the I2C function clock.<br>0x001 = A time-out will occur after 32 counts of the I2C function clock.<br>...<br>0xFFF = A time-out will occur after 65,536 counts of the I2C function clock. | 0xFFF |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | NA |

### 15.6.6 Clock Divider register

The CLKDIV register divides down the FCLK 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 MSTTIME register for Master operation.

See Section 15.7.1.1 “Rate calculations” for details on bus rate setup.

**Table 211. I²C Clock Divider register (CLKDIV, address 0x4005 0014 (I2C0), 0x4005 4014 (I2C1)) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 15:0 | DIVVAL | This field controls how the clock (FCLK) is used by the I²C functions that need an internal clock in order to operate.<br>0x0000 = FCLK is used directly by the I²C function.<br>0x0001 = FCLK is divided by 2 before use by the I²C function.<br>0x0002 = FCLK is divided by 3 before use by the I²C function.<br>...<br>0xFFFF = PCLK is divided by 65,536 before use by the I²C function. | 0 |
| 31:16 | - | Reserved. Read value is undefined, only zero should be written. | NA |

### 15.6.7 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 205 for detailed descriptions of the interrupt flags.

**Table 212. I²C Interrupt Status register (INTSTAT, address 0x4005 0018 (I²C0), 0x4005 4018 (I²C1)) 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 | MSTSTSTPERR | Master Start/Stop Error flag. | 0 |
| 7 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 8 | SLVPENDING | Slave Pending. | 0 |
| 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 |

### 15.6.8 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 205).

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.

**Table 213. Master Control register (MSTCTL, address 0x4005 0020 ($I^2C0$), 0x4005 4020 ($I^2C1$)) 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 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 $I^2C$ 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 $I^2C$ 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). | |
| 31:3 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

### 15.6.9 Master Time

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 211.

**Table 214. Master Time register (MSTTIME, address 0x4005 0024 ($I^2C0$), 0x4005 4024 ($I^2C1$)) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | MSTSCLLOW | | 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 $I^2C$ bus specification. $I^2C$ bus specification parameters $t_{BUF}$ and $t_{SU;STA}$ have the same values and are also controlled by MSTSCLLOW. | 0x7 |
| | | 0x0 | 2 clocks. Minimum SCL low time is 2 clocks of the $I^2C$ clock pre-divider. | |
| | | 0x1 | 3 clocks. Minimum SCL low time is 3 clocks of the $I^2C$ clock pre-divider. | |
| | | 0x2 | 4 clocks. Minimum SCL low time is 4 clocks of the $I^2C$ clock pre-divider. | |
| | | 0x3 | 5 clocks. Minimum SCL low time is 5 clocks of the $I^2C$ clock pre-divider. | |
| | | 0x4 | 6 clocks. Minimum SCL low time is 6 clocks of the $I^2C$ clock pre-divider. | |
| | | 0x5 | 7 clocks. Minimum SCL low time is 7 clocks of the $I^2C$ clock pre-divider. | |
| | | 0x6 | 8 clocks. Minimum SCL low time is 8 clocks of the $I^2C$ clock pre-divider. | |
| | | 0x7 | 9 clocks. Minimum SCL low time is 9 clocks of the $I^2C$ clock pre-divider. | |

**Table 214. Master Time register (MSTTIME, address 0x4005 0024 ($I^2C0$), 0x4005 4024 ($I^2C1$)) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 3 | - | | Reserved. | 0 |
| 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 $I^2C$ bus specification. $I^2C$ bus specification parameters $t_{SU;STO}$ and $t_{HD;STA}$ have the same values and are also controlled by MSTSCLHIGH. | 0x7 |
| | | 0x0 | 2 clocks. Minimum SCL high time is 2 clock of the $I^2C$ clock pre-divider. | |
| | | 0x1 | 3 clocks. Minimum SCL high time is 3 clocks of the $I^2C$ clock pre-divider . | |
| | | 0x2 | 4 clocks. Minimum SCL high time is 4 clock of the $I^2C$ clock pre-divider. | |
| | | 0x3 | 5 clocks. Minimum SCL high time is 5 clock of the $I^2C$ clock pre-divider. | |
| | | 0x4 | 6 clocks. Minimum SCL high time is 6 clock of the $I^2C$ clock pre-divider. | |
| | | 0x5 | 7 clocks. Minimum SCL high time is 7 clock of the $I^2C$ clock pre-divider. | |
| | | 0x6 | 8 clocks. Minimum SCL high time is 8 clock of the $I^2C$ clock pre-divider. | |
| | | 0x7 | 9 clocks. Minimum SCL high time is 9 clocks of the $I^2C$ clock pre-divider. | |
| 31:7 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

### 15.6.10 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 215. Master Data register (MSTDAT, address 0x4005 0028 ($I^2C0$), 0x4005 4028 ($I^2C1$)) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 7:0 | DATA | Master function data register.<br>Read: read the most recently received data for the Master function.<br>Write: transmit data using the Master function. | 0 |
| 31:8 | - | Reserved. Read value is undefined, only zero should be written. | NA |

### 15.6.11 Slave Control register

The SLVCTL register contains bits that control various functions of the $I^2C$ Slave interface. Only write to this register when the slave is pending (SLVPENDING = 1 in the STAT register, Table 205).

**Table 216. Slave Control register (SLVCTL, address 0x4005 0040 ($I^2C0$), 0x4005 4040 ($I^2C1$)) 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. This must done after writing transmit data, reading received data, or any other housekeeping related to the next bus operation. | |

**Table 216. Slave Control register (SLVCTL, address 0x4005 0040 ($I^2C0$), 0x4005 4040 ($I^2C1$)) bit description**

| Bit | Symbol | Value | Description | Reset Value |
|---|---|---|---|---|
| 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). | |
| 2 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 31:3 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

### 15.6.12 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 217. Slave Data register (SLVDAT, address 0x4005 0044 ($I^2C0$), 0x4005 4044 ($I^2C1$)) bit description**

| Bit | Symbol | Description | Reset Value |
|---|---|---|---|
| 7:0 | DATA | Slave function data register.<br>Read: read the most recently received data for the Slave function.<br>Write: transmit data using the Slave function. | 0 |
| 31:8 | - | Reserved. Read value is undefined, only zero should be written. | NA |

### 15.6.13 Slave Address registers

The SLVADR[0:3] registers allow enabling and defining one of the addresses that can be automatically recognized by the $I^2C$ slave hardware. The value in the SLVADR0 register is qualified by the setting of the SLVQUAL0 register.

When the slave address is compared to the receive address, the compare can be affected by the setting of the SLVQUAL0 register (see Section 15.6.14).

The $I^2C$ slave function has 4 address comparators. The additional 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 218. Slave Address registers (SLVADR[0:3], address 0x4005 0048 (SLVADR0) to 0x4005 0054 (SLVADR3) ($I^2C0$), 0x4005 4048 (SLVADR0) to 0x4005 4054 (SLVADR3) ($I^2C1$)) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | SADISABLE | | Slave Address n Disable. | 1 |
| | | 0 | Enabled. Slave Address n is enabled and will be recognized with any changes specified by the SLVQUAL0 register. | |
| | | 1 | Ignored Slave Address n is ignored. | |
| 7:1 | SLVADR | | 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 |

### 15.6.14 Slave address Qualifier 0 register

The SLVQUAL0 register can alter how Slave Address 0 is interpreted.

**Table 219. Slave address Qualifier 0 register (SLVQUAL0, address 0x4005 0058 ($I^2C0$), 0x4005 4058 ($I^2C1$)) bit description**

| Bit | Symbol | Value | Description | Reset Value |
|---|---|---|---|---|
| 0 | QUALMODE0 | | Reserved. Read value is undefined, only zero should be written. | 0 |
| | | 0 | The SLVQUAL0 field is used as a logical mask for matching address 0. | |
| | | 1 | 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.<br>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.<br>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] <= received address <= SLVQUAL0[7:1]). | 0 |
| 31:8 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

### 15.6.15 Monitor data register

The read-only MONRXDAT register provides information about events on the $I^2C$ bus, primarily to facilitate debugging of the $I^2C$ 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 $I^2C$ 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 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 $I^2C$ bus.

**Table 220. Monitor data register (MONRXDAT, address 0x4005 0080 ($I^2C0$), 0x4005 4080 ($I^2C1$)) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 7:0 | MONRXDAT | | Monitor function Receiver Data. This reflects every data byte that passes on the $I^2C$ pins, and adds indication of Start, Repeated Start, and data NACK. | 0 |
| 8 | MONSTART | | Monitor Received Start. | 0 |
| | | 0 | No detect. The monitor function has not detected a Start event on the $I^2C$ bus. | |
| | | 1 | Start detect. The monitor function has detected a Start event on the $I^2C$ bus. | |
| 9 | MONRESTART | | Monitor Received Repeated Start. | 0 |
| | | 0 | No start detect. The monitor function has not detected a Repeated Start event on the $I^2C$ bus. | |
| | | 1 | Repeated start detect. The monitor function has detected a Repeated Start event on the $I^2C$ bus. | |

**Table 220. Monitor data register (MONRXDAT, address 0x4005 0080 ($I^2C0$), 0x4005 4080 ($I^2C1$)) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 10 | MONNACK | | Monitor Received NACK. | 0 |
| | | 0 | Acknowledged. The data currently being provided by the monitor function was acknowledged by at least one master or slave receiver. | |
| | | 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 |

## 15.7 Functional description

### 15.7.1 Bus rates and timing considerations

Due to the nature of the $I^2C$ bus, it is generally not possible to guarantee a specific clock rate on the SCL pin. On the I2C-bus, the 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 (that is, when it is the only master on the bus, or during arbitration between masters).

Rate calculations give a base frequency that represents the fastest that the $I^2C$ bus could operate if nothing slows it down.

#### 15.7.1.1 Rate calculations

SCL high time (in $I^2C$ function clocks) = (CLKDIV + 1) * (MSTSCLHIGH + 2)

SCL low time (in $I^2C$ function clocks) = (CLKDIV + 1) * (MSTSCLLOW + 2)

Nominal SCL rate = $I^2C$ function clock rate / (SCL high time + SCL low time)

**Remark:** DIVVAL must be ≥ 1.

**Remark:** For 400 KHz clock rate, the clock frequency after the $I^2C$ divider (divval) must be ≤ 2 MHz. Table 221 shows the recommended settings for 400 KHz clock rate.

**Table 221. Settings for 400 KHz clock rate**

| Input clock to I2C | DIVVAL for CLKDIV register | MSTSCLHIGH for MSTTIME register | MSTSCLLOW for MSTTIME register |
|---|---|---|---|
| 30 MHz | 14 | 0 | 1 |
| 24 MHz | 14 | 0 | 0 |
| 24 MHz | 11 | 0 | 1 |
| 12 MHz | 5 | 0 | 1 |

### 15.7.2 Time-out

A time-out feature on an $I^2C$ 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 $I^2C$ 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 $I^2C$ 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 $I^2C$ bus within a system as part of a method to keep the bus running of problems occur.

The second type of $I^2C$ 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 $I^2C$ 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 when the $I^2C$ bus is considered busy.

### 15.7.3 Ten-bit addressing

Ten-bit addressing is accomplished by the $I^2C$ 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 $I^2C$ 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 $I^2C$ address.

For the Master function, the $I^2C$ is 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 15.6.14) 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.

### 15.7.4 Clocking and power considerations

The Master function of the $I^2C$ 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 $I^2C$ Slave function recognizes an address. Monitor mode can similarly wake up the device from a reduced power mode when information becomes available.

### 15.7.5 Interrupt handling

The I2C provides a single interrupt output that handles all interrupts for Master, Slave, and Monitor functions.

## 16.1 How to read this chapter

The standard timer is available on all LPC804 devices.

## 16.2 Features

- 32-bit counter/timer with a programmable 32-bit prescaler. The timer includes external capture and match pin connections.
- Counter or timer operation.
- Up to three 32-bit captures can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt. The number of capture inputs that are actually available on device pins may vary by device.
- 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.
  - Optional auto-reload from match shadow registers when counter is reset.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- Up to four external outputs corresponding to match registers with the following capabilities (the number of match outputs that are actually available on device pins may vary by device):
  - Set LOW on match.
  - Set HIGH on match.
  - Toggle on match.
  - Do nothing on match.
- Up to 4 match registers can be configured for PWM operation, allowing up to 3 single edged controlled PWM outputs. (The number of match outputs that are actually available on device pins may vary by device.)

## 16.3 Basic configuration

- Set the appropriate bit to enable the clock to timer: CTIMER in the AHBCLKCTRL0 register (Section 6.6.10).
- Clear the timer reset using the PRESETCTRL1 register (Table 66 for CTIMER). Note that the bit position in the reset control register matches the bit position in the clock control register.
- Pins: Select timer pins through switch matrix.

- Interrupts: See register MCR (Table 229) and CCR (Table 231) for match and capture events. Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register. For interrupt connections, see Table 38.

## 16.4 General description

The Counter/timer is designed to count cycles of the APB bus clock or an externally supplied clock and can optionally generate interrupts or perform other actions at specified timer values based on four match registers. The counter/timer also includes capture inputs 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. All match registers can optionally be auto-reloaded from a companion shadow register whenever the counter is reset to zero. This permits modifying the match values for the next counter cycle without risk of disrupting the PWM waveforms during the current cycle. When enabled, match reload will occur whenever the counter is reset either due to a match event or a write to bit 1 of the Timer Control Register (TCR).

### 16.4.1 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 APB bus clock. For more details see Section 16.6.11.

### 16.4.2 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.

### 16.4.3 Applications

- Interval timer for counting internal events.
- Pulse Width Modulator via match outputs.
- Pulse Width Demodulator via capture input.
- Free running timer.

### 16.4.4 Architecture

Figure 32 shows the block diagram for the timer.

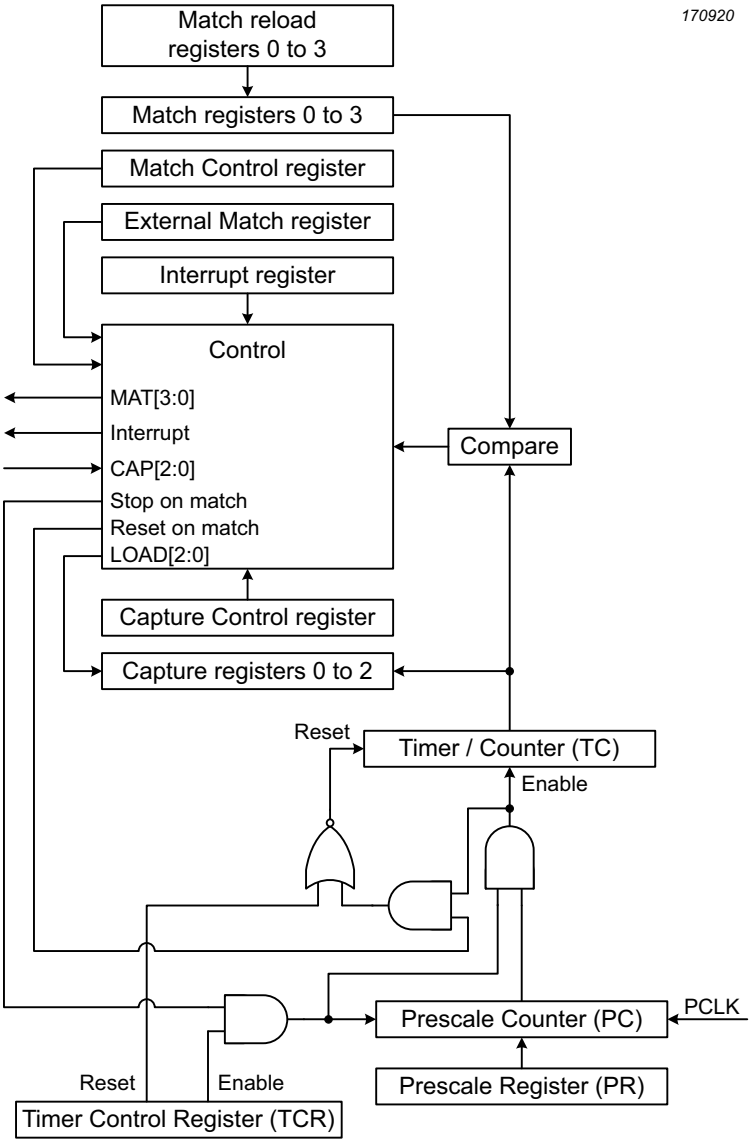



**Fig 32. 32-bit counter/timer block diagram**

## 16.5 Pin description

Table 222 gives a brief summary of each of the Timer/Counter related pins.

**Table 222. Timer/Counter pin description**

| Function | Type | Connect to | Use register | Description |
|---|---|---|---|---|
| TO_CAPx | Input | Any pin | PINASSIGN3 | 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.<br>Timer/Counter block can select a capture signal as a clock source instead of the APB bus clock. For more details see Section 16.6.11. |
| TO_MATx | Output | Any pin | PINASSIGN4 | 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. |

### 16.5.1 Multiple CAP and MAT pins

Software can select from multiple pins for the CAP or MAT functions in the switch matrix registers. Note that match conditions may be used internally without the use of a device pin.

## 16.6 Register description

The timer/counter contains the registers shown in Table 223.

**Table 223. Register overview: CTIMER (register base addresses 0x4003 8000)**

| Name | Access | Offset | Description | Reset value[1] | Section |
|---|---|---|---|---|---|
| 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 | 16.6.1 |
| 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 | 16.6.2 |
| TC | R/W | 0x08 | Timer Counter. The 32 bit TC is incremented every PR+1 cycles of the APB bus clock. The TC is controlled through the TCR. | 0 | 16.6.3 |
| 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 | 16.6.4 |
| 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 | 16.6.5 |
| MCR | R/W | 0x14 | The MCR is used to control whether an interrupt is generated, whether the TC is reset when a Match occurs, and whether the match register is reloaded from its shadow register when the TC is reset. | 0 | 16.6.6 |
| 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 | 16.6.7 |
| MR1 | R/W | 0x1C | Match Register 1. See MR0 description. | 0 | 16.6.7 |
| MR2 | R/W | 0x20 | Match Register 2. See MR0 description. | 0 | 16.6.7 |
| MR3 | R/W | 0x24 | Match Register 3. See MR0 description. | 0 | 16.6.7 |
| 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 | 16.6.8 |
| 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 | 16.6.9 |
| CR1 | RO | 0x30 | Capture Register 1. See CR0 description. | 0 | 16.6.9 |
| CR2 | RO | 0x34 | Capture Register 2. See CR0 description. | 0 | 16.6.9 |
| EMR | R/W | 0x3C | External Match Register. The EMR controls the match function and the external match pins. | 0 | 16.6.10 |
| 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 | 16.6.11 |
| PWMC | R/W | 0x74 | PWM Control Register. The PWMCON enables PWM mode for the external match pins. | 0 | 16.6.12 |
| MSR0 | R/W | 0x78 | Match 0 Shadow Register. If enabled, the Match 0 Register will be automatically reloaded with the contents of this register whenever the TC is reset to zero. | 0 | 16.6.13 |

**Table 223. Register overview: CTIMER (register base addresses 0x4003 8000)**

| Name | Access | Offset | Description | Reset value[1] | Section |
|---|---|---|---|---|---|
| MSR1 | R/W | 0x7C | Match 1 Shadow Register. If enabled, the Match 1 Register will be automatically reloaded with the contents of this register whenever the TC is reset to zero. | 0 | 16.6.13 |
| MSR2 | R/W | 0x80 | Match 2 Shadow Register. If enabled, the Match 2 Register will be automatically reloaded with the contents of this register whenever the TC is reset to zero. | 0 | 16.6.13 |
| MSR3 | R/W | 0x84 | Match 3 Shadow Register. If enabled, the Match 3 Register will be automatically reloaded with the contents of this register whenever the TC is reset to zero. | 0 | 16.6.13 |

[1] Reset Value reflects the data stored in used bits only. It does not include reserved bits content.

### 16.6.1 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.

**Table 224. Interrupt Register (IR, 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 |
| 31:7 | - | Reserved. Read value is undefined, only zero should be written. | - |

### 16.6.2 Timer Control Register

The Timer Control Register (TCR) is used to control the operation of the Timer/Counter.

**Table 225. Timer Control Register (TCR, 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. | |
| 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 the APB bus clock. The counters remain reset until TCR[1] is returned to zero. | |
| 31:2 | - | - | Reserved. Read value is undefined, only zero should be written. | NA |

### 16.6.3 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 226. Timer counter registers (TC, offset 0x08) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 31:0 | TCVAL | Timer counter value. | 0 |

### 16.6.4 Prescale register

The 32-bit Prescale register specifies the maximum value for the Prescale Counter.

**Table 227. Timer prescale registers (PR, offset 0x00C) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 31:0 | PRVAL | Prescale counter value. | 0 |

### 16.6.5 Prescale Counter register

The 32-bit Prescale Counter controls division of the APB bus clock 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 APB bus clock. When it reaches the value stored in the Prescale register, the Timer Counter is incremented and the Prescale Counter is reset on the next APB bus clock. This causes the Timer Counter to increment on every APB bus clock when PR = 0, every 2 APB bus clocks when PR = 1, etc.

**Table 228. Timer prescale counter registers (PC, offset 0x010) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 31:0 | PCVAL | Prescale counter value. | 0 |

### 16.6.6 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 229. Match Control Register (MCR, 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.<br>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.<br>0 = disabled. 1 = enabled. | 0 |
| 3 | MR1I | Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC.<br>0 = disabled. 1 = enabled. 0 = disabled. 1 = enabled. | 0 |
| 4 | MR1R | Reset on MR1: the TC will be reset if MR1 matches it.<br>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.<br>0 = disabled. 1 = enabled. | 0 |
| 6 | MR2I | Interrupt on MR2: an interrupt is generated when MR2 matches the value in the TC.<br>0 = disabled. 1 = enabled. | 0 |
| 7 | MR2R | Reset on MR2: the TC will be reset if MR2 matches it.<br>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.<br>0 = disabled. 1 = enabled. | 0 |
| 9 | MR3I | Interrupt on MR3: an interrupt is generated when MR3 matches the value in the TC.<br>0 = disabled. 1 = enabled. | 0 |
| 10 | MR3R | Reset on MR3: the TC will be reset if MR3 matches it.<br>0 = disabled. 1 = enabled. | 0 |

**Table 229. Match Control Register (MCR, offset 0x014) bit description**

| Bit | Symbol | Description | Reset Value |
|---|---|---|---|
| 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 |
| 23:12 | - | Reserved. Read value is undefined, only zero should be written. | NA |
| 24 | MR0RL | Reload MR0 with the contents of the Match 0 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled. | 0 |
| 25 | MR1RL | Reload MR1 with the contents of the Match 1 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled. | 0 |
| 26 | MR2RL | Reload MR2 with the contents of the Match 2 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled. | 0 |
| 27 | MR3RL | Reload MR3 with the contents of the Match 3 Shadow Register when the TC is reset to zero (either via a match event or a write to bit 1 of the TCR). 0 = disabled. 1 = enabled. | 0 |
| 31:28 | - | Reserved. Read value is undefined, only zero should be written. | NA |

### 16.6.7 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.

If the associated MRxRL bit in the Match Control Register is set, the Match Register will be automatically reloaded with the current contents of its corresponding Match Shadow register whenever the TC is cleared to zero. This transfer will take place on the same clock edge that clocks the TC to zero.

Note: The TC is typically reset in response to an occurrence of a match on the Match Register being used to set the cycle counter rate. A reset can also occur due to software writing a 1 to bit 1 of the Timer Control Register.

**Table 230. Timer match registers (MR[0:3], offset [0x018:0x024]) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 31:0 | MATCH | Timer counter match value. | 0 |

### 16.6.8 Capture Control Register

The Capture Control Register is used to control whether one of the three 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.

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 231. Capture Control Register (CCR, 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 |
| 31:9 | - | Reserved. Read value is undefined, only zero should be written. | NA |

### 16.6.9 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 232. Timer capture registers (CR[0:3], offsets [0x02C:0x038]) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 31:0 | CAP | Timer counter capture value. | 0 |

### 16.6.10 External Match Register

The External Match Register provides both control and status of the external match pins.

If the match outputs are configured as PWM output, the function of the external match registers is determined by the PWM rules (Section 16.7.1 “Rules for single edge controlled PWM outputs” on page 239).

**Table 233. Timer external match registers (EMR, 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 SWM. | 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 SWM. | 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 SWM. | 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 SWM. | 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 |

### 16.6.11 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 APB bus 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 APB bus 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 APB bus clock. Consequently, duration of the HIGH/LOWLOW levels on the same CAP input in this case cannot be shorter than 1/APB bus clock.

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 234. Count Control Register (CTCR, offset 0x070) bit description**

| Bit | Symbol | Value | Description | Reset Value |
|---|---|---|---|---|
| 1:0 | CTMODE | | Counter/Timer Mode<br>This field selects which rising APB bus clock edges can increment Timer's Prescale Counter (PC), or clear PC and increment Timer Counter (TC).<br>Timer Mode: the TC is incremented when the Prescale Counter matches the Prescale Register. | 00 |
| | | 0x0 | Timer Mode. Incremented every rising APB bus clock 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. | |
| 3:2 | CINSEL | | Count Input Select<br>When bits 1:0 in this register are not 00, these bits select which CAP pin is sampled for clocking.<br>**Note:** If Counter mode is selected 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 2 CAP inputs. | 0 |
| | | 0x0 | Channel 0. CAP0 | |
| | | 0x1 | Channel 1. CAP1 | |
| | | 0x2 | Channel 2. CAP2 | |
| | | 0x3 | Reserved | |
| 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 |

**Table 234. Count Control Register (CTCR, offset 0x070) bit description**

| Bit | Symbol | Value | Description | Reset Value |
|---|---|---|---|---|
| 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 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). | |
| | | 0x6 | Reserved | |
| | | 0x7 | Reserved | |
| 31:8 | - | - | Reserved. Read value is undefined, only zero should be written. | NA |

### 16.6.12 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).

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 235. PWM Control Register (PWMC, offset 0x074)) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | PWMEN0 | | PWM mode enable for channel0. | 0 |
| | | 0 | Match. CTIMER_MAT0 is controlled by EM0. | |
| | | 1 | PWM. PWM mode is enabled for CTIMER_MAT0. | |
| 1 | PWMEN1 | | PWM mode enable for channel1. | 0 |
| | | 0 | Match. CTIMER_MAT01 is controlled by EM1. | |
| | | 1 | PWM. PWM mode is enabled for CTIMER_MAT1. | |
| 2 | PWMEN2 | | PWM mode enable for channel2. | 0 |
| | | 0 | Match. CTIMER_MAT2 is controlled by EM2. | |
| | | 1 | PWM. PWM mode is enabled for CTIMER_MAT2. | |

**Table 235. PWM Control Register (PWMC, offset 0x074)) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 3 | PWMEN3 | | PWM mode enable for channel3. **Note:** It is recommended to use match channel 3 to set the PWM cycle. | 0 |
| | | 0 | Match. CTIMER_MAT3 is controlled by EM3. | |
| | | 1 | PWM. PWM mode is enabled for CTIMER_MAT3. | |
| 31:4 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

### 16.6.13 Match Shadow Registers

The Match Shadow registers contain the values that the corresponding Match Registers are (optionally) reloaded with at the start of each new counter cycle. Typically, the match that causes the counter to be reset (and instigates the match reload) will also be programmed to generate an interrupt. Software will then have one full counter cycle to modify the contents of the Match Shadow Register(s) before the next reload occurs.

**Table 236. Timer match shadow registers (MSR[0:3], offset [0x78:0x84]) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 31:0 | SHADOW | Timer counter match shadow value. | 0x0 |

## 16.7 Functional description

Figure 33 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 34 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.

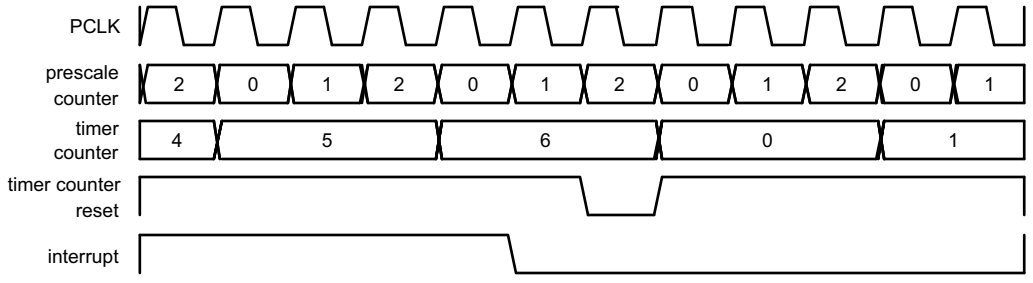



**Fig 33. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled**

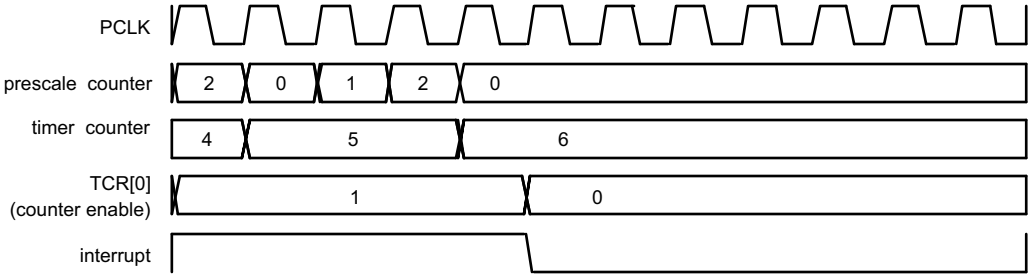



**Fig 34. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled**

### 16.7.1 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.

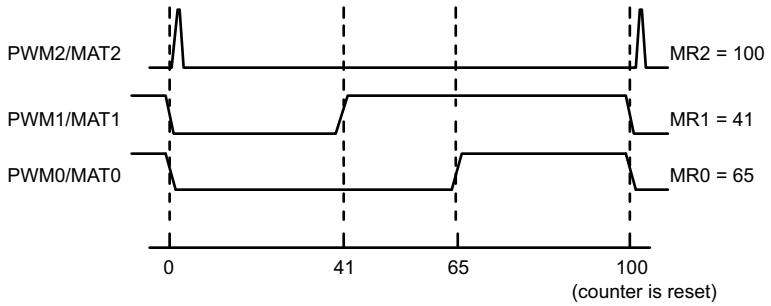



**Fig 35. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.**

## 17.1 How to read this chapter

The watchdog timer is identical on all LPC804 parts.

## 17.2 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 or watchdog feed sequence can optionally be protected so that it can only be performed after the “warning interrupt” time is reached.
- Flag to indicate Watchdog reset.
- The Watchdog clock (WDCLK) source is the WatchDog oscillator.
- The Watchdog timer can be configured to run in deep-sleep or power-down mode.
- Debug mode.

## 17.3 Basic configuration

The WWDT is configured through the following registers:

- Power to the register interface (WWDT PCLK clock): In the SYSAHBCLKCTRL register, set bit 17 in Table 64.
- Enable the WWDT clock source (the watchdog oscillator) in the PDRUNCFG register (Table 84). This is the clock source for the timer base.
- For waking up from a WWDT interrupt, enable the watchdog interrupt for wake-up in the STARTERP1 register (Table 81).

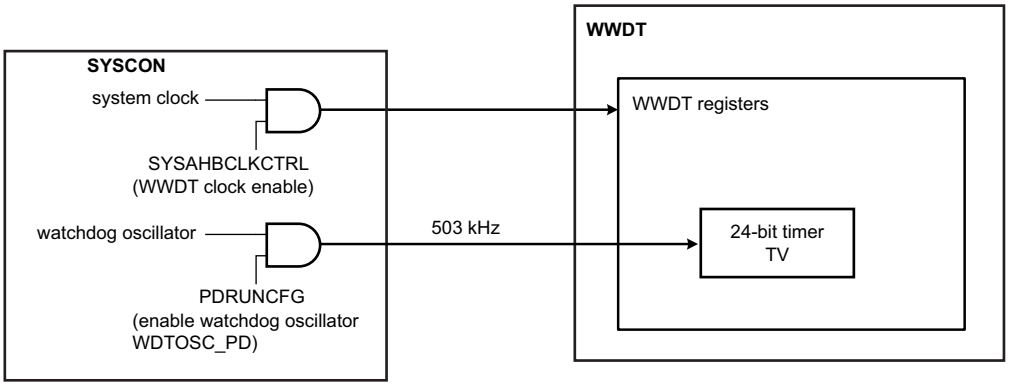



**Fig 36. WWDT clocking**

## 17.4 Pin description

The WWDT has no external pins.

## 17.5 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.
- Set the Watchdog timer update mode (WDPROTECT) in the MOD register after a delay of three WDCLK clock cycles.

- 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.

### 17.5.1 Block diagram

The block diagram of the Watchdog is shown below in the Figure 37. The synchronization logic (PCLK - WDCLK) is not shown in the block diagram.

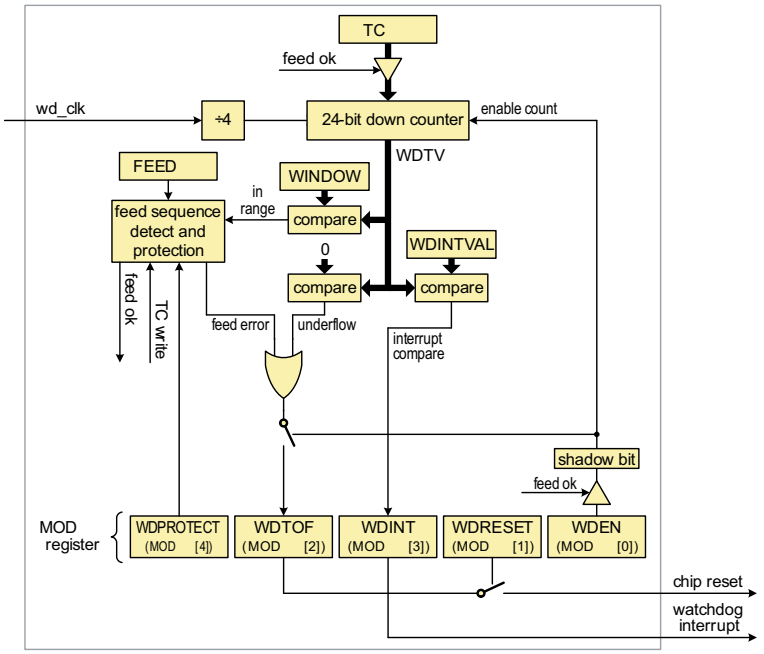



**Fig 37. Windowed Watchdog timer block diagram**

### 17.5.2 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 6). 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 WDTV 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.

### 17.5.3 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.
- Performing the WWDT reload or WWDT feed sequence.

#### 17.5.3.1 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.

In Deep power-down mode, no clock locking mechanism is in effect because no clocks are running. However, an additional lock bit in the PMU can be set to prevent the part from even entering Deep power-down mode (see Table 169).

#### 17.5.3.2 Changing the WWDT reload value

If bit 4 is set in the WWDT MOD register, the watchdog reload or watchdog feed sequence can be performed only after the watchdog timer is below the value of WDWARNING and WDWINDOW.

The reload overwrite lock mechanism can only be disabled by a reset of any type.

## 17.6 Register description

The Watchdog Timer contains the registers shown in Table 237.

The reset value reflects the data stored in used bits only. It does not include the content of reserved bits.

**Table 237. Register overview: Watchdog timer (base address 0x4000 0000)**

| 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 238 |
| TC | R/W | 0x004 | Watchdog timer constant register. This 24-bit register determines the time-out value. | 0xFF | Table 240 |
| FEED | WO | 0x008 | Watchdog feed sequence register. Writing 0xAA followed by 0x55 to this register reloads the Watchdog timer with the value contained in WDTC. | NA | Table 241 |
| TV | RO | 0x00C | Watchdog timer value register. This 24-bit register reads out the current value of the Watchdog timer. | 0xFF | Table 242 |
| - | - | 0x010 | Reserved | - | - |
| WARNINT | R/W | 0x014 | Watchdog Warning Interrupt compare value. | 0 | Table 243 |
| WINDOW | R/W | 0x018 | Watchdog Window compare value. | 0xFF FFFF | Table 244 |

### 17.6.1 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 238. Watchdog mode register (MOD, 0x4000 0000) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | WDEN | | Watchdog enable bit. Once this bit has been written with a 1, it cannot be re-written with a 0. Once this bit is set to one, the watchdog timer starts running after a watchdog feed. | 0 |
| | | 0 | The watchdog timer is stopped. | |
| | | 1 | 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 | A watchdog time-out will not cause a chip reset. | |
| | | 1 | 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 writing 0. Causes a chip reset if WDRESET = 1. | 0 (only after external reset) |

**Table 238. Watchdog mode register (MOD, 0x4000 0000) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 3 | WDINT | | Warning interrupt flag. Set when the timer reaches the value in WDWARNINT. Cleared by writing 1. | 0 |
| 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 reload or watchdog feed sequence can be performed when the watchdog timer is below the value of WDWINDOW. | |
| | | 1 | Threshold. The watchdog reload or watchdog feed sequence can be performed only after the watchdog timer is below the value of WDWARNING and WDWINDOW. | |
| 5 | LOCK | | A 1 in this bit prevents disabling or powering down the watchdog oscillator. This bit can be set once by software and is only cleared by any reset. | 0 |
| 31:6 | - | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | 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 STARTERP1 register in addition to the NVIC.

See the following registers:

Table 81 “Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description”

**Table 239. 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.<br>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.<br>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. |

## 17.6.2 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 perform the watchdog reload or watchdog feed sequence before the watchdog timer is below the values of WDWARNINT and WDWINDOW will cause a watchdog feed error and set the WDTOF flag.

**Table 240. Watchdog Timer Constant register (TC, 0x4000 0004) bit description**

| Bit | Symbol | Description | Reset Value |
|---|---|---|---|
| 23:0 | COUNT | Watchdog time-out value. | 0x00 00FF |
| 31:24 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

## 17.6.3 Watchdog Feed register

Writing 0xAA followed by 0x55 to this register will reload the Watchdog timer with the WDTC 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 241. Watchdog Feed register (FEED, 0x4000 0008) bit description

| Bit | Symbol | Description | Reset Value |
|---|---|---|---|
| 7:0 | FEED | Feed value should be 0xAA followed by 0x55. | NA |
| 31:8 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

### 17.6.4 Watchdog Timer Value register

The WDTV 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 WDTV is older than the actual value of the timer when it's being read by the CPU.

Table 242. Watchdog Timer Value register (TV, 0x4000 000C) bit description

| Bit | Symbol | Description | Reset Value |
|---|---|---|---|
| 23:0 | COUNT | Counter timer value. | 0x00 00FF |
| 31:24 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

### 17.6.5 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 is less than or equal to 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 is less than or equal to 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 243. Watchdog Timer Warning Interrupt register (WARNINT, 0x4000 0014) bit description

| Bit | Symbol | Description | Reset Value |
|---|---|---|---|
| 9:0 | WARNINT | Watchdog warning interrupt compare value. | 0 |
| 31:10 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

### 17.6.6 Watchdog Timer Window register

The WINDOW register determines the highest WDTV value allowed when a watchdog feed is performed. If a feed sequence occurs when WDTV is greater than the value in WINDOW, a watchdog event will occur.

WINDOW resets to the maximum possible WDTV value, so windowing is not in effect.

**Table 244. Watchdog Timer Window register (WINDOW, 0x4000 0018) bit description**

| Bit | Symbol | Description | Reset Value |
|---|---|---|---|
| 23:0 | WINDOW | Watchdog window value. | 0xFF FFFF |
| 31:24 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

## 17.7 Functional description

The following figures illustrate several aspects of Watchdog Timer operation.

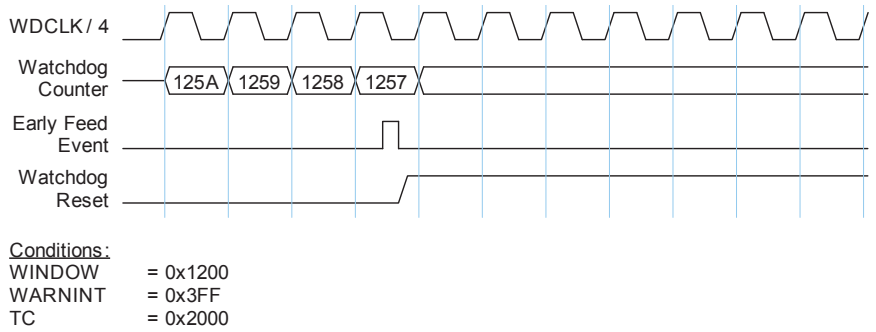



**Fig 38. Early watchdog feed with windowed mode enabled**

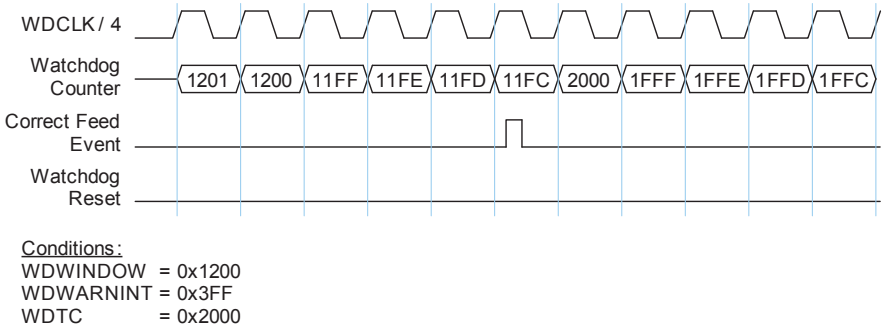



**Fig 39. Correct watchdog feed with windowed mode enabled**

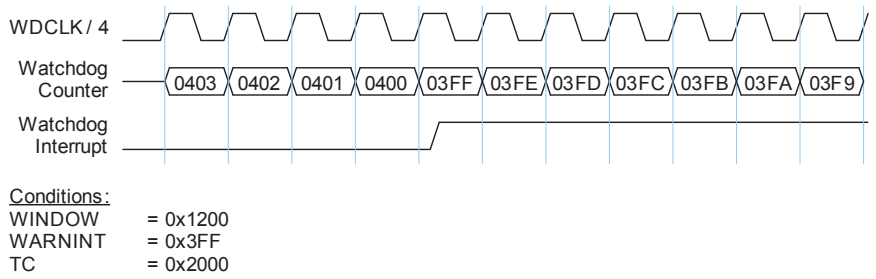



**Fig 40. Watchdog warning interrupt**

## 18.1 How to read this chapter

The self-wake-up timer is available on all LPC804 parts.

## 18.2 Features

- 32-bit, loadable down counter. Counter starts automatically when a count value is loaded. Time-out generates an interrupt/wake up request.
- The WKT supports three clock sources: The FRO, the internal low-power oscillator, or the WKTCLKIN pin. The low-power oscillator and the external clock can be configured to be valid clock sources in all power modes except deep power-down. The FRO can be used in sleep and active mode only.
- Depending on the clock source, the WKT can be used for waking up the part from different low power modes or for general-purpose timing.

## 18.3 Basic configuration

- In the SYSAHBCLKCTRL register, set bit 9 (Table 64) to enable the clock to the register interface.
- Clear the WKT reset using the PRESETCTRL register (Table 66).
- The WKT interrupt is connected to interrupt #15 in the NVIC. See Table 38.
- Enable the low power oscillator in the PDRUNCFG register if used as the clock source for the timer.
- Enable the FRO and FRO output in the PDRUNCFG register if used as the clock source for the timer (Table 84).
- To use an external clock source for the self-wake-up timer, enable the clock input for pin PIO0_11 by setting the switch matrix PINENABLE0 register and enable the external clock option in the self-wake-up timer CTRL register (see Table 246).

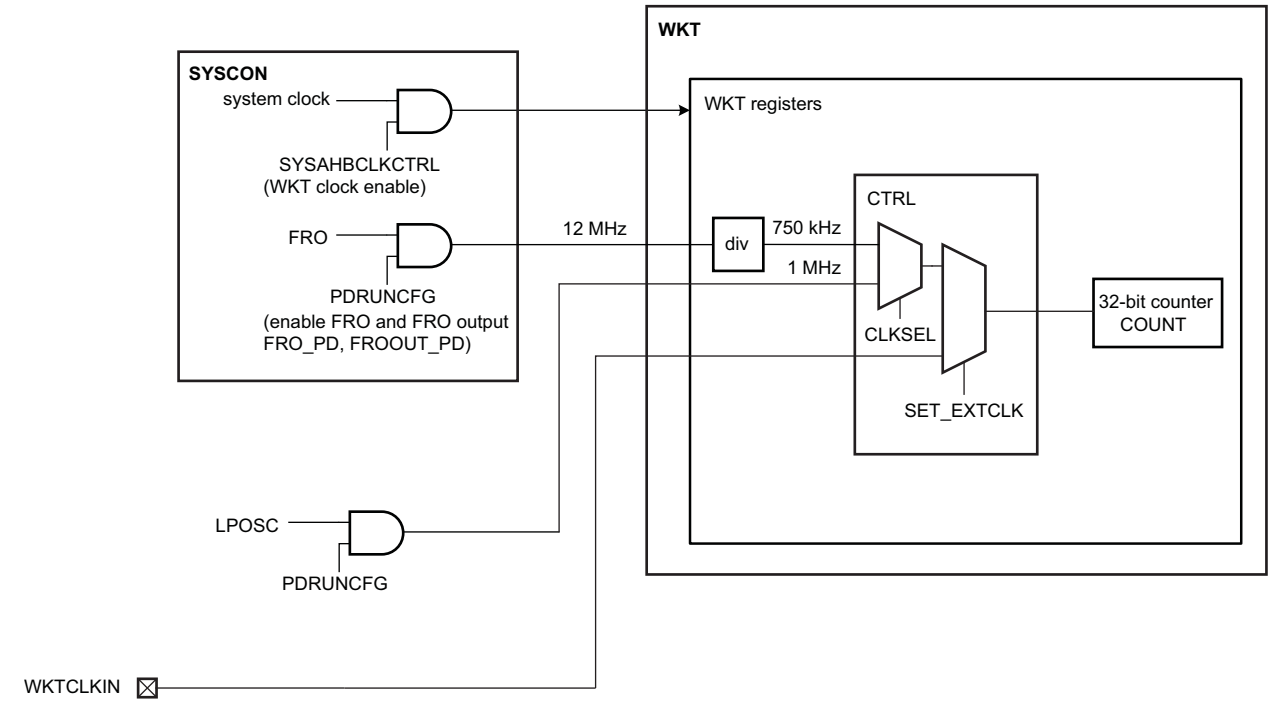



**Fig 41. WKT clocking**

## 18.4 Pin description

The WKT can use a clock input on the external pin PIO0_11 for clocking the wake-up timer. Select the external clock source by setting bit SET_EXTCLK in the CTRL register (see Table 246).

## 18.5 General description

The self-wake-up timer is a 32-bit, loadable down counter. Writing any non-zero value to this timer automatically enables the counter and launches a count-down 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 self-wake-up timer automatically turns on, counts from the pre-loaded value down to zero, generates an interrupt and/or a wake up request, and then turns itself off until re-launched by a subsequent software write.

### 18.5.1 WKT clock sources

The self-wake-up timer can be clocked from two alternative internal clock sources and one external clock:

- A 750 kHz clock derived from the FRO oscillator. This is the default clock.
- A 1 MHz, low-power clock with a dedicated on-chip oscillator as clock source.
- An external clock on the WKTCLKIN pin.

The FRO-derived clock is much more accurate than the alternative, low-power clock. However, the FRO is not available in most low-power modes. This clock must not be selected when the timer is being used to wake up from a power mode where the FRO is disabled.

The alternative clock source is a (nominally) 1 MHz, low-power clock, sourced from a dedicated oscillator. When the FRO clock is shut-down, the 1 MHz clock can be configured to be available during low-power modes (deep sleep and power down mode).

An external clock on the WKTCLKIN pin can be used to time the self-wake-up timer in low power modes, except deep power-down.

## 18.6 Register description

**Table 245. Register overview: WKT (base address 0x4000 8000)**

| Name | Access | Address offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| CTRL | R/W | 0x0 | Self-wake-up timer control register. | 0 | Table 246 |
| COUNT | R/W | 0xC | Counter register. | - | Table 247 |

### 18.6.1 Control register

The WKT interrupt must be enabled in the NVIC to wake up the part using the self-wake-up counter.

**Table 246. Control register (CTRL, address 0x4000 8000) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | CLKSEL | | Select the self-wake-up timer clock source.<br>**Remark:** This bit only has an effect if the SEL_EXTCLK bit is not set. | 0 |
| | | 0 | Divided FRO clock. This clock runs at 750 kHz and provides time-out periods of up to approximately 95 minutes in 1.33 µs increments.<br>**Remark:** This clock is not available in not available in deep-sleep, power-down, deep power-down modes. Do not select this option if the timer is to be used to wake up from one of these modes. | |
| | | 1 | Low power clock. This is the 1 MHz clock and provides time-out periods of up to approximately 71.6 minutes in 1 µs increments.<br>**Remark:** This clock is not available in deep power-down mode. Prior to use, enable the low-power oscillator. | |
| 1 | ALARMFLAG | | Wake-up or alarm timer flag. | - |
| | | 0 | No time-out. The self-wake-up timer has not timed out. Writing a 0 to has no effect. | |
| | | 1 | Time-out. The self-wake-up timer has timed out. This flag generates an interrupt request which can wake up the part from any reduced power mode. Writing a 1 clears this status bit. | |

**Table 246. Control register (CTRL, address 0x4000 8000) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2 | CLEARCTR | | Clears the self-wake-up timer. | 0 |
| | | 0 | No effect. Reading this bit always returns 0. | |
| | | 1 | Clear the counter. Counting is halted until a new count value is loaded. | |
| 3 | SEL_EXTCLK | | Select external or internal clock source for the self-wake-up timer. The internal clock source is selected by the CLKSEL bit in this register if SET_EXTCLK is set to internal. | 0 |
| | | 0 | Internal. The clock source is the internal clock selected by the CLKSEL bit. | |
| | | 1 | External. The self-wake-up timer uses the external WKTCLKIN pin. | |
| 31:4 | - | | Reserved. | - |

### 18.6.2 Count register

Do not write to this register while the counting is in progress.

**Remark:** In general, reading the timer state is not recommended. There is no mechanism to ensure that some bits of this register don't change while a read is in progress if the read happens to coincide with an self-wake-up timer clock edge. If you must read this value, it is recommended to read it twice in succession.

**Table 247. Counter register (COUNT, address 0x4000 800C) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 31:0 | VALUE | A write to this register pre-loads start count value into the timer and starts the count-down sequence.<br>A read reflects the current value of the timer. | - |

## 19.1 How to read this chapter

The MRT is available on all LPC804 parts.

## 19.2 Features

- 31-bit interrupt timer
- Two channels independently counting down from individually set values
- Repeat, bus-stall, and one-shot interrupt modes

## 19.3 Basic configuration

Configure the MRT using the following registers:

- In the SYSAHBCLKCTRL register, set bit 10 (Table 64) to enable the clock to the register interface.
- Clear the MRT reset using the PRESETCTRL register (Table 66).
- The global MRT interrupt is connected to interrupt #10 in the NVIC.

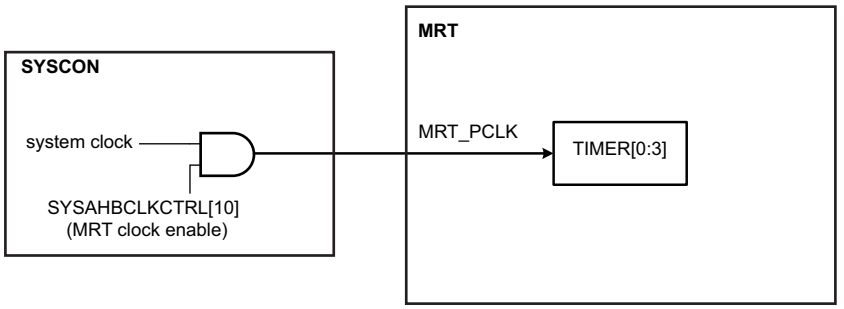



Fig 42. MRT clocking

## 19.4 Pin description

The MRT has no configurable pins.

## 19.5 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 19.5.1.

- One-shot interrupt mode. See Section 19.5.2.
- Bus-stall mode.

The modes for each timer are set in the timer's control register. See Table 251.

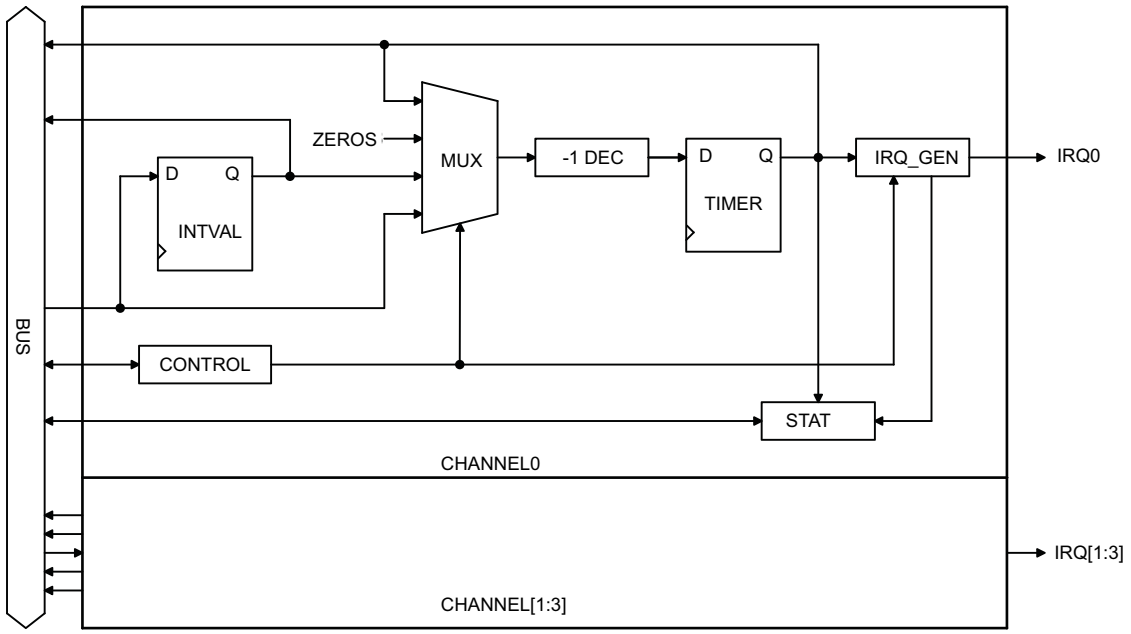



**Fig 43. MRT block diagram**

### 19.5.1 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, you can perform the following actions:

- 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.

### 19.5.2 One-shot interrupt mode

The one-shot interrupt generates one interrupt after a one-time count. With this mode, you can generate a single interrupt 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, you can perform the following actions:

- 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.

### 19.5.3 One-shot bus stall mode

The one-shot bus stall mode stalls the bus interface for IVALUE +3 cycles of the system clock. For the Cortex-M0+, this mode effectively stops all CPU activity until the MRT has finished counting down to zero. At the end of the count-down, no interrupt is generated, instead the bus resumes its transactions. The bus stall mode allows to halt an application for a predefined amount of time and then resume, as opposed to creating a software loop or polling a timer. Since in bus-stall mode, there are no bus transactions while the MRT is counting down, the CPU consumes a minimum amount of power during that time. Typically, this mode can be used when an application must be idle for a short time (in the order of µs or 10 to 50 clock cycles) - for example when compensating for a settling time and thus no CPU activity is required.

For longer wait times, use the one-shot interrupt mode, which allows other enabled interrupts to be serviced.

**Remark:** Because the MRT resides on the APB, the total amount of wait cycles inserted in bus stall mode, 3 cycles have to be added to IVALUE to account for the AHB-to-APB bridge.

## 19.6 Register description

The reset values shown in Table 248 are POR reset values.

**Table 248. Register overview: MRT (base address 0x4000 4000)**

| Name | Access | Address offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| INTVAL0 | R/W | 0x0 | MRT0 Time interval value register. This value is loaded into the TIMER0 register. | 0 | Table 249 |
| TIMER0 | RO | 0x4 | MRT0 Timer register. This register reads the value of the down counter. | 0x7FFF FFFF | Table 250 |
| CTRL0 | R/W | 0x8 | MRT0 Control register. This register controls the MRT0 modes. | 0 | Table 251 |
| STAT0 | R/W | 0xC | MRT0 Status register. | 0 | Table 252 |
| INTVAL1 | R/W | 0x10 | MRT1 Time interval value register. This value is loaded into the TIMER1 register. | 0 | Table 249 |
| TIMER1 | RO | 0x14 | MRT1 Timer register. This register reads the value of the down counter. | 0x7FFF FFFF | Table 250 |
| CTRL1 | R/W | 0x18 | MRT1 Control register. This register controls the MRT1 modes. | 0 | Table 251 |
| STAT1 | R/W | 0x1C | MRT1 Status register. | 0 | Table 252 |
| INTVAL2 | R/W | 0x20 | MRT2 Time interval value register. This value is loaded into the TIMER2 register. | 0 | |
| TIMER2 | RO | 0x24 | MRT2 Timer register. This register reads the value of the down counter. | 0x7FFF FFFF | |
| CTRL2 | R/W | 0x28 | MRT2 Control register. This register controls the MRT2 modes. | 0 | |
| STAT2 | R/W | 0x2C | MRT2 Status register. | 0 | |
| INTVAL3 | R/W | 0x30 | MRT3 Time interval value register. This value is loaded into the TIMER3 register. | 0 | |
| TIMER3 | RO | 0x34 | MRT3 Timer register. This register reads the value of the down counter. | 0 | |
| CTRL3 | R/W | 0x38 | MRT3 Control register. This register controls the MRT modes. | 0 | |
| STAT3 | R/W | 0x3C | MRT3 Status register. | 0 | |
| MODCFG | R/W | 0xF0 | Module Configuration. This register provides information about this particular MRT instance, and allows choosing an overall mode for the idle channel feature. | 0 | Table 253 |
| IDLE_CH | R | 0xF4 | Idle channel register. This register returns the number of the first idle channel. | 0 | Table 254 |
| IRQ_FLAG | R/W | 0xF8 | Global interrupt flag register. | 0 | Table 255 |

### 19.6.1 Time interval register

This register contains the MRT load value and controls how the timer is reloaded. The load value is IVALUE -1.

**Table 249. Time interval register (INTVAL[0:3], address 0x4000 4000 (INTVAL0) to 0x4000 4030 (INTVAL3)) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 30:0 | IVALUE | | Time interval load value. This value is loaded into the TIMERn register and the MRTn starts counting down from IVALUE -1.<br>If the timer is idle, writing a non-zero value to this bit field starts the timer immediately.<br>If the timer is running, writing a zero to this bit field does the following:<br>• If LOAD = 1, the timer stops immediately.<br>• If LOAD = 0, the timer stops at the end of the time interval. | 0 |
| 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. | |

### 19.6.2 Timer register

The timer register holds the current timer value. This register is read-only.

**Table 250. Timer register (TIMER[0:3], address 0x4000 4004 (TIMER0) to 0x4000 4034 (TIMER3)) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 30:0 | 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:<br>INTVALn register is updated in the idle state.<br>INTVALn register is updated with LOAD = 1.<br>When the timer is in idle state, reading this bit fields returns -1 (0x00FF FFFF). | 0x00FF FFFF |
| 31 | - | Reserved. | 0 |

### 19.6.3 Control register

The control register configures the mode for each MRT and enables the interrupt.

**Table 251. Control register (CTRL[0:3], address 0x4000 4008 (CTRL0) to 0x4000 4038 (CTRL3)) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | INTEN | | Enable the TIMERn interrupt. | 0 |
| | | 0 | Disable. | |
| | | 1 | Enable. | |
| 2:1 | MODE | | Selects timer mode. | 0 |
| | | 0x0 | Repeat interrupt mode. | |
| | | 0x1 | One-shot interrupt mode. | |
| | | 0x2 | One-shot bus stall mode. | |
| | | 0x3 | Reserved. | |
| 31:3 | - | | Reserved. | 0 |

### 19.6.4 Status register

This register indicates the status of each MRT.

**Table 252. Status register (STAT[0:3], address 0x4000 400C (STAT0) to 0x4000 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.<br>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. | |
| 31:2 | - | | Reserved. | 0 |

### 19.6.5 Module Configuration register

The MODCFG register provides the configuration (number of channels and timer width) for this MRT. See Section 19.6.6 “Idle channel register” for details.

**Table 253. Module Configuration register (MODCFG, offset 0xF0) bit description**

| Bit | Symbol | Value | Description | Reset Value |
|---|---|---|---|---|
| 3:0 | NOC | - | Identifies the number of channels in this MRT. (4 channels on this device.) | 0x3 |
| 8:4 | NOB | - | Identifies the number of timer bits in this MRT. (24 bits wide on this device.) | 0x17 |
| 30:9 | - | - | Reserved. Read value is undefined, only zero should be written. | - |
| 31 | MULTITASK | | Selects the operating mode for the INUSE flags and the IDLE_CH register. | 0x0 |
| | | 0 | Hardware status mode. In this mode, the INUSE(n) flags for all channels are reset. | |
| | | 1 | Multi-task mode. | |

### 19.6.6 Idle channel register

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 254. Idle channel register (IDLE_CH, address 0x4000 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. If all timer channels are running, CHAN = 4. | 0 |
| 31:8 | - | Reserved. | 0 |

### 19.6.7 Global interrupt flag register

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 255. Global interrupt flag register (IRQ_FLAG, address 0x4000 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.<br>Writing a 1 to this bit clears the interrupt request. | |
| 1 | GFLAG1 | | Monitors the interrupt flag of TIMER1. | 0 |
| | | 0 | No pending interrupt. Writing a zero is equivalent to no operation. | |
| | | 1 | Pending interrupt. The interrupt is pending because TIMER1 has reached the end of the time interval. If the INTEN bit in the CONTROL1 register is also set to 1, the interrupt for timer channel 1 and the global interrupt are raised.<br>Writing a 1 to this bit clears the interrupt request. | |
| 31:2 | - | | Reserved. | 0 |

## 20.1 How to read this chapter

The SysTick timer is available on all LPC804 parts.

## 20.2 Features

- Simple 24-bit timer.
- Uses dedicated exception vector.
- Clocked internally by the system clock or the system clock/2.

## 20.3 Basic configuration

The system tick timer is configured using the following registers:

1. The system tick timer is enabled through the SysTick control register (Table 257). The system tick timer clock is fixed to half of the system clock frequency.
2. Enable the clock source for the SysTick timer in the SYST_CSR register (Table 257).
3. The calibration value of the SysTick timer is contained in the SYSTCKCAL register in the system configuration block SYSCON (see Table 76).

## 20.4 Pin description

The SysTick has no configurable pins.

## 20.5 General description

The block diagram of the SysTick timer is shown in Figure 44.

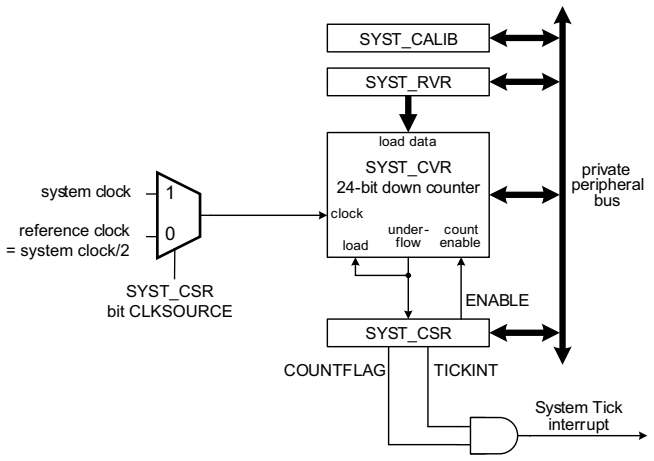



**Fig 44. System tick timer block diagram**

The SysTick timer is an integral part of the Cortex-M0+. 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 Cortex-M0+, it facilitates porting of software by providing a standard timer that is available on Cortex-M0 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 Ref. 6 for details.

## 20.6 Register description

The SysTick timer registers are located on the Arm Cortex-M0+ private peripheral bus (see Figure 2), and are part of the Arm Cortex-M0+ core peripherals. For details, see Ref. 6.

**Table 256. Register overview: SysTick timer (base address 0xE000 E000)**

| Name | Access | Address offset | Description | Reset value[1] |
|---|---|---|---|---|
| SYST_CSR | R/W | 0x010 | System Timer Control and status register | 0x000 0000 |
| SYST_RVR | R/W | 0x014 | System Timer Reload value register | 0 |
| SYST_CVR | R/W | 0x018 | System Timer Current value register | 0 |
| SYST_CALIB | R/W | 0x01C | System Timer Calibration value register | 0x4 |

[1] Reset Value reflects the data stored in used bits only. It does not include content of reserved bits.

### 20.6.1 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 Arm Cortex-M0+ core system timer register block. For a bit description of this register, see Ref. 6.

This register determines the clock source for the system tick timer.

**Table 257. 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 (CPU) clock is selected. When 0, the system clock/2 is selected as the reference clock. | 0 |
| 15:3 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 16 | COUNTFLAG | Returns 1 if the SysTick timer counted to 0 since the last read of this register. | 0 |
| 31:17 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

## 20.6.2 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 258. 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, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

## 20.6.3 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 259. 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, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

### 20.6.4 System Timer Calibration value register

The value of the SYST_CALIB register is driven by the value of the SYSTCKCAL register in the system configuration block SYSCON (see Table 76).

**Table 260. System Timer Calibration value register (SYST_CALIB, 0xE000 E01C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 23:0 | TENMS | | See Ref. 6. | 0x4 |
| 29:24 | - | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 30 | SKEW | | See Ref. 6. | 0 |
| 31 | NOREF | | See Ref. 6. | 0 |

## 20.7 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 6) 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. A default value is provided in the SYST_CALIB register and may be changed by software.

### 20.7.1 Example timer calculation

To use the system tick timer, do the following:

1. Program the SYST_RVR register with the reload value RELOAD to obtain the desired time interval.
2. Clear the SYST_CVR register by writing to it. This ensures that the timer will count from the SYST_RVR value rather than an arbitrary value when the timer is enabled.
3. Program the SYST_SCR register with the value 0x7 which enables the SysTick timer and the SysTick timer interrupt.

The following example illustrates selecting the SysTick timer reload value to obtain a 10 ms time interval with the system clock set to 20 MHz.

#### Example (system clock = 20 MHz)

The system tick clock = system clock = 20 MHz. Bit CLKSOURCE in the SYST_CSR register set to 1 (system clock).

RELOAD = (system tick clock frequency $\times$ 10 ms) $-1$ = (20 MHz $\times$ 10 ms) $-1$ = 200000 $-1$ = 199999 = 0x00030D3F.

## 21.1 How to read this chapter

The Capacitive Touch is available on all devices.

## 21.2 Features

The Capacitive Touch supports:

- Up to five mutual-capacitance touch sensors.
- Both GPIO port pin, and analog comparator measurement methods are available.
- Wake up from sleep, deep-sleep, and power-down modes.

## 21.3 Introduction

The Capacitive Touch module measures the change in capacitance of an electrode plate when an earth-ground connected object (for example, the finger or stylus) is brought within close proximity. Simply stated, the module delivers a small charge to an X capacitor (a mutual capacitance touch sensor), then transfers that charge to a larger Y capacitor (the measurement capacitor), and counts the number of iterations necessary for the voltage across the Y capacitor to cross a predetermined threshold.

The finger or stylus will impact the fringe capacitance field (between mutual electrodes of the X capacitor), effectively adding to, or subtracting from, the built up charge on the X capacitor. Once the threshold is crossed, the Y capacitor is discharged, and the process is repeated for the next X sensor. The number of iterations necessary to cross the threshold with an untouched sensor is used as the baseline count. For a given system and its grounding characteristics, a touched sensor triggers at either a lower or a higher count than an untouched sensor.

Once a calibrated and configured system operates normally, each X sensor generates either a ‘no-touch’ or a ‘touch’ event every time that sensor is polled. These events can be used to generate interrupts. The system requires re-calibration when environmental factors change, including temperature and humidity, which affect fringe capacitance fields.

## 21.4 Quick setup guide

- Enable the bus clock to the Capacitive Touch module by setting the appropriate bit in one of the SYSAHBCLKCTRL registers in SYSCON.
- Take the module out of reset by setting the appropriate bit in one of the PRESETCTRL registers in SYSCON.
- Provide a function clock (FCLK) to the module by writing to the CAPTCLKSEL register in SYSCON.
- Choose a divider value to produce 4 MHz from the given FCLK, and program the divider into the FDIV field of the module in the CTRL register.

- Enable the fixed-pin functions CAPT_YH, CAPT_YL, and any of CAPT_X0, CAPT_X1, etc. that will be used, by writing to the appropriate PINENABLE register(s) in the SWM.
- Disable pull-ups and pull-downs on those pads by writing to the appropriate registers in the IOCON.
- If the analog comparator measurement method is used, enable one of the ACMP_I fixed pin functions by writing to the appropriate PINENABLE register(s) in the SWM. If the device does not provide a shared CAPT_YH and ACMP_I pad, the package pins must be wire-OR'd externally.
- Identify to the module the set of X pins to use by writing to the XPINSEL field in the control register. The maximum number of X pins available for a given device can be read from the XMAX field in the status register and is equal to XMAX+1.
- In many systems a finger-touched sensor triggers at a lower count than an untouched sensor. If touch triggers lower than no-touch, set the TCHLOWER bit in the POLL_TCNT register.
- Set the threshold, and various other timing and counting parameters, by writing to the POLL_TCNT register (an initial POLL_NOW polling operation can be launched in order for the baseline no-touch trigger point to be determined).
- Enable interrupts in the CTRL register, as required, and start polling.

## 21.5 General description

### 21.5.1 Pin usage

The Capacitive Touch module uses one standard GPIO pin for YL and up to sixteen (depending on product family and part number) standard GPIOs for X0 through XMAX.

One standard GPIO is used for YH on products with nine or fewer X pins. For products with greater than nine X pins, there is a second YH GPIO pin.

YH, YL, and X functions are typically enabled on their pins using the switch matrix or IOCON, depending on the product family. Additionally, the set of X pins that the application will use must be enabled, or identified to the module by writing '1's to their bit positions in the XPINSEL field of the control register.

On devices with an analog comparator, if the analog comparator measurement method is used, an analog comparator input must be enabled via the switch matrix or IOCON, and also wire-OR'd externally to the YH pin. Some devices may provide a shared YH and ACMP_IN pin, in which case the external connection is not necessary.

Figure 45 shows the connections necessary for a basic button configuration using two sensors. The same connection scheme applies to slider and touch-wheel sensors. Additional sensors are added in parallel, each with its own X pin.

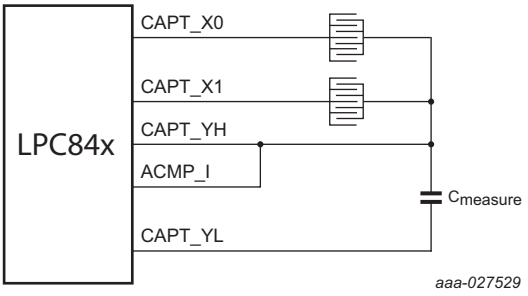



**Fig 45. Basic button configuration using two sensors.**

### 21.5.2 Basic measurement algorithm

The basic measurement algorithm, represents the states of the hardware module when any given sensor is polled.

Step state:

1. 0 Reset
2. 1 Charge
3. 2 Transfer
4. 3 Measure

In all states, X channels not identified to the module (that is, disabled in the XPINSEL field of the CTRL register) is kept in High-Z or is held low, depending on the value written to the XPINUSE field of the CTRL register.

Between measurements, the module remains in Step 0.

During a measurement, steps 1 to 3 are repeated until the ACMP (or YH port pin) reaches threshold or the module times out (too many iterations).

On completion, the module returns to Step 0 before any new measurements are taken.

### 21.5.3 Function clock divider

The function clock (FCLK) provided to the module is divided internally by the function clock divider, to produce the divided FCLK. All clocking and counting discussed fin this chapter is in reference to the divided FCLK, unless otherwise noted.

Divided FCLK can be any of FCLK-divided-by-1 through FCLK-divided-by-16. A good rule-of-thumb is to provide an FCLK frequency so that the divided FCLK can be 4 MHz or less.

## 21.6 Timing and counting parameters

### 21.6.1 Threshold count

The THRESHOLD COUNT (TCNT) sets the count boundary between touch and no-touch. The software configures the module to consider a touch to occur at either a higher or a lower count than the threshold count, based on the system specifics.

### 21.6.2 Time-out count

If the time-out count reaches before the YH port pin or the analog comparator crosses the zero-to-one voltage threshold, there is a time-out event and the X measurement is terminated. This should be set to a value much greater than the larger of the expected touch / no-touch counts. Under normal circumstances the time-out count should never be reached because each X sensor measurement triggers either a touch or a no-touch event before reaching time-out.

### 21.6.3 Poll counter

The poll counter sets the time delay between polling rounds (successive sets of X measurements). After each polling round completes, the module will wait 4096 x POLL divided FCLKs before starting the next polling round. Measuring too often can be inefficient and unnecessary because finger movement is relatively slow.

### 21.6.4 Measurement delay

The measurement delay specifies the number of divided FCLKs the module will wait after entering Measure Voltage state (Section 21.5.2, step 3), before sampling the YH port pin or analog comparator output. This gives the analog comparator time to react to the transferred charge.

### 21.6.5 Reset delay

The reset delay specifies the number of divided FCLKs the module will remain in Reset or Draining Cap. state (Section 21.5.2, step 0) before starting the next X measurement in the polling round. Larger capacitors may need more time to drain and/or the unselected Xs may need more time to go back to their resting state.

## 21.7 Modes of Operation

### 21.7.1 Measurement methods

The Capacitive Touch module can use two distinct methods of measuring the voltage across the measurement capacitor, based on the setting of the TRIGGER bit in the Control register.

#### 21.7.1.1 YH port pin measurement

In Measure Voltage state (Section 21.5.2, step 3), the module samples the YH port pin (which is in input mode). Until the voltage on the pad increases above the $V_{IH}$ (HIGH-level input voltage) of the pad, the module will sample ‘0’, above VIH it will sample ‘1’ (that is, the YH port pin has “triggered”).

#### 21.7.1.2 Analog comparator measurement

In Measure Voltage state (Section 21.5.2, step 3), the module samples the analog comparator output, which is connected internally to the module. The analog comparator must be enabled and properly configured, and one of the comparator analog inputs must be enabled and connected to the YH port pin. On some devices the YH port pin and an analog comparator input may share a pad or pin. Otherwise, the YH port pin and the analog comparator input pin must be connected externally (wire-OR'd).

Until the voltage on the analog comparator input increases above the configured threshold of the comparator, the module will sample '0', above that it will sample '1' (that is, the analog comparator has "triggered").

The voltage threshold of the comparator may be set lower than the VIH of the YH port pin, which allows for faster sensing than the YH port pin measurement method.

### 21.7.2 Polling modes

The Capacitive Touch module has the following polling modes, based on the POLLMODE field in the Control register.

#### 21.7.2.1 Inactive

No measurements are taken, no polls are performed. The module remains in the Reset / Draining Cap. state (Section 21.5.2, step 0). Counters are inactive. Entering this mode will immediately terminate any polling / measurement in progress. Entering this mode does not affect the five interrupt flags in the Status register.

#### 21.7.2.2 Poll Now

Immediately launches (ignoring Poll Delay) a one-time-only, simultaneous poll of all X pins that are enabled in the XPINSEL field of the Control register, then stops, returning to Reset / Draining Cap. state (Section 21.5.2, step 0). Note that all enabled X pins are activated concurrently, rather than walked one-at-a-time.

Useful for baselining threshold count and time-out count.

#### 21.7.2.3 Continuous

Polling rounds are continuously performed, by walking through the enabled X pins. Each polling round is preceded by the time delay specified by the poll counter in the POLL_TCNT register (which can be zero).

##### 21.7.2.3.1 Low power polling

The software may choose to take certain steps to conserve power, before putting the device into Deep-sleep / Power-down, with the intention of using a touch event to wake up the processor.

The software would do the following before going into a low-power state:

- Switch FCLK to a low-frequency, low-power clock source that stays on in deep-sleep or power-down mode, and adjust the Function clock divider, as well as the fields in the POLL_TCNT register, for this new frequency. Polling once every 250 ms or so may be sufficient for detecting a wake-up touch.
- Power-down the analog comparator, and switch to YH port pin measurement method.

- Choose the set of X pins in XPINSEL that will be polled for wake-up.
- Make sure that only the YESTOUCH interrupt is enabled.
- Enable the Capacitive Touch wake-up source in the SYSCON.
- Put the device into deep sleep or power down mode.
- A touch event will wake up the CPU and cause an interrupt. Software would then revert to the regular configuration.

### 21.7.3 Polling Types

#### 21.7.3.1 Normal

Normal polling (TYPE = 0) treats all enabled X elements individually, and they are scanned one-at-a-time in ascending order. This is used for buttons, basic sliders, etc.

## 21.8 Touch Data

The Touch information for the most recent measurement can be read from the Touch Data register. The information available are:

- COUNT: The count value reached at trigger or time-out. If the measurement resulted in a time-out, COUNT will contain the time-out count value minus one, that is, $2^{TOUT} - 1$.
- XVAL: For normal polling (TYPE = 0) in continuous mode (POLLMODE = 2), XVAL contains the index of the X pin for the current measurement. For a multiple-pin measurement in poll now mode (POLLMODE = 1), XVAL will contain the index of the lowest enabled X pin.
- ISTOUCH: Set if the trigger is due to a touch event, cleared if due to a no-touch event.
- ISTO: Set if the measurement resulted in a time-out event.
- SEQ: Sequence number, a 4-bit rolling counter that increments after all enabled X pins are scanned in a polling round.
- CHANGE: An indicator that will be set for one bus clock at the end of an X measurement, while the touch data are changing. Touch data read while this bit is high are invalid.

## 21.9 Interrupts

### 21.9.1 Interrupts

The Capacitive Touch module has five types of interrupts, each with a unique flag in the status register.

#### 21.9.1.1 Yes touch

The YESTOUCH flag is set when the YH port pin or analog comparator triggers at a count value that is greater than TCNT (if the TCHLOWER bit equals ‘0’), or a count value that is less-than-or-equal-to TCNT (if the TCHLOWER bit equals ‘1’).

#### 21.9.1.2 No touch

The NOTOUCH flag is set when the YH port pin or analog comparator triggers at a count value that is less-than-or-equal-to TCNT (if the TCHLOWER bit equals ‘0’), or a count value that is greater than TCNT (if the TCHLOWER bit equals ‘1’).

#### 21.9.1.3 Poll done

The POLLDONE flag is set at the end of each polling round, or when a Poll Now completes.

#### 21.9.1.4 Timeout

The TIMEOUT flag is set if the count reaches the time-out count value before the YH port pin or analog comparator crosses the zero-to-one voltage threshold.

#### 21.9.1.5 Overrun

The OVERRUN flag is set if the Touch Data register is updated before the software has read the previous data, and the trigger is due to a touch event (ISTOUCH = ‘1’). The OVERRUN flag will not be set on no-touch events (ISTOUCH = ‘0’, in which case touch data are silently overwritten) or on TIMEOUTs. If the WAIT bit in the Control register equals ‘1’, the OVERRUN flag will not be set.

## 21.10 Register description

**Table 261. Register overview: base address 0x4006 0000**

| Name | Access | Address offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| CTRL | R/W | 0x000 | Control register. Contains control and configuration fields. | 0x0 | 21.10.1 |
| STATUS | R/W | 0x004 | Status register. Contains the interrupt flags, busy, and XMAX information. | Part-number dependent | 21.10.2 |
| POLL_TCNT | R/W | 0x008 | Poll and measurement counter register. This sets up the polling counter and measurement counter rules. | 0x0 | 21.10.3 |
| INTENSET | R/W | 0x010 | Interrupt enable read and set. Write ‘1’ to set. | 0x0 | 21.10.4 |
| INTENCLR | WO | 0x014 | Interrupt enable clear. Write ‘1’ to clear. | 0x0 | 21.10.5 |
| INTSTAT | RO | 0x018 | Interrupt status. Logical AND of interrupt flags in STATUS register and corresponding bits in INTENSET register. | 0x0 | 21.10.6 |
| TOUCH | RO | 0x20 | Touch data register. TOUCH contains details of the most recent X sensor interrogation. | 0x0 | 21.10.7 |
| ID | RO | 0xFFC | Block ID | 0xE100 0000 | 21.10.8 |

### 21.10.1 Control register

The control register is used to specify the modes and parameters for the operation of the module.

It is recommended that the POLLMODE field is set to 0x0 before making changes to this register. No writes to this register should be made while INCHANGE (bit 15) equals ‘1’.

**Table 262. Control register (CTRL, offset 0x000) bit description**

| Bit | Symbol | Value | Description | Reset value | Access |
|---|---|---|---|---|---|
| 1:0 | POLLMODE | | Selects the method of polling. This field may only change from 0x0 to another value. Therefore, if software wishes to change between two active polling modes, it must first write 0x0 before selecting the new mode. | 0x0 | R/W |
| | | 0x0 | Inactive. | | |
| | | 0x1 | Poll Now. | | |
| | | 0x2 | Continuous. | | |
| | | 0x3 | Reserved. | | |
| 3:2 | TYPE | | Selects the polling type and sensor arrangement. | 0x0 | R/W |
| | | 0x0 | Normal. | | |
| | | 0x1 | Reserved. | | |
| | | 0x2 | Reserved. | | |
| | | 0x3 | Reserved. | | |
| 4 | TRIGGER | | Selects the measurement method. | 0x0 | R/W |
| | | 0x0 | YH port pin measurement. | | |
| | | 0x1 | Analog comparator measurement. | | |
| 5 | WAIT | | Controls when the next X measurement in the sequence may commence. | 0x0 | R/W |
| | | 0x0 | The next X measurement starts at the normal time. | | |
| | | 0x1 | When the ISTOUCH bit in the TOUCH register equals ‘1’, the module will wait until the TOUCH register is read before starting the next measurement. Other-wise, measurements continue. | | |
| 7:6 | - | | Reserved | 0x0 | R/W |
| 11:8 | FDIV | 0x0 – 0xF | Function clock divider. The function clock is divided by FDIV+1 to produce the divided FCLK for the module. | 0x0 | R/W |
| 13:12 | XPINUSE | | Determines how X pins enabled in the XPINSEL field are controlled when not active. | 0x0 | R/W |
| | | 0x0 | Inactive X pins are High-Z. | | |
| | | 0x1 | Inactive X pins are driven low. | | |
| | | 0x2 | Reserved. | | |
| | | 0x2 | Reserved. | | |
| 14 | - | - | Reserved. | - | - |
| 15 | INCHANGE | | Shows the status of the most recent update to the control register. | | |
| | | 0x0 | The last change has propagated. | 0x0 | RO |
| | | 0x1 | The last change has not propagated. Propagation time is dependent on synchronization between the bus clock and divided FCLK domains. Wait for IN-CHANGE = 0 before updating this register. | | |
| 31:16 | XPINSEL | | Selects which of the available X pins are enabled. Writing ‘1’ to a bit enables the corresponding X pin, ‘0’ disables.<br>Bit 16 (XPINSEL[0]) controls X0.<br>Bit 17 (XPINSEL[1]) controls X1.<br>Bit 18 (XPINSEL[2]) controls X2. | - | |

### 21.10.2 Status register

The status register indicates the status of the module in terms of touch events and polling. The interrupt flags are bits in this register, which are cleared by writing '1' to their bit position.

**Table 263. Status register (STATUS, offset 0x004) bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| 0 | YESTOUCH | Set if a touch has been detected, based on the count at which trigger occurred. | 0 | R/W1-to- clear |
| 1 | NOTOUCH | Set if a no-touch has been detected, based on the count at which trigger occurred. | 0 | R/W1-to- clear |
| 2 | POLLDONE | Set at the end of a polling round, or when a POLLNOW completes. | 0 | R/W1-to- clear |
| 3 | TIMEOUT | Set if the count reaches the time-out count value before a trigger occurs. | 0 | R/W1-to- clear |
| 4 | OVERRUN | Set if the Touch Data register has been up-dated before software has read the previous data, and the ISTOUCH bit in Touch Data equals '1'. Will not be set if ISTOUCH = '0' in which case touch data are silently overwritten. | 0 | R/W1-to- clear |
| 7:5 | - | Reserved. | - | - |
| 8 | BUSY | Set while a poll is currently in progress, otherwise cleared. | 0 | RO |
| 15:9 | - | Reserved. | - | - |
| 19:16 | XMAX | The maximum number of X pins available for a given device is equal to XMAX+1. | Part-number dependent | RO |

### 21.10.3 Poll and measurement counter register

The poll and measurement counter register contains the counting, delay, and threshold information, and is the basis for all measurements.
Make changes to this register while the POLLMODE field in the control register equals 0x0.

**Table 264. Poll and Measurement Counter Register (POLL_TCNT, offset 0x008) bit description**

| Bit | Symbol | Value | Description | Reset value | Access |
|---|---|---|---|---|---|
| 11:0 | TCNT | | Sets the count boundary in divided FCLKs between touch and no-touch, based on the TCHLOWER bit (Bit 31). | 0x000 | R/W |
| 15:12 | TOUT | | Sets the count value at which a time-out event occurs if a measurement has not triggered.<br>The time-out count value is calculated as $2^{TOUT}$.<br>TOUT must be less than 13, so the legal val-ues are 0, 1, ... 12. | 0x0 | R/W |
| 23:16 | POLL | | Sets the time delay between polling rounds (successive sets of X measurements). After each polling round completes, the module will wait 4096 x POLL divided FCLKs before starting the next polling round. | 0x00 | R/W |

**Table 264. Poll and Measurement Counter Register (POLL_TCNT, offset 0x008) bit description**

| Bit | Symbol | Value | Description | Reset value | Access |
|---|---|---|---|---|---|
| 25:24 | MDELAY | | Specifies the time delay after entering step 3 (measure voltage state), before sampling the YH port pin or analog comarator output. | 0x0 | R/W |
| | | 0x0 | Don't wait. | | |
| | | 0x1 | Wait 3 divided FCLKs. | | |
| | | 0x2 | Wait 5 divided FCLKs. | | |
| | | 0x3 | Wait 9 divided FCLKs. | | |
| 27:26 | RDELAY | | Specifies the number of divided FCLKs the module will remain in step 0 before starting the next X measurement in the polling round. | 0x0 | R/W |
| | | 0x0 | 1 divided FCLK. | | |
| | | 0x1 | 2 divided FCLKs. | | |
| | | 0x2 | 4 divided FCLKs. | | |
| | | 0x3 | 8 divided FCLKs. | | |
| 30:28 | - | | Reserved. | - | - |
| 31 | TCHLOW-ER | | Specifies whether a touched sensor triggers at a lower or higher count than an untouched sensor.<br>TOUCHLOWER = 0: Trigger at count > TCNT is a touch. Trigger at count ≤ TCNT is a no-touch.<br>TOUCHLOWER = 1: Trigger at count ≤ TCNT is a touch. Trigger at count > TCNT is a no-touch. | 0x0 | R/W |

### 21.10.4 Interrupt Enable Read and Set Register

The INTENSET register is used to set interrupt enables for individual interrupt flags in the STATUS register. An enabled interrupt can contribute to the module's interrupt request. Reading INTENSET returns the current values of the interrupt enables

Interrupt enables are set by writing '1' to their bit position in INTENSET. Writing '0' has no effect. Use INTENCLR to clear interrupt enables.

**Table 265. Interrupt Enable Read and Set Register (INTENSET, offset 0x010) bit description**

| Bit | Symbol | Value | Description | Reset value | Access |
|---|---|---|---|---|---|
| 0 | YESTOUCH | | | 0x0 | R/W1-to-set |
| | | 0 | Disabled | | |
| | | 1 | enabled | | |
| 1 | NOTOUCH | | | 0x0 | R/W1-to-set |
| | | 0 | Disabled | | |
| | | 1 | enabled | | |
| 2 | POLLDONE | | | 0x0 | R/W1-to-set |
| | | 0 | Disabled | | |
| | | 1 | enabled | | |
| 3 | TIMEOUT | | | 0x0 | R/W1-to-set |
| | | 0 | Disabled | | |
| | | 1 | enabled | | |

**Table 265. Interrupt Enable Read and Set Register (INTENSET, offset 0x010) bit description**

| Bit | Symbol | Value | Description | Reset value | Access |
|---|---|---|---|---|---|
| 4 | OVERRUN | | | 0x0 | R/W1-to-set |
| | | 0 | Disabled | | |
| | | 1 | enabled | | |

### 21.10.5 Interrupt Enable Clear Register

The write-only INTENCLR register is used to clear interrupt enables for individual interrupt flags in the STATUS register.

Interrupt enables are cleared by writing ‘1’ to their bit position in INTENCLR. Writing ‘0’ has no effect.

**Table 266. Interrupt Enable Clear Register (INTENCLR, offset 0x014) bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| 0 | YESTOUCH | Writing ‘1’ clears this interrupt enable. | 0x0 | W1-to-clear |
| 1 | NOTOUCH | Writing ‘1’ clears this interrupt enable. | 0x0 | W1-to-clear |
| 2 | POLLDONE | Writing ‘1’ clears this interrupt enable. | 0x0 | W1-to-clear |
| 3 | TIMEOUT | Writing ‘1’ clears this interrupt enable. | 0x0 | W1-to-clear |
| 4 | OVERRUN | Writing ‘1’ clears this interrupt enable. | 0x0 | W1-to-clear |

### 21.10.6 Interrupt Status Register

The INTSTAT register is used to view the status of enabled interrupts only. Reading INTSTAT returns the logical AND of the interrupt flags in the STATUS register with their corresponding interrupt enable bits.

**Table 267. Interrupt status register (INTSTAT, offset 0x018) bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| 0 | YESTOUCH | 0 = no interrupt, 1 = enabled and pending. | 0x0 | R/O |
| 1 | NOTOUCH | 0 = no interrupt, 1 = enabled and pending. | 0x0 | R/O |
| 2 | POLLDONE | 0 = no interrupt, 1 = enabled and pending. | 0x0 | R/O |
| 3 | TIMEOUT | 0 = no interrupt, 1 = enabled and pending. | 0x0 | R/O |
| 4 | OVERRUN | 0 = no interrupt, 1 = enabled and pending. | 0x0 | R/O |

### 21.10.7 Touch Data Register

The TOUCH register contains the data from the most recent X measurement.

**Table 268. Touch data register (TOUCH, offset 0x020) bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| 11:0 | COUNT | Contains the count value reached at trigger or time-out. | 0x0 | R/O |
| 15:12 | XVAL | Contains the index of the X pin for the current measurement, or lowest X for a multiple-pin poll now measurement. | 0x0 | R/O |
| 16 | ISTOUCH | ‘1’ if the trigger is due to a touch event, ‘0’ if the trigger is due to a no-touch event. | 0x0 | R/O |
| 17 | ISTO | ‘1’ if the measurement resulted in a time-out event, ‘0’ otherwise. | 0x0 | R/O |
| 19:18 | - | Reserved. | - | - |

**Table 268. Touch data register (TOUCH, offset 0x020) bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| 23:20 | SEQ | Contains the 4-bit sequence number, which increments at the end of each polling round. | 0x0 | R/O |
| 30:24 | - | Reserved. | - | - |
| 31 | CHANGE | Will be ‘1’ for one bus clock at the end of each X measurement while the data are changing, otherwise ‘0’. Touch data read while this bit is ‘1’ are invalid. | | |

### 21.10.8 ID register

The ID register identifies the type and revision of the module. A generic SW driver can make use of this information register to implement module-type or implementation-specific behavior.

**Table 269. ID register (ID, offset 0xFFC) bit description**

| Bit | Symbol | Description | Reset value | Access |
|---|---|---|---|---|
| 7:0 | APERTURE | Aperture: encoded as (aperture size/4K) -1, so 0x00 is a 4 K aperture. | 0x00 | R/O |
| 11:8 | MINOR_REV | Minor revision of module implementation, starting at 0. Software compatibility is expected between minor revisions. | 0x0 | R/O |
| 15:12 | MAJOR_REV | Major revision of module implementation, starting at 0. There may not be software compatibility between major revisions. | 0x0 | R/O |
| 31:16 | ID | Unique module identifier for this IP block. | 0xE100 | R/O |

## 22.1 How to read this chapter

The ADC is available on all parts. The number of available ADC channels depends on the package type.

**Table 270. Pinout summary**

| Package | ADC channels available |
|---|---|
| TSSOP20 | ADC_0 to ADC_11. |
| TSSOP24 package with dual VDDIO (Dual Supply) | ADC_0 to ADC_8, ADC_10 to ADC_11. |
| TSSOP24 (Single Supply) | ADC_0 to ADC_11. |
| HVQFN33 | ADC_0 to ADC_11. |

## 22.2 Features

- 12-bit successive approximation analog to digital converter.
- Input multiplexing among 12 pins.
- Two configurable conversion sequences with independent triggers.
- Optional automatic high/low threshold comparison and “zero crossing” detection.
- Power-down mode and low-power operating mode.
- Measurement range VREFN (GND) (typically 3 V; not to exceed VDDA voltage level).
- 12-bit conversion rate of up to 480 Ksamples/s.
- Burst conversion mode for single or multiple inputs.

## 22.3 Basic configuration

Configure the ADC as follows:

- Use the PDRUNCFG register to power the ADC. See Table 84. Once the ADC is powered by the PDRUNCFG register bit, the low-power mode bit in the ADC CTRL register can be used to turn off the ADC when it is not sampling and turn on the ADC automatically when any of the ADC conversion triggers are raised. See Table 275 and Section 22.7.5.
- Use the SYSAHBCLKCTRL register (Table 64) to enable the clock to the ADC register interface and the ADC clock.
- The ADC block creates four interrupts with individual entries in the NVIC. See Table 38.
- The ADC analog inputs are enabled in the switch matrix block.See Table 102.
- The power to the ADC block is controlled by the PDRUNCFG register in the SYSCON block. See Table 84.

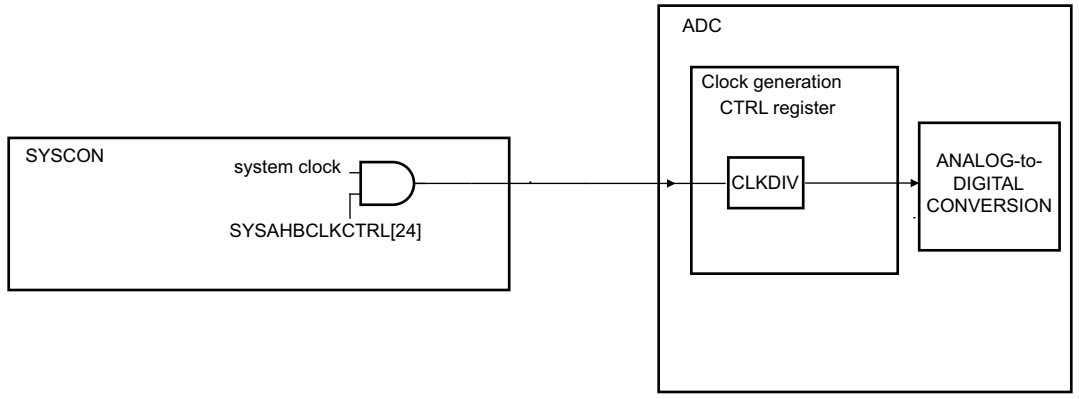



**Fig 46. ADC clocking**

### 22.3.1 Perform a single ADC conversion using a software trigger

**Remark:** When A/D conversions are triggered by software only and hardware triggers are not used in the conversion sequence, set the trigger source in the SEQA_CTRL and SEQB_CTRL registers to 0x0 (default).

Once the sequence is enabled, the ADC converts a sample whenever the START bit is written to. The TRIGPOL bit can be set in the same write that sets the SEQ_ENA and the START bits. Be careful not to modify the TRIGGER, TRIGPOL, and SEQ_ENA bits on subsequent writes to the START bit. See also Section 22.7.2.1 “Avoiding spurious hardware triggers”.

The ADC converts an analog input signal VIN on the ADC_[11:0]. The VREFP and VREFN pins provide a positive and negative reference voltage input. The result of the conversion is (4095 x (VIN-VREFN))/(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 ADC_1, follow these steps:

1. Enable the analog function ADC_1.
2. Configure the system clock to be 15 MHz and select a CLKDIV value of 0 for a sampling rate of 480 Ksamples/s using the ADC CTRL register.
3. Select ADC channel 1 to perform the conversion by setting the CHANNELS bits to 0x2 in the SEQA_CTL register.
4. Set the TRIGPOL bit to 1 and the SEQA_ENA bit to 1 in the SEQA_CTRL register.
5. Set the START bit to 1 in the SEQA_CTRL register.
6. Read the RESULT bits in the DAT1 register for the conversion result.

### 22.3.2 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 22.3.3.

To perform a single-step conversion on the first four channels of ADC0 triggered by a rising edge on PINT0 pin, follow these steps:

1. Enable the analog functions ADC_0 to ADC_3 through the switch matrix. See Table 273.
2. Configure the system clock to be 15 MHz and select a CLKDIV value of 0 for a sampling rate of 1 Msamples/s using the ADC CTRL register.
3. Select ADC channels 0 to 3 to perform the conversion by setting the CHANNELS bits to 0xF in the SEQA_CTL register.
4. Assign the input port PIO0_15 to be pin interrupt 0 by writing 0xF to PINTSEL[0] in SYSCON.
5. Configure the pin interrupt block for level-sensitive and active-high on pin interrupt 0, and enable it.
   - LPC_PIN_INT->ISEL |= 0x1; // level-sensitive
   - LPC_PIN_INT->IENF |= 0x1; // active high
   - LPC_PIN_INT->SIENR = 0x1; // enabled
6. Select PININT0_IRQ by writing 0x1 to 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 PIO0_15 goes from LOW to HIGH. The conversion on the next channel (channel 1) is triggered on the next rising edge of PIO0_15. The ADC_SEQA_IRQ interrupt is generated when the sequence has finished after four rising edges on PIO0_15.
10. Read the RESULT bits in the DAT0 to DAT3 registers for the conversion result.

### 22.3.3 ADC hardware trigger inputs

An analog-to-digital conversion can be initiated by a hardware trigger. You can select the trigger 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 276 “A/D Conversion Sequence A Control Register (SEQA_CTRL, address 0x4001 C008) bit description”
- Table 277 “A/D Conversion Sequence B Control Register (SEQB_CTRL, address 0x4001 C00C) bit description”

**Table 271. ADC hardware trigger inputs**

| Input # | Source | Description |
|---|---|---|
| 0 | - | No hardware trigger. |
| 1 | PININT0_IRQ | GPIO_INT interrupt 0. |
| 2 | PININT1_IRQ | GPIO_INT interrupt 1. |

**Table 271. ADC hardware trigger inputs**

| Input # | Source | Description |
|---|---|---|
| 5 | T0_MAT3 | CTIMER match 3. |
| 6 | CMP0_OUT_ADC | Analog comparator output. |
| 7 | GPIO_INT_BMAT | GPIO_INT bmatch. |
| 8 | ARM_TXEV | Arm core TXEV event. |

### 22.3.4 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 the sample period is 6.5 ADC clocks. With this default setting, the total duration of each A/D conversion is 31 ADC clocks. Under certain conditions longer sample times may be required. A variety of factors influence the required sampling time:

- Output impedance of the analog source driver.
- Operating conditions.
- ADC clock frequency.
- Selected ADC resolution (10 or 12 bits).
- Number of channels converted and if channel 0 (the slow channel) is included.

The number of additional clocks of sample time required can be programmed in the TSAMP field of the ADSEQA_CTRL and ADCSEQB_CTRL registers. Note that the TSAMP fields can be different for each sequence.

**Remark:** The TSAMP fields can be different for each sequence.

## 22.4 Pin description

The ADC cell can measure the voltage on any of the input signals on the analog input channel. Digital signals are disconnected from the ADC input pins when the ADC function is selected on that pin via the SWM.

**Remark:** If the ADC is used, signal levels on analog input pins must not be above the level of $V_{DD}$ 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 configured as digital I/O pins and are 5 V tolerant.

The VREFP and VREFN (GND) pins provide a positive and negative reference voltage input. The result of the conversion is (4095 x input voltage 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).

When the ADC is not used, tie VREFP to VDD and VREFP to $V_{SS}$.

**Remark:** For best performance, select VREFP at the same voltage levels as $V_{DD}$. When selecting VREFP different from VDD, ensure that the voltage midpoints are the same:

$$(VREFP-VREFN)/2 + VREFN = V_{DD}/2$$

**Table 272. ADC supply and reference voltage pins**

| Function | Description |
|---|---|
| $V_{REFP}$ | Positive voltage reference. The VREFP voltage level must be between 2.4 V and $V_{DDA}$. For best performance, select VREFP = $V_{DDA}$ and VREFN = $V_{SSA}$. |
| $V_{REFN}$ | GND pin. |
| $V_{DDA} = V_{DD}$ | The analog supply voltage is internally connected to $V_{DD}$. |
| $V_{SSA} = V_{SS}$ | ADC ground is internally connected to VSS. |

**Table 273. ADC pin description**

| Function | Direction | Type | Connect to | Use register | Reference | Description |
|---|---|---|---|---|---|---|
| ADC_0 | AI | external to pin | PIO0_1 | PINENABLE0 | Table 102 | Analog input channel 0. |
| ADC_1 | AI | external to pin | PIO0_7 | PINENABLE0 | Table 102 | Analog input channel 1. |
| ADC_2 | AI | external to pin | PIO0_14 | PINENABLE0 | Table 102 | Analog input channel 2. |
| ADC_3 | AI | external to pin | PIO0_16 | PINENABLE0 | Table 102 | Analog input channel 3. |
| ADC_4 | AI | external to pin | PIO0_9 | PINENABLE0 | Table 102 | Analog input channel 4. |
| ADC_5 | AI | external to pin | PIO0_8 | PINENABLE0 | Table 102 | Analog input channel 5. |
| ADC_6 | AI | external to pin | PIO0_11 | PINENABLE0 | Table 102 | Analog input channel 6. |
| ADC_7 | AI | external to pin | PIO0_10 | PINENABLE0 | Table 102 | Analog input channel 7. |
| ADC_8 | AI | external to pin | PIO0_15 | PINENABLE0 | Table 102 | Analog input channel 8. |
| ADC_9 | AI | external to pin | PIO0_17 | PINENABLE0 | Table 102 | Analog input channel 9. |
| ADC_10 | AI | external to pin | PIO0_13 | PINENABLE0 | Table 102 | Analog input channel 10. |
| ADC_11 | AI | external to pin | PIO0_4 | PINENABLE0 | Table 102 | Analog input channel 11. |

### 22.4.1 ADC versus digital receiver

The ADC function must be selected via the switch matrix registers in order to get accurate voltage readings on the monitored pin. The MODE bits in the IOCON register should also disable both pull-up and pull-down resistors. For a pin hosting an ADC input, it is not possible to have a have a digital function selected and yet get valid ADC readings. An inside circuit disconnects ADC hardware from the associated pin whenever a digital function is selected on that pin.

## 22.5 General description

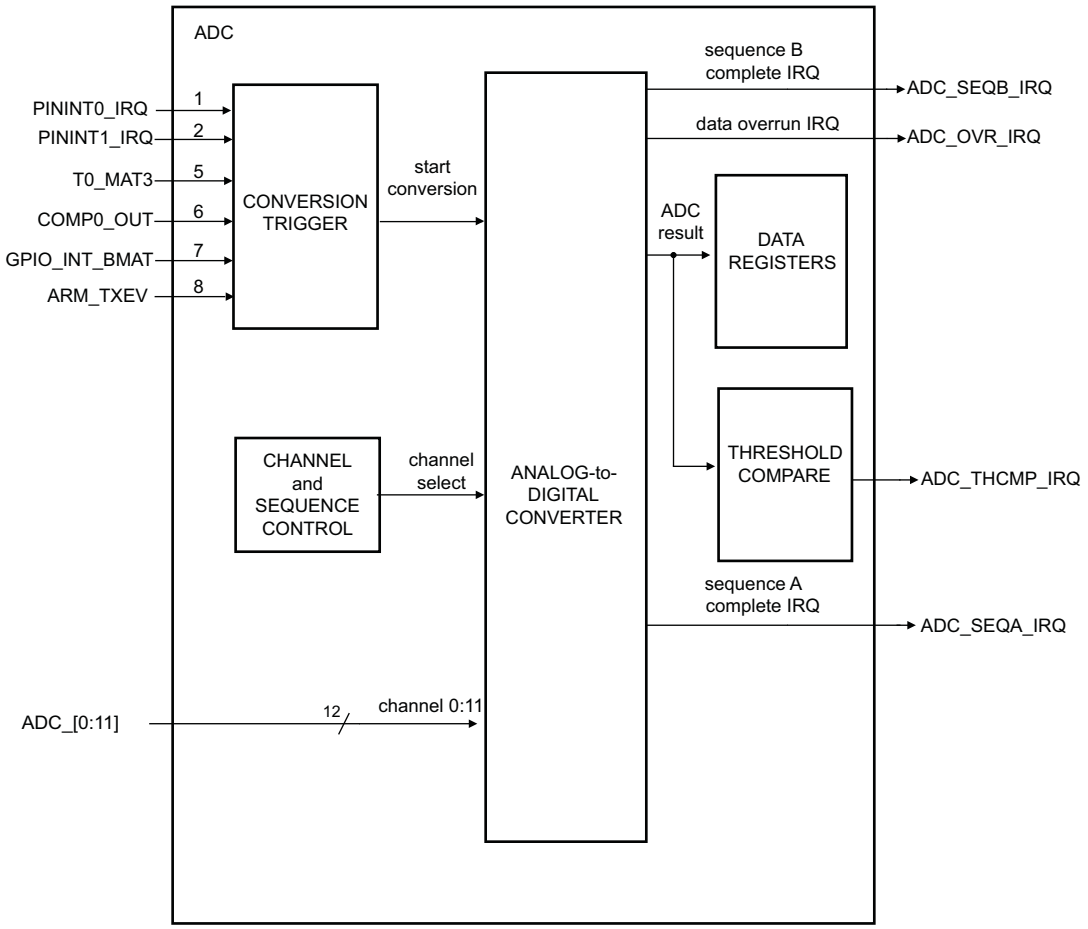



**Fig 47. ADC block diagram**

The ADC controller provides great flexibility in launching and controlling sequences of A/D conversions using the associated 12-bit, successive approximation A/D converter. A/D conversion sequences can be initiated under software control or in response to a selected hardware trigger. The ADC supports six hardware triggers.

Once the triggers are set up (software and hardware triggers can be mixed), the ADC runs through the pre-defined conversion sequence, 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 ADC clock is derived from the system clock. A programmable divider is included to scale the system clock to the maximum ADC clock rate of 15 MHz. The ADC clock drives the successive approximation process.

A fully accurate conversion requires 31 of these ADC clocks.

## 22.6 Register description

The reset value reflects the data stored in used bits only. It does not include reserved bits content.

**Table 274. Register overview : ADC (base address 0x4001 C000 )**

| Name | Access | Address offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| CTRL | R/W | 0x000 | A/D Control Register. Contains the clock divide value, enable bits for each sequence and the A/D power-down bit. | 0x0 | Table 275 |
| - | - | 0x004 | Reserved. | - | - |
| SEQA_CTRL | R/W | 0x008 | A/D Conversion Sequence-A control Register: Controls triggering and channel selection for conversion sequence-A. Also specifies interrupt mode for sequence-A. | 0x0 | Table 276 |
| SEQB_CTRL | RO | 0x00C | A/D Conversion Sequence-B Control Register: Controls triggering and channel selection for conversion sequence-B. Also specifies interrupt mode for sequence-B. | 0x0 | Table 277 |
| SEQA_GDAT | RO | 0x010 | A/D Sequence-A Global Data Register. This register contains the result of the most recent A/D conversion performed under sequence-A | NA | Table 278 |
| SEQB_GDAT | R/W | 0x014 | A/D Sequence-B Global Data Register. This register contains the result of the most recent A/D conversion performed under sequence-B | NA | Table 279 |
| DAT0 | RO | 0x020 | A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 0. | NA | Table 280 |
| DAT1 | RO | 0x024 | A/D Channel 1 Data Register. This register contains the result of the most recent conversion completed on channel 1. | NA | Table 280 |
| DAT2 | RO | 0x028 | A/D Channel 2 Data Register. This register contains the result of the most recent conversion completed on channel 2. | NA | Table 280 |
| DAT3 | RO | 0x02C | A/D Channel 3 Data Register. This register contains the result of the most recent conversion completed on channel 3. | NA | Table 280 |
| DAT4 | RO | 0x030 | A/D Channel 4 Data Register. This register contains the result of the most recent conversion completed on channel 4. | NA | Table 280 |
| DAT5 | RO | 0x034 | A/D Channel 5 Data Register. This register contains the result of the most recent conversion completed on channel 5. | NA | Table 280 |
| DAT6 | RO | 0x038 | A/D Channel 6 Data Register. This register contains the result of the most recent conversion completed on channel 6. | NA | Table 280 |
| DAT7 | RO | 0x03C | A/D Channel 7 Data Register. This register contains the result of the most recent conversion completed on channel 7. | NA | Table 280 |
| DAT8 | RO | 0x040 | A/D Channel 8 Data Register. This register contains the result of the most recent conversion completed on channel 7. | NA | Table 280 |
| DAT9 | RO | 0x044 | A/D Channel 9 Data Register. This register contains the result of the most recent conversion completed on channel 7. | NA | Table 280 |
| DAT10 | RO | 0x048 | A/D Channel 10 Data Register. This register contains the result of the most recent conversion completed on channel 7. | NA | Table 280 |
| DAT11 | RO | 0x04C | A/D Channel 11 Data Register. This register contains the result of the most recent conversion completed on channel 7. | NA | Table 280 |
| THR0_LOW | R/W | 0x050 | A/D Low Compare Threshold Register 0 : Contains the lower threshold level for automatic threshold comparison for any channels linked to threshold pair 0. | 0x0 | Table 281 |

**Table 274. Register overview : ADC (base address 0x4001 C000 )**

| Name | Access | Address offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| THR1_LOW | R/W | 0x054 | A/D Low Compare Threshold Register 1: Contains the lower threshold level for automatic threshold comparison for any channels linked to threshold pair 1. | 0x0 | Table 282 |
| THR0_HIGH | R/W | 0x058 | A/D High Compare Threshold Register 0: Contains the upper threshold level for automatic threshold comparison for any channels linked to threshold pair 0. | 0x0 | Table 283 |
| THR1_HIGH | R/W | 0x05C | A/D High Compare Threshold Register 1: Contains the upper threshold level for automatic threshold comparison for any channels linked to threshold pair 1. | 0x0 | Table 284 |
| CHAN_THRSEL | R/W | 0x060 | A/D Channel-Threshold Select Register. Specifies which set of threshold compare registers are to be used for each channel | 0x0 | Table 285 |
| INTEN | R/W | 0x064 | A/D Interrupt Enable Register. This register contains enable bits that enable the sequence-A, sequence-B, threshold compare and data overrun interrupts to be generated. | 0x0 | Table 286 |
| FLAGS | R/W | 0x068 | A/D Flags Register. Contains the four interrupt request flags and the individual component overrun and threshold-compare flags. (The overrun bits replicate information stored in the result registers). | 0x0 | Table 287 |

### 22.6.1 ADC Control Register

This register specifies the clock divider value to be used to generate the ADC clock and general operating mode bits including a low power mode that allows the A/D to be turned off to save power when not in use.

**Table 275. A/D Control Register (CTRL, addresses 0x4001 C000) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 7:0 | CLKDIV | | The system clock is divided by this value plus one to produce the sampling clock. The sampling clock should be less than or equal to 15 MHz for 480 Ksamples/s.<br>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. | 0 |
| 9:8 | - | | Reserved. Do not write a one to these bits. | 0 |

**Table 275. A/D Control Register (CTRL, addresses 0x4001 C000) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 10 | LPWRMODE | | Select low-power ADC mode.<br>The analog circuitry is automatically powered-down when no conversions are taking place. When any (hardware or software) triggering event is detected, the analog circuitry is enabled. After the required start-up time, the requested conversion will be launched. Once the conversion completes, the analog-circuitry will again be powered-down provided no further conversions are pending.<br>Using this mode can save an appreciable amount of current when conversions are required relatively infrequently.<br>The penalty for using this mode is an approximately 15 ADC clock delay, based on the frequency specified in the CLKDIV field, from the time the trigger event occurs until sampling of the A/D input commences.<br>**Remark:** This mode will NOT power-up the ADC when the ADC analog block is powered down in the system control block. | 0 |
| | | 0 | Disabled. The low-power ADC mode is disabled.<br>The analog circuitry remains activated even when no conversions are requested. | |
| | | 1 | Enabled. The low-power ADC mode is enabled. | |
| 29:11 | | | Reserved, do not write ones to reserved bits. | 0 |
| 31:30 | - | | Reserved. | 0 |

### 22.6.2 A/D 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.

To avoid conversions on spurious triggers, only change the trigger configuration when the conversion sequence is disabled. A conversion can be triggered by software or hardware in the conversion sequence, but if conversions are triggered by software only, spurious hardware triggers must be prevented. See Section 22.3.1 “Perform a single ADC conversion using a software trigger”.

**Remark:** Set the BURST and SEQU_ENA bits at the same time.

**Table 276. A/D Conversion Sequence A Control Register (SEQA_CTRL, address 0x4001 C008) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 11:0 | CHANNELS | | Selects which one or more of the twelve 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.<br>When this conversion sequence is triggered, either by a hardware trigger or via software command, A/D conversions will be performed on each enabled channel, in sequence, beginning with the lowest-ordered channel.<br>**Remark:** This field can ONLY be changed while the SEQA_ENA bit (bit 31) is LOW. It is allowed to change this field and set bit 31 in the same write. | 0x00 |
| 14:12 | TRIGGER | | Selects which of the available hardware trigger sources will cause this conversion sequence to be initiated. Program the trigger input number in this field.<br>**Remark:** In order to avoid generating a spurious trigger, it is recommended writing to this field only when the SEQA_ENA bit (bit 31) is low. It is safe to change this field and set bit 31 in the same write. | 0x0 |
| 17:15 | - | | Reserved. | - |
| 18 | TRIGPOL | | Select the polarity of the selected input trigger for this conversion sequence.<br>**Remark:** In order to avoid generating a spurious trigger, it is recommended writing to this field only when the SEQA_ENA bit (bit 31) is low. It is safe to change this field and set bit 31 in the same write. | 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 | | Setting this bit allows the hardware trigger input to bypass synchronization flip-flops 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:<br>Synchronous mode: 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.<br>Asynchronous mode: 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 and 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. | 0 |
| | | 0 | Enable synchronization. The hardware trigger bypass is not enabled. | |
| | | 1 | Bypass synchronization. The hardware trigger bypass is enabled. | |
| 25:20 | - | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | N/A |
| 26 | START | | 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.<br>**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. | 0 |

**Table 276. A/D Conversion Sequence A Control Register (SEQA_CTRL, address 0x4001 C008) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 27 | BURST | | 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.<br>Repeated conversions can be halted by clearing this bit. The sequence currently in progress will be completed before conversions are terminated. | 0 |
| 28 | SINGLESTEP | | 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.<br>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. | 0 |
| 29 | LOWPRIO | | Set priority for sequence A. | 0 |
| | | 0 | Low priority. Any B trigger which occurs while an A conversion sequence is active will be ignored and lost. | |
| | | 1 | High priority.<br>Setting this bit to a 1 will permit any enabled B sequence trigger (including a B sequence software start) to immediately interrupt this sequence and launch a B sequence in it's place. The conversion currently in progress will be terminated.<br>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. | |
| 30 | MODE | | 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.<br>Impacts when conversion-complete interrupt triggers for sequence-A will be generated and which overrun conditions contribute to an overrun interrupt as described below: | 0 |
| | | 0 | End of conversion. The sequence A interrupt flag will be set at the end of each individual A/D conversion performed under sequence A. This flag will mirror the DATAVALID bit in the SEQA_GDAT register.<br>The OVERRUN bit in the SEQA_GDAT register will contribute to generation of an overrun interrupt if enabled. | |
| | | 1 | End of sequence. The sequence A interrupt flag will be set when the entire set of sequence-A conversions completes. This flag will need to be explicitly cleared by software in this mode.<br>The OVERRUN bit in the SEQA_GDAT register will NOT contribute to generation of an overrun interrupt trigger since it is assumed this register may not be utilized in this mode. | |

**Table 276. A/D Conversion Sequence A Control Register (SEQA_CTRL, address 0x4001 C008) 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. | 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. | |

### 22.6.3 A/D 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.

To avoid conversions on spurious triggers, only change the trigger configuration when the conversion sequence is disabled. A conversion can be triggered by software or hardware in the conversion sequence, but if conversions are triggered by software only, spurious hardware triggers must be prevented. See Section 22.3.1 “Perform a single ADC conversion using a software trigger”.

**Remark:** Set the BURST and SEQB_ENA bits at the same time.

**Table 277. A/D Conversion Sequence B Control Register (SEQB_CTRL, address 0x4001 C00C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 11:0 | CHANNELS | | Selects which one or more of the twelve 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.<br>When this conversion sequence is triggered, either by a hardware trigger or via software command, A/D conversions will be performed on each enabled channel, in sequence, beginning with the lowest-ordered channel.<br>**Remark:** This field can ONLY be changed while the SEQB_ENA bit (bit 31) is LOW. It is permissible to change this field and set bit 31 in the same write. | 0x00 |
| 14:12 | TRIGGER | | Selects which of the available hardware trigger sources will cause this conversion sequence to be initiated. Program the trigger input number in this field.<br>**Remark:** In order to avoid generating a spurious trigger, it is recommended writing to this field only when the SEQA_ENA bit (bit 31) is low. It is safe to change this field and set bit 31 in the same write. | 0x0 |
| 17:15 | - | | Reserved. | - |

**Table 277. A/D Conversion Sequence B Control Register (SEQB_CTRL, address 0x4001 C00C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 18 | TRIGPOL | | Select the polarity of the selected input trigger for this conversion sequence.<br>**Remark:** In order to avoid generating a spurious trigger, it is recommended writing to this field only when the SEQA_ENA bit (bit 31) is low. It is safe to change this field and set bit 31 in the same write. | 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 | | Setting this bit allows the hardware trigger input to bypass synchronization flip-flops 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:<br>Synchronous mode: 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.<br>Asynchronous mode: 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 and 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. | 0 |
| | | 0 | Enable synchronization. The hardware trigger bypass is not enabled. | |
| | | 1 | Bypass synchronization. The hardware trigger bypass is enabled. | |
| 25:20 | - | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | N/A |
| 26 | START | | 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 a 1 to this bit if the BURST bit is set.<br>**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. | 0 |
| 27 | BURST | | Writing a 1 to this bit will cause this conversion sequence to be continuously cycled through. Other B triggers will be ignored while this bit is set.<br>Repeated conversions can be halted by clearing this bit. The sequence currently in progress will be completed before conversions are terminated. | 0 |

**Table 277. A/D Conversion Sequence B Control Register (SEQB_CTRL, address 0x4001 C00C) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 28 | SINGLESTEP | | 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.<br><br>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. | 0 |
| 29 | - | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | N/A |
| 30 | MODE | | 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.<br><br>Impacts when conversion-complete interrupt trigger for sequence-B will be generated and which overrun conditions contribute to an overrun interrupt as described below: | 0 |
| | | 0 | End of conversion. The sequence B interrupt flag will be set at the end of each individual A/D conversion performed under sequence B. This flag will mirror the DATAVALID bit in the SEQB_GDAT register.<br><br>The OVERRUN bit in the SEQB_GDAT register will contribute to generation of an overrun interrupt if enabled. | |
| | | 1 | End of sequence. The sequence B interrupt flag will be set when the entire set of sequence B conversions completes. This flag will need to be explicitly cleared by software in this mode.<br><br>The OVERRUN bit in the SEQB_GDAT register will NOT contribute to generation of an overrun interrupt since it is assumed this register will not be utilized in this mode. | |
| 31 | SEQB_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. | 0 |
| | | 0 | 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. | |
| | | 1 | Enabled. Sequence B is enabled. | |

### 22.6.4 A/D Global Data Register A and B

The A/D Global Data Registers contain the result of the most recent A/D conversion completed under each conversion sequence.

Results of A/D conversions can be read in one of two ways. One is to use these A/D Global Data Registers to read data from the ADC at the end of each A/D conversion. Another is to read the individual A/D 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 ADSEQn_CTRL register since this will impact interrupt and overrun flag generation.

**Table 278. A/D Sequence A Global Data Register (SEQA_GDAT, address 0x4001 C010) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 3:0 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 15:4 | RESULT | This field contains the 12-bit A/D conversion result from the most recent conversion performed under conversion sequence associated with this register.<br><br>The result is the a binary fraction representing the voltage on the currently-selected input channel 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 $V_{REFP}$.<br><br>DATAVALID = 1 indicates that this result has not yet been read. | NA |
| 17:16 | THCMPRANGE | 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 | THCMPCROSS | 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, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | 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...). | NA |
| 30 | OVERRUN | 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.<br><br>This bit will contribute to an overrun interrupt request if the MODE bit (in SEQA_CTRL) for the corresponding sequence is set to ‘0’ (and if the overrun interrupt is enabled). | 0 |
| 31 | DATAVALID | 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.<br><br>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 279. A/D Sequence B Global Data Register (SEQB_GDAT, address 0x4001 C014) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 3:0 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 15:4 | RESULT | This field contains the 12-bit A/D conversion result from the most recent conversion performed under conversion sequence associated with this register.<br><br>This will be a binary fraction representing the voltage on the currently-selected input channel 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 $V_{REFP}$.<br><br>DATAVALID = 1 indicates that this result has not yet been read. | NA |
| 17:16 | THCMPRANGE | 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).<br>Threshold Range Comparison result.<br>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).<br>0x1 = Below Range: The last completed conversion on was less than the value programmed into the designated LOW threshold register (THRn_LOW).<br>0x2 = Above Range: The last completed conversion was greater than the value programmed into the designated HIGH threshold register (THRn_HIGH).<br>0x3 = Reserved. | |
| 19:18 | THCMPCROSS | 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.<br>0x0 = No threshold Crossing detected:<br>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.<br>0x1 = Reserved.<br>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.<br>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. | |
| 25:20 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

**Table 279. A/D Sequence B Global Data Register (SEQB_GDAT, address 0x4001 C014) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 29:26 | CHN | These bits contain the channel from which the RESULT bits were converted (e.g. 0b0000 identifies channel 0, 0b0001 channel 1...). | NA |
| 30 | OVERRUN | 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.<br><br>This bit will contribute to an overrun interrupt request if the MODE bit (in SEQB_CTRL) for the corresponding sequence is set to 0 (and if the overrun interrupt is enabled). | 0 |
| 31 | DATAVALID | 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.<br><br>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). | 0 |

### 22.6.5 A/D Channel Data Registers 0 to 11

The A/D Channel Data Registers hold the result of the last conversion completed for each A/D 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 A/D conversion can be read in one of two ways. One is to use the A/D Global Data Registers for each of the sequences to read data from the ADC at the end of each A/D conversion. Another is to use these individual A/D 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 280. A/D Data Registers (DAT[0:11], address 0x4001 C020 (DAT0) to 0x4001 C04C (DAT11)) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 3:0 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 15:4 | RESULT | This field contains the 12-bit A/D 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 | THCMPRANGE | Threshold Range Comparison result.<br>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).<br>0x1 = Below Range: The last completed conversion on was less than the value programmed into the designated LOW threshold register (THRn_LOW).<br>0x2 = Above Range: The last completed conversion was greater than the value programmed into the designated HIGH threshold register (THRn_HIGH).<br>0x3 = Reserved. | NA |
| 19:18 | THCMPCROSS | Threshold Crossing Comparison result.<br>0x0 = No threshold Crossing detected:<br>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.<br>0x1 = Reserved.<br>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.<br>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. | NA |
| 25:20 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

**Table 280. A/D Data Registers (DAT[0:11], address 0x4001 C020 (DAT0) to 0x4001 C04C (DAT11)) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 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 | OVERRUN | 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.<br>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.<br>This bit (in any of the 12 registers) will cause an overrun interrupt request to be asserted if the overrun interrupt is enabled.<br>**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. | NA |
| 31 | DATAVALID | This bit is set to 1 when an A/D conversion on this channel completes.<br>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.<br>**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. | NA |

### 22.6.6 A/D Compare Low Threshold Registers 0 and 1

These registers set the LOW threshold levels against which A/D 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 THCMPRANGE status bits for that channel to be set to 0b01. This result will also generate an interrupt request 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, than a threshold crossing has occurred. In this case the MSB of the THCMPCROSS 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 request if enabled to do so via the ADCMPINTEN bits associated with each channel in the INTEN register.

**Table 281. A/D Compare Low Threshold register 0 (THR0_LOW, address 0x4001 C050) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 3:0 | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 15:4 | THRLOW | Low threshold value against which A/D results will be compared | 0x000 |
| 31:16 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

**Table 282. A/D Compare Low Threshold register 1 (THR1_LOW, address 0x4001 C054) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 3:0 | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 15:4 | THRLOW | Low threshold value against which A/D results will be compared | 0x000 |
| 31:16 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

### 22.6.7 A/D Compare High Threshold Registers 0 and 1

These registers set the HIGH threshold level against which A/D 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 request if enabled to do so via the ADCMPINTEN bits associated with each channel in the INTEN register.

**Table 283. Compare High Threshold register0 (THR0_HIGH, address 0x4001 C058) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 3:0 | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 15:4 | THRHIGH | High threshold value against which A/D results will be compared | 0x000 |
| 31:16 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

**Table 284. Compare High Threshold register 1 (THR1_HIGH, address 0x4001 C05C) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 3:0 | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 15:4 | THRHIGH | High threshold value against which A/D results will be compared | 0x000 |
| 31:16 | - | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

### 22.6.8 A/D Channel Threshold Select register

For each channel, this register indicates which pair of threshold registers conversion results should be compared to.

**Table 285. A/D Channel Threshold Select register (CHAN_THRSEL, addresses 0x4001 C060) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | CH0_THRSEL | | Threshold select by channel. | 0 |
| | | 0 | Threshold 0. Channel 0 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers | |
| | | 1 | Threshold 1. Channel 0 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers | |
| 1 | CH1_THRSEL | | Threshold select by channel. | 0 |
| | | 0 | Threshold 0. Channel 1 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers | |
| | | 1 | Threshold 1. Channel 1 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers | |
| 2 | CH2_THRSEL | | Threshold select by channel. | 0 |
| | | 0 | Threshold 0. Channel 2 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers | |
| | | 1 | Threshold 1. Channel 2 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers | |
| 3 | CH3_THRSEL | | Threshold select by channel. | 0 |
| | | 0 | Threshold 0. Channel 3 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers | |
| | | 1 | Threshold 1. Channel 3 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers | |
| 4 | CH4_THRSEL | | Threshold select by channel. | 0 |
| | | 0 | Threshold 0. Channel 4 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers | |
| | | 1 | Threshold 1. Channel 4 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers | |
| 5 | CH5_THRSEL | | Threshold select by channel. | 0 |
| | | 0 | Threshold 0. Channel 5 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers | |
| | | 1 | Threshold 1. Channel 5 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers | |

**Table 285. A/D Channel Threshold Select register (CHAN_THRSEL, addresses 0x4001 C060) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 6 | CH6_THRSEL | | Threshold select by channel. | 0 |
| | | 0 | Threshold 0. Channel 6 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers | |
| | | 1 | Threshold 1. Channel 6 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers | |
| 7 | CH7_THRSEL | | Threshold select by channel. | 0 |
| | | 0 | Threshold 0. Channel 7 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers | |
| | | 1 | Threshold 1. Channel 7 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers | |
| 8 | CH8_THRSEL | | Threshold select by channel. | 0 |
| | | 0 | Threshold 0. Channel 8 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers | |
| | | 1 | Threshold 1. Channel 8 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers | |
| 9 | CH9_THRSEL | | Threshold select by channel. | 0 |
| | | 0 | Threshold 0. Channel 9 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers | |
| | | 1 | Threshold 1. Channel 9 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers | |
| 10 | CH10_THRSEL | | Threshold select by channel. | 0 |
| | | 0 | Threshold 0. Channel 10 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers | |
| | | 1 | Threshold 1. Channel 10 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers | |
| 11 | CH11_THRSEL | | Threshold select by channel. | 0 |
| | | 0 | Threshold 0. Channel 11 results will be compared against the threshold levels indicated in the THR0_LOW and THR0_HIGH registers | |
| | | 1 | Threshold 1. Channel 11 results will be compared against the threshold levels indicated in the THR1_LOW and THR1_HIGH registers | |
| 31:12 | | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

### 22.6.9 A/D Interrupt Enable Register

There are four separate interrupt requests generated by the ADC: conversion-complete or sequence-complete interrupts for each of the two sequences, a threshold-comparison out-of-range interrupt, and a data overrun interrupt.

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 A/D threshold-compare interrupt request.

- 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 A/D threshold-compare interrupt request.
- Crossing threshold: Detection of a threshold crossing on channel n will set the channel n THCMP flag in the ADFLAGS register and generate an A/D threshold-compare interrupt request.

**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 286. A/D Interrupt Enable register (INTEN, address 0x4001 C064 ) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | SEQA_INTEN | | Sequence A interrupt enable. | 0 |
| | | 0 | Disabled. The sequence A interrupt trigger is disabled. | |
| | | 1 | Enabled. The sequence A interrupt 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 trigger is disabled. | |
| | | 1 | Enabled. The sequence B interrupt 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 request.<br>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 request to be asserted. | |
| 4:3 | ADCMPINTEN0 | | Threshold comparison interrupt enable. | 00 |
| | | 0x0 | Disabled. | |
| | | 0x1 | Outside threshold. | |
| | | 0x2 | Crossing threshold. | |
| | | 0x3 | Reserved | |
| 6:5 | ADCMPINTEN1 | | Threshold comparison interrupt enable. | 00 |
| | | 0x0 | Disabled. | |
| | | 0x1 | Outside threshold. | |
| | | 0x2 | Crossing threshold. | |
| | | 0x3 | Reserved. | |
| 8:7 | ADCMPINTEN2 | | Threshold comparison interrupt enable. | 00 |
| | | 0x0 | Disabled. | |
| | | 0x1 | Outside threshold. | |

**Table 286. A/D Interrupt Enable register (INTEN, address 0x4001 C064 ) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| | | 0x2 | Crossing threshold. | |
| | | 0x3 | Reserved | |
| 10:9 | ADCMPINTEN3 | | Threshold comparison interrupt enable. | 00 |
| | | 0x0 | Disabled. | |
| | | 0x1 | Outside threshold. | |
| | | 0x2 | Crossing threshold. | |
| | | 0x3 | Reserved | |
| 12:11 | ADCMPINTEN4 | | Threshold comparison interrupt enable. | 00 |
| | | 0x0 | Disabled. | |
| | | 0x1 | Outside threshold. | |
| | | 0x2 | Crossing threshold. | |
| | | 0x3 | Reserved | |
| 14:13 | ADCMPINTEN5 | | Threshold comparison interrupt enable. | 00 |
| | | 0x0 | Disabled. | |
| | | 0x1 | Outside threshold. | |
| | | 0x2 | Crossing threshold. | |
| | | 0x3 | Reserved | |
| 16:15 | ADCMPINTEN6 | | Threshold comparison interrupt enable. | 00 |
| | | 0x0 | Disabled. | |
| | | 0x1 | Outside threshold. | |
| | | 0x2 | Crossing threshold. | |
| | | 0x3 | Reserved. | |
| 18:17 | ADCMPINTEN7 | | Threshold comparison interrupt enable. | 00 |
| | | 0x0 | Disabled. | |
| | | 0x1 | Outside threshold. | |
| | | 0x2 | Crossing threshold. | |
| | | 0x3 | Reserved | |
| 20:19 | ADCMPINTEN8 | | Threshold comparison interrupt enable. | 00 |
| | | 0x0 | Disabled. | |
| | | 0x1 | Outside threshold. | |
| | | 0x2 | Crossing threshold. | |
| | | 0x3 | Reserved | |
| 22:21 | ADCMPINTEN9 | | Threshold comparison interrupt enable. | 00 |
| | | 0x0 | Disabled. | |
| | | 0x1 | Outside threshold. | |
| | | 0x2 | Crossing threshold. | |
| | | 0x3 | Reserved | |
| 24:23 | ADCMPINTEN10 | | Threshold comparison interrupt enable. | 00 |
| | | 0x0 | Disabled. | |
| | | 0x1 | Outside threshold. | |

**Table 286. A/D Interrupt Enable register (INTEN, address 0x4001 C064 ) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| | | 0x2 | Crossing threshold. | |
| | | 0x3 | Reserved | |
| 26:25 | ADCMPINTEN11 | | Threshold comparison interrupt enable. | 00 |
| | | 0x0 | Disabled. | |
| | | 0x1 | Outside threshold. | |
| | | 0x2 | Crossing threshold. | |
| | | 0x3 | Reserved | |
| 31:27 | - | | Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |

### 22.6.10 A/D Flag register

The A/D Flags registers contains the four interrupt request flags along with the individual overrun flags that contribute to an overrun interrupt and the component threshold-comparison flags that contribute to that interrupt.

The channel OVERRUN flags, mirror those in the appearing in the individual ADDAT registers for each channel, 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).

**Table 287. A/D Flags register (FLAGS, address 0x4001 C068) 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. 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 |
| 2 | THCMP2 | Threshold comparison event on Channel 2. 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 |
| 3 | THCMP3 | Threshold comparison event on Channel 3. 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 |
| 4 | THCMP4 | Threshold comparison event on Channel 4. 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 |
| 5 | THCMP5 | Threshold comparison event on Channel 5. 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 |
| 6 | THCMP6 | Threshold comparison event on Channel 6. 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 |

**Table 287. A/D Flags register (FLAGS, address 0x4001 C068) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 7 | THCMP7 | Threshold comparison event on Channel 7. 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 |
| 8 | THCMP8 | Threshold comparison event on Channel 8. 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 |
| 9 | THCMP9 | Threshold comparison event on Channel 9. 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 |
| 10 | THCMP10 | Threshold comparison event on Channel 10. 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 |
| 11 | THCMP11 | Threshold comparison event on Channel 11. 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 |
| 12 | OVERRUN0 | Mirrors the OVERRRUN status flag from the result register for A/D channel 0 | 0 |
| 13 | OVERRUN1 | Mirrors the OVERRRUN status flag from the result register for A/D channel 1 | 0 |
| 14 | OVERRUN2 | Mirrors the OVERRRUN status flag from the result register for A/D channel 2 | 0 |
| 15 | OVERRUN3 | Mirrors the OVERRRUN status flag from the result register for A/D channel 3 | 0 |
| 16 | OVERRUN4 | Mirrors the OVERRRUN status flag from the result register for A/D channel 4 | 0 |
| 17 | OVERRUN5 | Mirrors the OVERRRUN status flag from the result register for A/D channel 5 | 0 |
| 18 | OVERRUN6 | Mirrors the OVERRRUN status flag from the result register for A/D channel 6 | 0 |
| 19 | OVERRUN7 | Mirrors the OVERRRUN status flag from the result register for A/D channel 7 | 0 |
| 20 | OVERRUN8 | Mirrors the OVERRRUN status flag from the result register for A/D channel 8 | 0 |
| 21 | OVERRUN9 | Mirrors the OVERRRUN status flag from the result register for A/D channel 9 | 0 |
| 22 | OVERRUN10 | Mirrors the OVERRRUN status flag from the result register for A/D channel 10 | 0 |
| 23 | OVERRUN11 | Mirrors the OVERRRUN status flag from the result register for A/D 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, user software should not write ones to reserved bits. The value read from a reserved bit is not defined. | NA |
| 28 | SEQA_INT | Sequence A interrupt flag.<br>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 A/D conversion performed as part of sequence A. It will be cleared automatically when the SEQA_GDAT register is read.<br>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.<br>This interrupt must be enabled in the INTEN register. | 0 |

**Table 287. A/D Flags register (FLAGS, address 0x4001 C068) bit description**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 29 | SEQB_INT | Sequence A interrupt flag.<br>If the MODE bit in the SEQB_CTRL register is 0, this flag will mirror the DATAVALID bit in the sequence A global data register (SEQB_GDAT), which is set at the end of every A/D conversion performed as part of sequence B. It will be cleared automatically when the SEQB_GDAT register is read.<br>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.<br>This interrupt must be enabled in the INTEN register. | 0 |
| 30 | THCMP_INT | Threshold Comparison Interrupt flag.<br>This bit will be set if any of the 12 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).<br>Each type of threshold comparison interrupt on each channel must be individually enabled in the INTEN register to cause this interrupt.<br>This bit will be cleared when all of the component flags in bits 11:0 are cleared via writing 1s to those bits. | 0 |
| 31 | OVR_INT | Overrun Interrupt flag.<br>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.<br>This interrupt must be enabled in the INTEN register.<br>This bit will be cleared when all of the individual overrun bits have been cleared via reading the corresponding data registers. | 0 |

## 22.7 Functional description

### 22.7.1 Conversion Sequences

A conversion sequence is a single pass through a series of A/D conversions performed on a selected set of A/D channels. Software can set-up 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. It is not necessary to employ both sequences.

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.

### 22.7.2 Hardware-triggered conversion

Software can select which of these 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 ADSEQn_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 A/D 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.

#### 22.7.2.1 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_CTRL register should only be written to when the sequence is disabled (while the SEQn_ENA bit is LOW). 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.

### 22.7.3 Software-triggered conversion

There are two ways that software can trigger a conversion sequence:

1. Start Bit: The first way to software-trigger an sequence is by 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: The other way to initiate conversions is to 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 SEQA_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.

### 22.7.4 Interrupts

There are four interrupts that can be generated by the ADC:

- Conversion-Complete or Sequence-Complete interrupts for sequences A and 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.

#### 22.7.4.1 Conversion-Complete or Sequence-Complete interrupts

For each of the two sequences, an interrupt request can either be asserted at the end of each A/D 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 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 request.

If the MODE bit for the sequence is 1 (sequence-complete mode) then the interrupt flag must be written-to by software to clear it.

#### 22.7.4.2 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.

#### 22.7.4.3 Data Overrun Interrupt

This interrupt request 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 request 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.

### 22.7.5 Optional operating modes

The following optional mode of A/D operation may be selected in the CTRL register:

Low-power mode. When this mode is selected, the analog portions of the ADC are automatically shut down when no conversions are in progress. The ADC is automatically restarted whenever any hardware or software trigger event occurs. This mode can save an appreciable amount of power when the ADC is not in continuous use, but at the expense of a delay between the trigger event and the onset of sampling and conversion.

### 22.7.6 Hardware Trigger Source Selection

Each ADC has a selection of several on-chip and off-chip hardware trigger sources. The trigger to be used for each conversion sequence is specified in the TRIGGER fields in the two SEQn_CTRL registers.

## 23.1 Basic configuration

The DAC is configured using the following registers:

1. Power: Use the PDRUNCFG register to power the DAC. See Section 6.6.30.
2. Peripheral clock: Use the SYSAHBCLKCTRL register to enable the clock to the DAC register interface. See Section 6.6.10.
3. Pins: Enable the DAC pin and select the PIO0_19 pin mode for DACOUT through the relevant SWM and IOCON register (Table 105). This must be done before accessing any DAC registers.

## 23.2 Features

- 10-bit digital to analog converter
- Resistor string architecture
- Buffered output
- Power-down mode
- Selectable speed vs. power
- Maximum update rate of 1 MHz.

## 23.3 Architecture

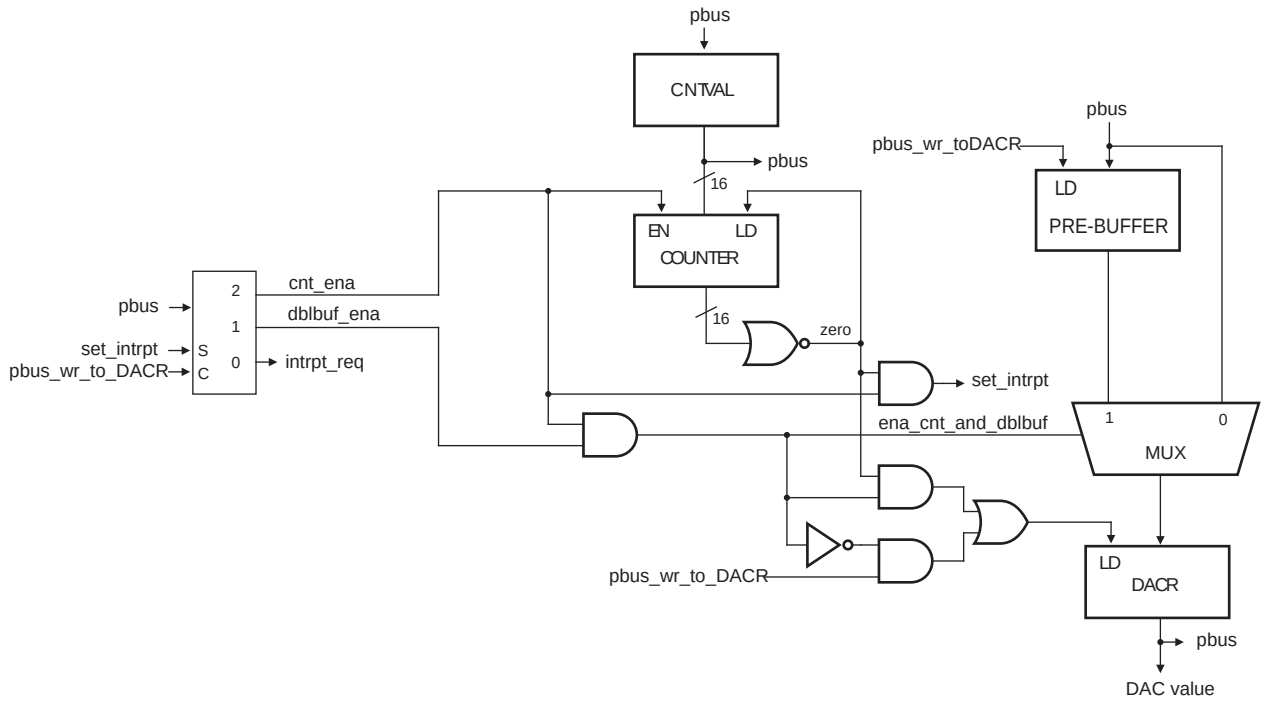



**Fig 48. DAC control with interrupt and timer**

## 23.4 Pin description

Table 288 gives a brief summary of each of DAC related pins.

**Table 288. D/A Pin Description**

| Pin | Type | Description |
|---|---|---|
| DAC_OUT | Output | **Analog Output.** After the selected settling time after the DACR is written with a new value, the voltage on this pin (with respect to $V_{SSA}$) is VALUE × (($V_{REFP}$)/1024). Note that DAC_OUT is disabled when the CPU is in Deep-sleep, Power-down, or Deep Power-down modes. |
| $V_{REFP}$ | Reference | **Voltage Reference.** This pin provides a voltage reference level for the ADC and DAC. Note: $V_{REFP}$ should be tied to VDD if the ADC and DAC are not used. |
| $V_{DD}$ and $V_{SS}$ | Power | **Power and Ground.** |

## 23.5 Register description

**Table 289. Register overview: DAC (base address 0x4001 4000)**

| Name | Access | Address offset | Description | Reset value[1] | Table |
|---|---|---|---|---|---|
| CR | R/W | 0x000 | D/A Converter Register. This register contains the digital value to be converted to analog and a power control bit. | 0 | 290 |
| CTRL | R/W | 0x004 | DAC Control register. This register controls timer operation. | 0 | 291 |
| CNTVAL | R/W | 0x008 | DAC Counter Value register. This register contains the reload value for the DAC Interrupt timer. | 0 | 292 |

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

### 23.5.1 D/A Converter Register

This read/write register includes the digital value to be converted to analog, and a bit that trades off performance vs. power. Bits 5:0 are reserved for future, higher-resolution D/A converters.

**Table 290. D/A Converter Register (CR - address 0x4001 4000) bit description**

| Bit | Symbol | Value | Description | Reset Value |
|---|---|---|---|---|
| 5:0 | - | | Reserved. Read value is undefined, only zero should be written. | NA |
| 15:6 | VALUE | | After the selected settling time after this field is written with a new VALUE, the voltage on the DAC_OUT pin (with respect to $V_{SSA}$) is VALUE × (($V_{REFP}$)/1024). | 0 |
| 16 | BIAS | | Settling time<br>The settling times noted in the description of the BIAS bit are valid for a capacitance load on the DAC_OUT pin not exceeding 100 pF. A load impedance value greater than that value will cause settling time longer than the specified time. One or more graphs of load impedance vs. settling time will be included in the final data sheet. | 0 |
| | | 0 | The settling time of the DAC is 1 μs max, and the maximum current is 700 mA. This allows a maximum update rate of 1 MHz. | |
| | | 1 | The settling time of the DAC is 2.5 μs and the maximum current is 350 μA. This allows a maximum update rate of 400 kHz. | |
| 31:17 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

### 23.5.2 D/A Converter Control register

This read/write register enables the interrupt set and controls the interrupt timer.

**Table 291. D/A Control register (CTRL - address 0x4001 4004) bit description**

| Bit | Symbol | Value | Description | Reset Value |
|---|---|---|---|---|
| 0 | INT_CPU_REQ | | Interrupt request to CPU. This interrupt request is handled by the CPU. | 0 |
| | | 0 | Clear on any write to the DACR register. | |
| | | 1 | Set by hardware when the timer times out. | |

**Table 291. D/A Control register (CTRL - address 0x4001 4004) bit description**

| Bit | Symbol | Value | Description | Reset Value |
|---|---|---|---|---|
| 1 | DBLBUF_ENA | | Double buffering | 0 |
| | | 0 | Disable | |
| | | 1 | Enable. When this bit and the CNT_ENA bit are both set, the double-buffering feature in the DACR register will be enabled. Writes to the DACR register are written to a pre-buffer and then transferred to the DACR on the next time-out of the counter. | |
| 2 | CNT_ENA | | Time-out counter operation | 0 |
| | | 0 | Disable | |
| | | 1 | Enable | |
| 31:3 | - | | Reserved. Read value is undefined, only zero should be written. | NA |

### 23.5.3 D/A Converter Counter Value register

This read/write register contains the reload value for the Interrupt counter.

**Table 292. D/A Converter Counter Value register (CNTVAL - address 0x4001 4008) bit description**

| Bit | Symbol | Description | Reset Value |
|---|---|---|---|
| 15:0 | VALUE | 16-bit reload value for the DAC interrupt timer. | 0 |
| 31:16 | - | Reserved | - |

## 23.6 Operation

### 23.6.1 Interrupt counter

When the counter enable bit CNT_ENA in DACCTRL is set, a 16-bit counter will begin counting down, at the rate selected by PCLK, from the value programmed into the DACCNTVAL register. The counter is decremented. Each time the counter reaches zero, the counter will be reloaded by the value of DACCNTVAL and the Interrupt request bit INT_REQ will be set in hardware.

Note that the contents of the DACCTRL and DACCNTVAL registers are read and write accessible, but the timer itself is not accessible for either read or write.

### 23.6.2 Double buffering

Double-buffering is enabled only if both, the CNT_ENA and the DBLBUF_ENA bits are set in DACCTRL. In this case, any write to the DACR register will only load the pre-buffer, which shares its register address with the DACR register. The DACR itself will be loaded from the pre-buffer whenever the counter reaches zero and the interrupt request is set. At the same time the counter is reloaded with the COUNTVAL register value.

Reading the DACR register will only return the contents of the DACR register itself, not the contents of the pre-buffer register.

If either the CNT_ENA or the DBLBUF_ENA bits are 0, any writes to the DACR address will go directly to the DACR register.

## 24.1 How to read this chapter

The analog comparator is available on all LPC804 parts.

## 24.2 Features

- Selectable external inputs can be used as either the positive or negative input of the comparator.
- The Internal voltage reference (0.9 V bandgap reference) can be used as either the positive or negative input of the comparator.
- 32-stage voltage ladder can be used as either the positive or negative input of the comparator.
- Voltage ladder source selectable between the supply pin $V_{DD}$ or VDDCMP pin.
- Voltage ladder can be separately powered down when not required.
- Interrupt capability

## 24.3 Basic configuration

Configure the analog comparator using the following registers:

- In the SYSAHBCLKCTRL register, set bit 19 (Table 64) to enable the clock to the register interface.
- You can enable or disable the power to the analog comparator through the PDRUNCFG register (Table 84).
- Clear the analog comparator peripheral reset using the PRESETCTRL register (Table 66).
- The analog comparator interrupt is connected to interrupt #11 in the NVIC.
- Configure the analog comparator pin functions through the switch matrix. See Section 24.4.

### 24.3.1 Connect the comparator output to the ADC

The comparator output function (ACMP_O) can be used to start the ADC conversion, more generally, to create an ADC conversion event without assigning a pin through the switch matrix. To create an ADC event internally connected to the comparator output, select the comparator output as one of the ADC trigger inputs through the ADC trigger select register (see Table 276 “A/D Conversion Sequence A Control Register (SEQA_CTRL, address 0x4001 C008) bit description” and Table 277 “A/D Conversion Sequence B Control Register (SEQB_CTRL, address 0x4001 C00C) bit description”.

## 24.4 Pin description

The analog comparator reference voltage, the inputs, and the output are assigned to external pins through the switch matrix. You can assign the analog comparator output to any pin on the package that is not a supply or ground pin. The comparator inputs and the reference voltage are fixed-pin functions that must be enabled through the switch matrix and can only be assigned to special pins on the package.

See Section 8.3.1 “Connect an internal signal to a package pin” to assign the analog comparator output to any pin on the LPC804 package.

**Table 293. Analog comparator pin description**

| Function | Type | Pin | Description | SWM register | Reference |
|---|---|---|---|---|---|
| ACMP_I1 | I | PIO0_0 | Comparator input 1 | PINENABLE0 | Table 102 |
| ACMP_I2 | I | PIO0_1 | Comparator input 2 | PINENABLE0 | Table 102 |
| ACMP_I3 | I | PIO0_14 | Comparator input 3 | PINENABLE0 | Table 102 |
| ACMP_I4 | I | PIO0_16 | Comparator input 4 | PINENABLE0 | Table 102 |
| ACMP_I5 | I | PIO0_21 | Comparator input 5 | PINENABLE0 | |
| ACMP_O | O | Any | Comparator output | PINASSIGN5 | Table 96 |
| VDDCMP | I | PIO0_7 | External reference voltage source for 32-stage Voltage Ladder. | PINENABLE0 | Table 102 |

## 24.5 General description

The analog comparator can compare voltage levels on external pins and internal voltages.

The comparator has seven inputs multiplexed separately to its positive and negative inputs. The multiplexers are controlled by the comparator register CTL (see Figure 49 and Table 295).

Input 0 of the multiplexer is the programmable voltage ladder output.

Inputs 1 to 5 connect the external inputs ACMP_I[5:1].

Input 6 of the multiplexer connects the internal reference voltage input.

Input 7 of the multiplexer connects the DACOUT0.

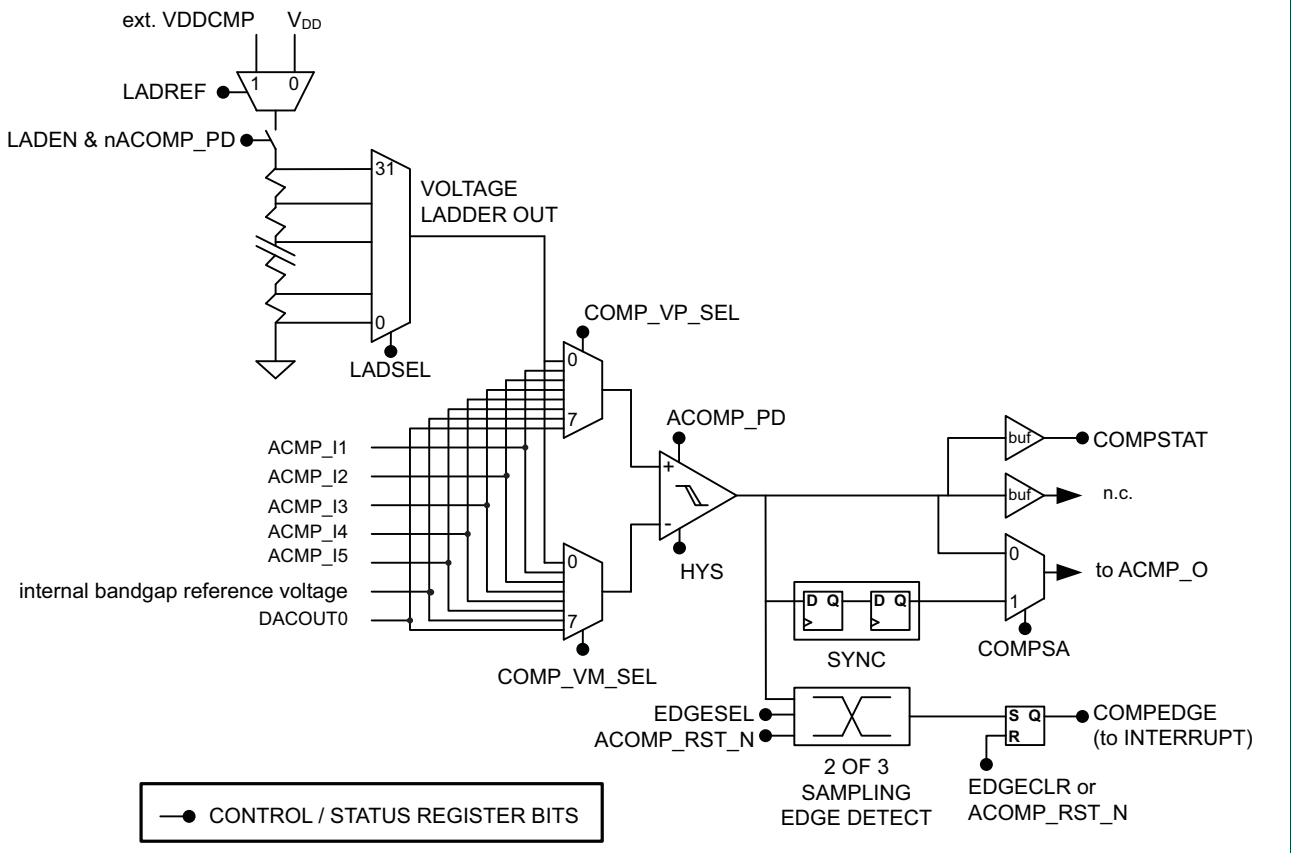



**Fig 49. Comparator block diagram**

### 24.5.1 Reference voltages

The voltage ladder can use two reference voltages, from the VDDCMP or the VDD pin. The voltage ladder selects one of 32 steps between the pin voltage and VSS inclusive. The voltage on VDDCMP should not exceed that on VDD.

### 24.5.2 Settling times

After the voltage ladder is powered on, it requires stabilization time until comparisons using it are accurate. Much shorter settling times apply after the LADSEL value is changed and when either or both voltage sources are changed. Software can deal with these factors by repeatedly reading the comparator output until a number of readings yield the same result.

### 24.5.3 Interrupts

The interrupt output comes from edge detection circuitry in this module. Rising edges, falling edges, or both edges can set the COMPEDGE bit and thus request an interrupt. COMPEDGE and the interrupt request are cleared when software writes a 1 to EDGECLR.

### 24.5.4 Comparator outputs

The comparator output (conditioned by COMPSA bit) can be routed to an external pin. When COMPSA is 0 and the comparator interrupt is disabled, the comparator can be used with the bus clock disabled (Table 64 “System clock control 0 register (SYSAHBCLKCTRL0, address 0x4004 8080) bit description”) to save power if the control registers don’t need to be written.

The status of the comparator output can be observed through the comparator status register bit.

## 24.6 Register description

**Table 294. Register overview: Analog comparator (base address 0x4002 4000)**

| Name | Access | Address offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| CTRL | R/W | 0x000 | Comparator control register | 0 | Table 295 |
| LAD | R/W | 0x004 | Voltage ladder register | 0 | Table 296 |

### 24.6.1 Comparator control register

This register enables the comparator, configures the interrupts, and controls the input multiplexers on both sides of the comparator. All bits not shown in Table 295 are reserved and should be written as 0.

**Table 295. Comparator control register (CTRL, address 0x4002 4000) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 2:0 | - | | Reserved. Write as 0. | 0 |
| 4:3 | EDGESEL | | This field controls which edges on the comparator output set the COMPEDGE bit (bit 23 below): | 0 |
| | | 0x0 | Falling edges | |
| | | 0x1 | Rising edges | |
| | | 0x2 | Both edges | |
| | | 0x3 | Both edges | |
| 5 | - | | Reserved. Write as 0. | 0 |
| 6 | COMPSA | | Comparator output control | 0 |
| | | 0 | Comparator output is used directly. | |
| | | 1 | Comparator output is synchronized to the bus clock for output to other modules. | |
| 7 | - | | Reserved. Write as 0. | 0 |

**Table 295. Comparator control register (CTRL, address 0x4002 4000) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 10:8 | COMP_VP_SEL | | Selects positive voltage input | 0 |
| | | 0x0 | Voltage ladder output | |
| | | 0x1 | ACMP_I1 | |
| | | 0x2 | ACMP_I2 | |
| | | 0x3 | ACMP_I3 | |
| | | 0x4 | ACMP_I4 | |
| | | 0x5 | ACMP_I5 | |
| | | 0x6 | Band gap. Internal reference voltage. | |
| | | 0x7 | DACOUT0 | |
| 13:11 | COMP_VM_SEL | | Selects negative voltage input | 0 |
| | | 0x0 | Voltage ladder output | |
| | | 0x1 | ACMP_I1 | |
| | | 0x2 | ACMP_I2 | |
| | | 0x3 | ACMP_I3 | |
| | | 0x4 | ACMP_I4 | |
| | | 0x5 | ACMP_I5 | |
| | | 0x6 | Band gap. Internal reference voltage. | |
| | | 0x7 | DACOUT0 | |
| 19:14 | - | | Reserved. Write as 0. | 0 |
| 20 | EDGECLR | | Interrupt clear bit. To clear the COMPEDGE bit and thus negate the interrupt request, toggle the EDGECLR bit by first writing a 1 and then a 0. | 0 |
| 21 | COMPSTAT | | Comparator status. This bit reflects the state of the comparator output. | 0 |
| 22 | - | | Reserved. Write as 0. | 0 |
| 23 | COMPEDGE | | Comparator edge-detect status. | 0 |
| 24 | INTENA | | Must be set to generate interrupts. | 0 |
| 26:25 | HYS | | Controls the hysteresis of the comparator. When the comparator is outputting a certain state, this is the difference between the selected signals, in the opposite direction from the state being output, that will switch the output. | 0 |
| | | 0x0 | None (the output will switch as the voltages cross) | |
| | | 0x1 | 5 mV | |
| | | 0x2 | 10 mV | |
| | | 0x3 | 20 mV | |
| 31:27 | - | | Reserved | - |

### 24.6.2 Voltage ladder register

This register enables and controls the voltage ladder. The fraction of the reference voltage produced by the ladder is programmable in steps of 1/31.

**Table 296. Voltage ladder register (LAD, address 0x4002 4004) bit description**

| Bit | Symbol | Value | Description | Reset value |
|---|---|---|---|---|
| 0 | LADEN | | Voltage ladder enable | 0 |
| 5:1 | LADSEL | | Voltage ladder value. The reference voltage Vref depends on the LADREF bit below.<br>00000 = $V_{SS}$<br>00001 = 1 x Vref/31<br>00010 = 2 x Vref/31<br>...<br>11111 = Vref | 0 |
| 6 | LADREF | | Selects the reference voltage Vref for the voltage ladder: | 0 |
| | | 0 | Supply pin VDD | |
| | | 1 | VDDCMP pin | |
| 31:7 | - | | Reserved. | 0 |

## 25.1 How to read this chapter

The PLU is available on all parts.

## 25.2 Features

- The Programmable Logic Unit is used to create small combinatorial and/or sequential logic networks including simple state machines.
- The PLU is comprised of an array of 26 inter-connectable, 5-input Look-up Table (LUT) elements, and four flip-flops.
- Eight primary outputs can be selected using a multiplexer from among all of the LUT outputs and the four flip-flops.
- An external clock to drive the four flip-flops must be applied to the PLU_CLKIN pin if a sequential network is implemented.
- Programmable logic can be used to drive on-chip inputs/triggers through external pin-to-pin connections.
- A tool suite is provided to facilitate programming of the PLU to implement the logic network described in a Verilog RTL design.

## 25.3 General Description

The PLU is comprised of 26 5-input LUT elements. Each LUT element contains a 32-bit truth table (look-up table) register and a 32:1 multiplexer. During operation, the five LUT inputs control the select lines of the multiplexer. This structure allows any desired logical combination of the five LUT inputs.

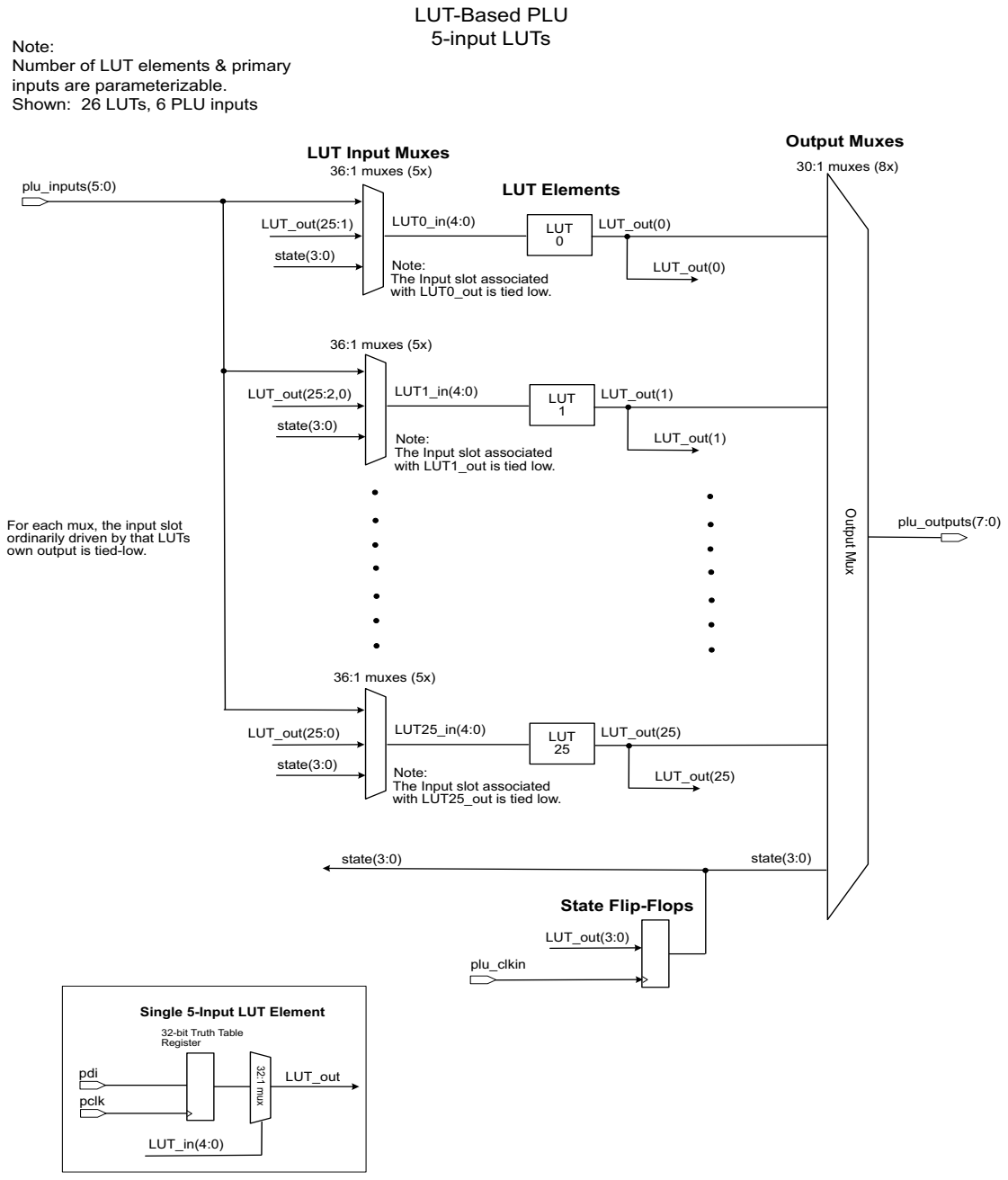
The five inputs to each LUT can be driven from a selection of sources comprised of primary inputs from pins, together with the outputs from all of the other LUTs and the outputs of the four “state” flip-flops. A set of multiplexers associated with each LUT is used to select the five inputs to that LUT. These multiplexers are controlled by registers, which are programmed during initialization. Connecting multiple LUT elements together permits construction of complex boolean expressions.

The outputs of up to four of the LUTs can be captured in one of the four “state” flip-flops, which can then be used as primary outputs and/or connected to the inputs of other LUTs. These four flip-flops, if used by the target logic network, are clocked by the external “plu_clkin” supplied by the user.

Note: The four “state” flip-flops are automatically cleared to ‘0’ by the internal chip reset signal. If a different initial state is required than all-zeros for these flip-flops, the user must force the primary inputs at start-up to some combination that will achieve the required state. That start-up combination must be maintained through at least one plu_clkin rising-edge.

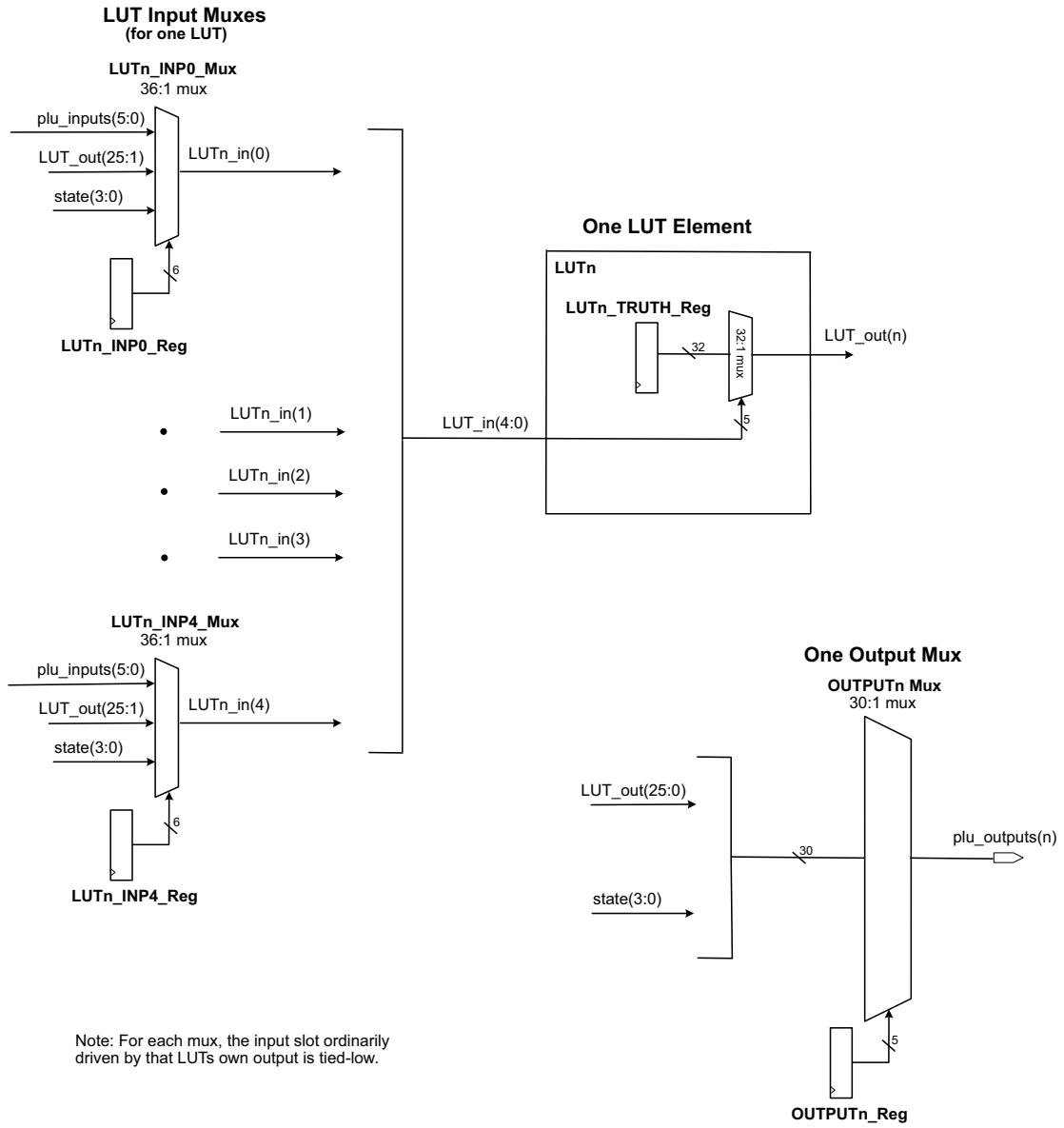
There are eight primary output pins. Any of the ‘n’ LUT outputs or the four flip-flops can be selected to construct the eight primary outputs.

**Remark:** In general, once the PLU module is configured, the PLU bus clock can be shut-off to conserve power. The only exception to this is when there is a need to read the Outputs Register while the PLU is operational. In that case, the PLU bus clock must be re-enabled prior to performing the read.




**Example configuration: 26 LUT elements; 6 primary inputs**

**Fig 50. PLU block diagram**




Example configuration

**Fig 51. PLU register/multiplexing detail**

### 25.3.1 Using the Programmable Logic Unit

Programming the PLU to implement a particular logic network involves writing to the various Truth Table registers to specify the logic functions to be performed by each of the LUT elements, programming the Input multiplexer registers to select the five inputs presented to each LUT, and programming the Output multiplexer register to select the eight primary outputs from the PLU module.

Programming of all of these registers is performed during initialization. Do not write to the registers during operation.

To facilitate programming of the PLU, a tool suite is provided. The tools report the values that must be written to all the registers in Table 298 “Register overview: PLU (base address 0x40028000)” to implement a logic network described in a Verilog RTL input file. See Section 25.3.2 for a complete description of the tool flow.

Programming the I/O multiplexing through the SWM is needed to connect the required number of PLU primary inputs and outputs to pins. See Section 8.5.11 “Pin fixed assign register 0”.

### 25.3.2 Description of Tool Flow

The PLU programming tool suite that is provided facilitates configuring the PLU to implement the desired logic network. The input to the tool is a Verilog RTL description of the functionality to be implemented.

The output of the tool suite provides the following:

1. Values to be programmed into all LUT INPUT MUX registers.
2. Values to be programmed into all LUT TRUTH registers
3. Values to be programmed into all OUTPUT MUX registers
4. Error response if the described network cannot be implemented (most likely due to an excessive amount of logic or use of more than four flip-flops)
5. AC Timing parameters for the implemented design.

## 25.4 Pin description

There are up to six primary inputs into the PLU module, one clock input, and eight primary outputs. All the inputs are connected directly to the package pins via chip-level I/O multiplexing. All these pins can be enabled by configuring the relevant SWM register.

A particular logic network may not require all of the available inputs or outputs. The user can specify which inputs and outputs to use, and which package pins those inputs and outputs will connect to as part of the overall top-level IO configuration.

If the logic network utilizes one or more of the four “state” flip-flops, an external clock must be applied to the PLU_CLKIN input. The package pin used for this function is specified using the chip’s top-level I/O multiplexing. All other PLU inputs must meet specified setup and hold times relative to this clock input. Output timing is also specified relative to this pin.

If the logic network is purely combinatorial, there is no need to provide an input clock to PLU_CLKIN.

**Table 297. PLU pins**

| Pin | Description |
|---|---|
| plu_inputs(5:0) | Primary inputs. All plu_inputs are available as input sources to all the LUT elements. |
| plu_clkin | Input clock to the four “state” flip-flops, if used. Not required for purely combinatorial networks. Input/Output timing specified relative to this clock. |
| plu_outputs(7:0) | Primary outputs. Selectable via mux from among all LUT element outputs and the four “state” flip-flops. |

## 25.5 Register description

There are 26 LUT elements in the PLU. For each LUT element there are five Input Multiplexer registers and one Truth-Table (“Look-up Table”) register.

The PLU has eight Output Multiplexer registers .

Each LUT input has 35 possible input sources to choose from. Since six bits are required to encode 35 choices, all of the Input Multiplexer registers are six bits wide. The 26 MSBs for each of these registers should be considered reserved.

All of the registers shown are clocked by the internal bus clock - not the plu_clkin pin. Only the four “state” flip-flops used as part of the target logic network use the externally applied plu_clkin.

**Table 298. Register overview: PLU (base address 0x40028000)**

| Name | Access | Address offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| LUTn_INPx_MUX | R/W | 0x000-0x010, 0x020-0x030, 0x040-0x050 ... 0x320-0x330 | Input select register for LUTn (0 to 25), Inputx (0 to 4)<br>As an example, register offsets for:<br>LUT0_INP0_MUX is 0x000<br>LUT0_INP1_MUX is 0x004<br>LUT0_INP2_MUX is 0x008<br>LUT0_INP3_MUX is 0x00C<br>LUT0_INP4_MUX is 0x010<br>LUT1_INP0_MUX is 0x020<br>LUT1_INP1_MUX is 0x024<br>LUT1_INP2_MUX is 0x028<br>LUT1_INP3_MUX is 0x02C<br>LUT1_INP4_MUX is 0x030 | All 1s | Section 25.5.1 |
| LUTn_TRUTH | R/W | 0x800, 0x804, 0x80C ... 0x8FC | Truth-Table (“Look-up Table”) programming for LUTn (0 to 25).<br>As an example, register offsets for:<br>LUT0_TRUTH is 0x800<br>LUT1_TRUTH is 0x804<br>LUT2_TRUTH is 0x808<br>LUT3_TRUTH is 0x80<br>LUT4_TRUTH is 0x810 | 0x0 | Section 25.5.2 |
| - | - | - | Reserved | - | - |

**Table 298. Register overview: PLU (base address 0x40028000)**

| Name | Access | Address offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| OUTPUT0_MUX | R/W | 0xC00 | Select register for PLU Output0 | 0x1F | Section 25.5.3 |
| OUTPUT1_MUX | R/W | 0xC04 | Select register for PLU Output1 | 0x1F | Section 25.5.3 |
| OUTPUT2_MUX | R/W | 0xC08 | Select register for PLU Output2 | 0x1F | Section 25.5.3 |
| OUTPUT3_MUX | R/W | 0xC0C | Select register for PLU Output3 | 0x1F | Section 25.5.3 |
| OUTPUT4_MUX | R/W | 0xC10 | Select register for PLU Output4 | 0x1F | Section 25.5.3 |
| OUTPUT5_MUX | R/W | 0xC14 | Select register for PLU Output5 | 0x1F | Section 25.5.3 |
| OUTPUT6_MUX | R/W | 0xC18 | Select register for PLU Output6 | 0x1F | Section 25.5.3 |
| OUTPUT7_MUX | R/W | 0xC1C | Select register for PLU Output7 | 0x1F | Section 25.5.3 |
| OUTPUTS | RO | 0x900 | PLU Outputs Register (Read-only) | 0x0 | Section 25.5.3 |

[1] For the LUTn_INP and OUTPUT mux registers, all ones in the implemented bits (lsb's) means none of the input options is selected. In this case the associated multiplexor output is a fixed logic '0'. Typically these registers must be programmed to some valid value.

### 25.5.1 PLU LUT input multiplexer registers

Each LUT has five inputs and a selection of input sources that can be connected to each of those inputs. These registers control the multiplexers to select which input sources to connect to each LUT input.

**Remark:** The values that must be programmed into each of these registers is provided by the tool suite.

**Table 299. PLU LUT Input Mux registers (LUTn_INPx_MUX) address 0x40028000, 0x000-0x010, 0x020-0x030, 0x040-0x050, ..... 0x320-0x330**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 5:0 | LUTn_INPx | Selects the input source to be connected to LUTn Input x<br>For each LUT input the available input sources are, in sequence:<br>1. The PLU primary inputs, beginning with plu_inputs(0)<br>2. The outputs from all of the other LUT elements (aside from LUTn itself) in order from the lowest to highest-numbered remaining LUTs. (For each LUT, the slot associated with the output from LUTn itself is tied low).<br>3. The four "state" Flip-Flops, beginning with state(0).<br><br>0x0 programmed in this field will select plu_inputs(0) as the source of this LUT input. Each higher binary value will select one of the other sources in the above list in the order shown.<br>The reset value of "all ones" causes a fixed '0' to be passed to this LUT input.<br>**Remark:** Note that the total number of bits "m" in this field is variable based on the total number of available input sources which, in turn is dependent on the number of primary PLU inputs and the number of LUTs | All 1s |
| 31:6 | Reserved | Software should not write ones to reserved bits. | 0x0 |

### 25.5.2 PLU LUT truth table registers

Each LUT element contains one 32-bit Truth Table (“Look-up Table”) Register which specifies whether the LUT will output a ‘0’ or a ‘1’ for each combination of the 5 LUT inputs. In other words, these registers specify the complex Boolean expression each individual LUT is to perform

**Remark:** The values that must be programmed into each of these registers is provided by the tool suite.

**Table 300. PLU LUT truth table registers (LUTn_TRUTH) address0x40028000, 0x800, 0x804, 0x80C ..... 0x8FC**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 31:0 | LUTn_TRUTH | Specifies the Truth Table contents for LUTn | 0 |

### 25.5.3 PLU output multiplexer registers

The eight PLU module outputs are specified using these eight registers.

The available choices to comprise the eight PLU outputs are all of the individual LUT element outputs and the four “state” flip-flop outputs.

**Remark:** The values that must be programmed into each of these registers is provided by the tool suite.

**Table 301. PLU Output Mux Registers (PLU_OUTPUTn_MUX, address 0x40028000, 0xC00-0xC1C)**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 4:0 | OUTPUTn | Selects the source to be connected to PLU Output n<br>For each LUT input the available input sources are, in sequence:<br>1. The outputs from all of the available LUT elements, beginning with LUT0<br>2. The four “state” Flip-Flops, beginning with state(0).<br><br>0x0 programmed in this field will select LUT0 as the source of this output. Each higher binary value will select one of the other sources in the above list in the order shown.<br>**Remark:** Note that the total number of bits “m” in this field is variable based on the total number of LUTs provided. | 0x0 |
| 31:5 | Reserved | Software should not write ones to reserved bits. | 0x0 |

**Remark:** Note that the eight PLU outputs can only be routed to GPIO pins via SWM. There is no provision to internally connect outputs from the programmable logic to on-chip resources such as interrupts, ADC triggers, SCT inputs, and Timer-Capture inputs. Driving these inputs from the programmable logic can be accomplished by externally connecting a PLU output pin to the desired GPIO input pin.

### 25.5.4 PLU outputs register

The eight selected PLU outputs can be read via this read-only register address.

**Remark:** There is no guarantee that a read of this register will not capture transitional data because of the asynchronous nature of the PLU. It is strongly recommended to read this data multiple times until a consistent result is returned unless it is known that the PLU inputs will be stable during the read operation.

**Table 302. PLU Outputs Register (PLU_OUTPUTS address 0x40028000, 0x900)**

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 7:0 | OUTPUT | Provides the current state of the 8 designated PLU Outputs.<br>(All 8 bits are available to be read regardless of whether or not they are routed to package pins) | 0x0 |
| 31:8 | Reserved | Software should not write ones to reserved bits. | 0x0 |

## 26.1 How to read this chapter

The CRC engine is available on all LPC804 parts.

## 26.2 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.
- 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)

## 26.3 Basic configuration

Enable the clock to the CRC engine in the SYSAHBCLKCTRL register (Table 64, bit 13).

## 26.4 Pin description

The CRC engine has no configurable pins.

## 26.5 General description

The Cyclic Redundancy Check (CRC) generator with programmable polynomial settings supports several CRC standards commonly used.

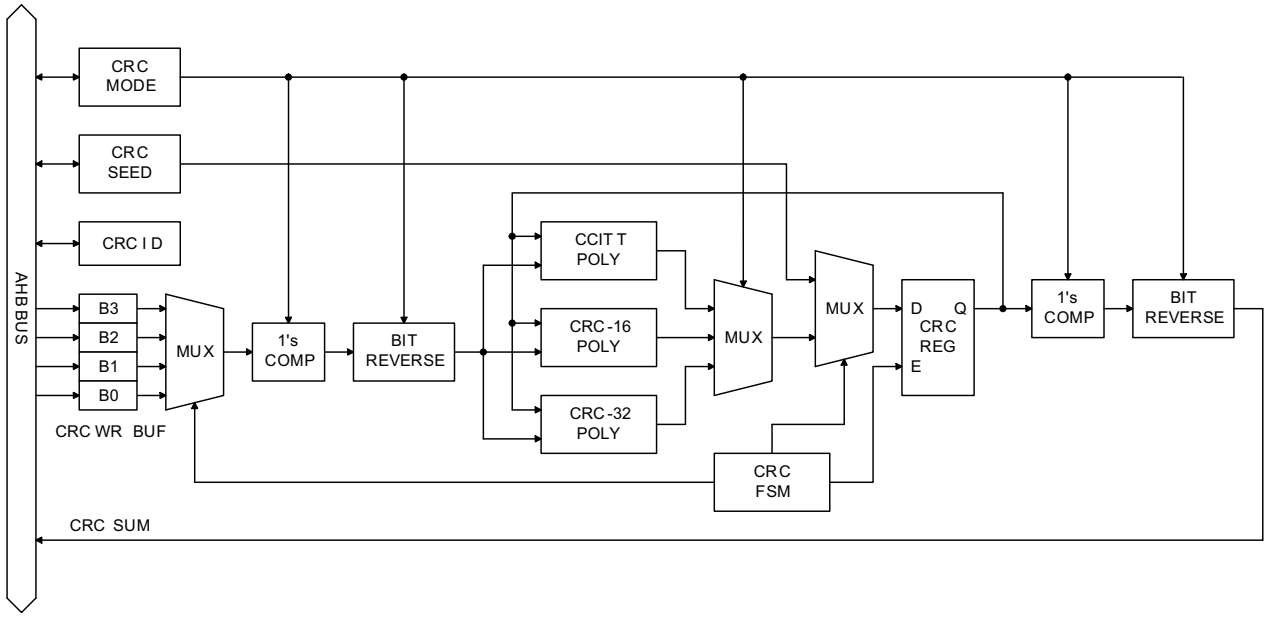



**Fig 52. CRC block diagram**

## 26.6 Register description

Table 303. Register overview: CRC engine (base address 0x5000 0000)

| Name | Access | Address offset | Description | Reset value | Reference |
|---|---|---|---|---|---|
| MODE | R/W | 0x000 | CRC mode register | 0x0000 0000 | Table 304 |
| SEED | R/W | 0x004 | CRC seed register | 0x0000 FFFF | Table 305 |
| SUM | RO | 0x008 | CRC checksum register | 0x0000 FFFF | Table 306 |
| WR_DATA | WO | 0x008 | CRC data register | - | Table 307 |

### 26.6.1 CRC mode register

Table 304. CRC mode register (MODE, address 0x5000 0000) bit description

| Bit | Symbol | Description | Reset value |
|---|---|---|---|
| 1:0 | CRC_POLY | CRC polynom:<br>1X= CRC-32 polynomial<br>01= CRC-16 polynomial<br>00= CRC-CCITT polynomial | 00 |
| 2 | BIT_RVS_WR | Data bit order:<br>1= Bit order reverse for CRC_WR_DATA (per byte)<br>0= No bit order reverse for CRC_WR_DATA (per byte) | 0 |
| 3 | CMPL_WR | Data complement:<br>1= 1’s complement for CRC_WR_DATA<br>0= No 1’s complement for CRC_WR_DATA | 0 |
| 4 | BIT_RVS_SUM | CRC sum bit order:<br>1= Bit order reverse for CRC_SUM<br>0= No bit order reverse for CRC_SUM | 0 |
| 5 | CMPL_SUM | CRC sum complement:<br>1= 1’s complement for CRC_SUM<br>0=No 1’s complement for CRC_SUM | 0 |
| 31:6 | Reserved | Always 0 when read | 0x0000000 |

### 26.6.2 CRC seed register

**Table 305. CRC seed register (SEED, address 0x5000 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.<br>**Remark:** A write access to this register will overrule the CRC calculation in progresses. | 0x0000 FFFF |

### 26.6.3 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 306. CRC checksum register (SUM, address 0x5000 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 |

### 26.6.4 CRC data register

This register is a Write-only register containing the data block for which the CRC sum will be calculated.

**Table 307. CRC data register (WR_DATA, address 0x5000 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. | - |

## 26.7 Functional description

The following sections describe the register settings for each supported CRC standard:

### 26.7.1 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

### 26.7.2 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

### 26.7.3 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

## 27.1 How to read this chapter

The ROM-based 32-bit integer division routines are available on all parts.

## 27.2 Features

- Performance-optimized signed/unsigned integer division.
- Performance-optimized signed/unsigned integer division with remainder.
- ROM-based routines to reduce code size.
- Support for integers up to 32 bit.
- ROM calls can easily be added to EABI-compliant functions to overload “/” and “%” operators in C.

## 27.3 General description

The API calls to the ROM are performed by executing functions which are pointed by a pointer within the ROM Driver Table. Figure 53 shows the pointer structure used to call the Integer divider API.

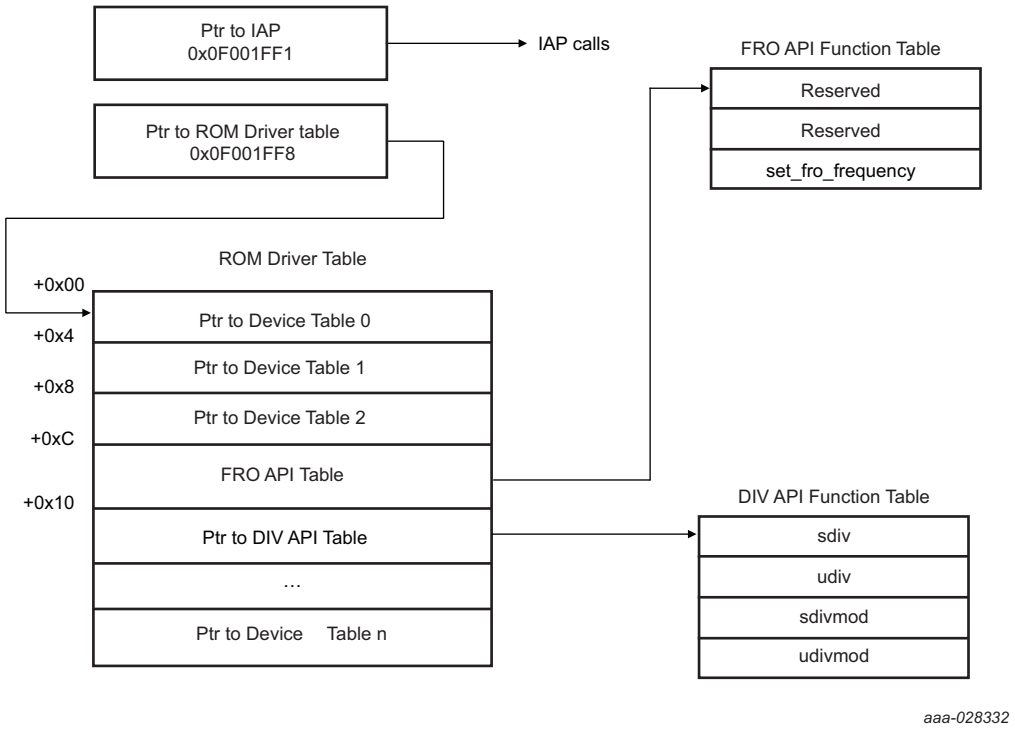



**Fig 53. ROM pointer structure**

## 27.4 API description

The integer division routines perform arithmetic integer division operations and can be called in the application code through simple API calls.

**Table 308. Divide API calls**

| API call | Description | Reference |
|---|---|---|
| `int(*sdiv)(int numerator, int denominator);` | Signed integer division | Table 309 |
| `unsigned(*udiv) (int numerator, int denominator);` | Unsigned integer division | Table 310 |
| `sdiv_t (*sdivmod)(int numerator, int denominator);` | Signed integer division with remainder | Table 311 |
| `udiv_t (*udivmod)(unsigned numerator, unsigned denominator);` | Unsigned integer division with remainder | Table 312 |

The following function prototypes are used:

```
typedef struct {
     int quot;       /*!< Quotient */
     int rem;        /*!< Remainder */
} IDIV_RETURN_T;

typedef struct {
     unsigned quot;/*!< Quotient */
     unsigned rem;/*!< Reminder */
} UIDIV_RETURN_T;

typedef struct {
     int (*sidiv)(int numerator, int denominator); /*!< Signed integer division */
     unsigned (*uidiv)(unsigned numerator, unsigned denominator);/*!< Unsigned
     integer division */
     IDIV_RETURN_T (*sidivmod)(int numerator, int denominator); /*!< Signed integer
     division with remainder */
     UIDIV_RETURN_T (*uidivmod)(unsigned numerator, unsigned denominator);/*!<
     Unsigned integer division
          with remainder */
} ROM_DIV_API_T;
ROM_DIV_API_T const *pROMDiv = LPC_ROM_API->divApiBase;
```

The ROM API table shown in Section 3.4.2 “ROM-based APIs” must be included in the code.

### 27.4.1 DIV signed integer division

**Table 309. sidiv**

| Routine | sidiv |
|---|---|
| Prototype | `int(*sidiv)(int32_t numerator, int32_t denominator);` |
| Input parameter | numerator: Numerator signed integer. denominator: Denominator signed integer. |
| Return | Signed division result without remainder. |
| Description | Signed integer division |

### 27.4.2 DIV unsigned integer division

**Table 310. uidiv**

| Routine | uidiv |
|---|---|
| Prototype | `int(*uidiv)(int32_t numerator, int32_t denominator);` |
| Input parameter | numerator: Numerator signed integer. denominator: Denominator signed integer. |
| Return | Unsigned division result without remainder. |
| Description | Unsigned integer division |

### 27.4.3 DIV signed integer division with remainder

**Table 311. sidivmod**

| Routine | sidivmod |
|---|---|
| Prototype | `IDIV_RETURN_T (*sidivmod) (int32_t numerator, int32_t denominator);` |
| Input parameter | numerator: Numerator signed integer. denominator: Denominator signed integer. |
| Return | Signed division result remainder. |
| Description | Signed integer division with remainder |

### 27.4.4 DIV unsigned integer division with remainder

**Table 312. uidivmod**

| Routine | uidivmod |
|---|---|
| Prototype | `UIDIV_RETURN_T(*uidiv)(uint32_t numerator, uint32_t denominator);` |
| Input parameter | numerator: Numerator unsigned integer. denominator: Denominator unsigned integer. |
| Return | Unsigned division result with remainder. |
| Description | Unsigned integer division |

## 27.5 Functional description

### 27.5.1 Signed division

The example C-code listing below shows how to perform a signed integer division via the ROM API.

```
/* Divide (-99) by (+6) */
int32_t result;
result = pROMDiv->sidiv(-99, 6);
/* result now contains (-16) */
```

### 27.5.2 Unsigned division with remainder

The example C-code listing below shows how to perform an unsigned integer division with remainder via the ROM API.

```
/* Modulus Divide (+99) by (+4) */
uidiv_return result;
result = pROMDiv-> uidivmod (+99, 4);
/* result.div contains (+24) */
/* result.mod contains (+3) */
```

## 28.1 How to read this chapter

The debug functionality is identical for all LPC804 parts.

## 28.2 Features

- Supports Arm Serial Wire Debug mode.
- Direct debug access to all memories, registers, and peripherals.
- No target resources are required for the debugging session.
- Four breakpoints.
- Two data watchpoints that can also be used as triggers.
- Supports JTAG boundary scan.

## 28.3 General description

Debug functions are integrated into the Arm Cortex-M0+. Serial wire debug functions are supported. The Arm Cortex-M0+ is configured to support up to four breakpoints and two watchpoints.

Support for boundary scan is available.

## 28.4 Pin description

The SWD functions are assigned to pins through the switch matrix. The SWD functions are fixed-pin functions that are enabled through the switch matrix and can only be assigned to special pins on the package. The SWD functions are enabled by default.

See Section 8.3.2 to enable the analog comparator inputs and the reference voltage input.

**Table 313. SWD pin description**

| Function | Type | Pin | Description | SWM register | Reference |
|---|---|---|---|---|---|
| SWCLK | I/O | SWCLK/PIO0_3/ TCLK | 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. | PINENABLE0 | Table 102 |
| SWDIO | I/O | SWDIO/PIO0_2/ TMS | **Serial wire debug data input/output.** The SWDIO pin is used by an external debug tool to communicate with and control the LPC804. This pin is pulled up internally. | PINENABLE0 | Table 102 |

The boundary scan mode and the pins needed are selected by hardware (see Section 28.5.3). There is no access to the boundary scan pins through the switch matrix.

**Table 314. JTAG boundary scan pin description**

| Function | Pin name | Type | Description |
|---|---|---|---|
| TCK | SWCLK/PIO0_3/ TCK | I | **JTAG Test Clock.** This pin is the clock for JTAG boundary scan when the RESET pin is LOW. |
| TMS | SWDIO/PIO0_2/ TMS | I | 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 RESET pin is LOW. |
| TDI | PIO0_1/ACMP_I2/ CLKIN/TDI | I | 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 RESET pin is LOW. |
| TDO | PIO0_0/ACMP_I1/ TDO | O | 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 RESET pin is LOW. |
| TRST | PIO0_4/ WAKEUP/TRST | I | JTAG **Test Reset.** The 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 RESET pin is LOW. |

## 28.5 Functional description

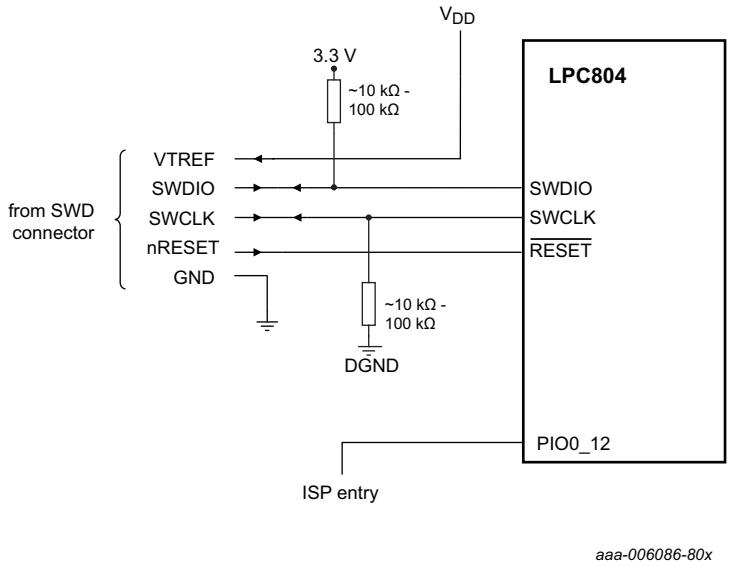
### 28.5.1 Debug limitations

It is recommended not to use the debug mode during deep-sleep or power-down mode.

During a debugging session, the System Tick Timer is automatically stopped whenever the CPU is stopped. Other peripherals are not affected.

### 28.5.2 Debug connections for SWD

For debugging purposes, it is useful to provide access to the ISP entry pin PIO0_12. This pin can be used to recover the part from configurations which would disable the SWD port such as re-configuration of SWD pins, entry into deep power-down mode out of reset, etc. This pin can be used for other functions such as GPIO, but it should not be held LOW on power-up or reset.




The VTREF pin on the SWD connector enables the debug connector to match the target voltage.

**Fig 54. Connecting the SWD pins to a standard SWD connector**

### 28.5.3 Boundary scan

The $\overline{\text{RESET}}$ pin selects between the JTAG boundary scan ($\overline{\text{RESET}}$ = LOW) and the Arm SWD debug ($\overline{\text{RESET}}$ = HIGH). The Arm SWD debug port is disabled while the part is in reset.

To perform boundary scan testing, follow these steps:

1. Erase any user code residing in flash.
2. Power up the part with the $\overline{\text{RESET}}$ pin pulled HIGH externally.
3. Wait for at least 1 ms.
4. Pull the $\overline{\text{RESET}}$ pin LOW externally.
5. Perform boundary scan operations.
6. Once the boundary scan operations are completed, assert the TRST pin to enable the SWD debug mode and release the $\overline{\text{RESET}}$ pin (pull HIGH).

**Remark:** The JTAG interface cannot be used for debug purposes.

**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 $\overline{\text{RESET}}$ = HIGH places the test TAP in Run-Test Idle mode.

## 29.1 Abbreviations

**Table 315. Abbreviations**

| Acronym | Description |
|---|---|
| A/D | Analog-to-Digital |
| ADC | Analog-to-Digital Converter |
| AHB | Advanced High-performance Bus |
| APB | Advanced Peripheral Bus |
| BOD | BrownOut Detection |
| GPIO | General Purpose Input/Output |
| JTAG | Joint Test Action Group |
| PLL | Phase-Locked Loop |
| RC | Resistor-Capacitor |
| SPI | Serial Peripheral Interface |
| SSI | Serial Synchronous Interface |
| SSP | Synchronous Serial Port |
| TAP | Test Access Port |
| UART | Universal Asynchronous Receiver/Transmitter |
| USART | Universal Synchronous Asynchronous Receiver/Transmitter |

## 29.2 References

[1] **LPC804 —** LPC804 Data sheet

[2] **ES_LPC804 —** LPC804 Errata sheet

[3] **DDI0484B_cortex_m0p_r0p0_trm —** Arm Cortex-M0+ Technical Reference Manual

[4] **DDI0486A —** Arm technical reference manual

[5] **AN11538 —** AN11538 application note and code bundle (SCT cookbook)

[6] ARMv6-M Architecture Reference Manual

# 29.3 Legal information

## 29.3.1 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.

## 29.3.2 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. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

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 and its suppliers accept 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.

## 29.3.3 Trademarks

Notice: All referenced brands, product names, service names and trademarks are the property of their respective owners.

**$I^2C$-bus** — logo is a trademark of NXP B.V.

## 29.4 Tables

## 29.5 Figures

## 29.6 Contents

### Chapter 1: LPC804 Introductory information



### Chapter 2: LPC804 memory mapping



### Chapter 3: LPC804 Boot Process



### Chapter 4: LPC804 ISP and IAP



### Chapter 5: LPC804 Nested Vectored Interrupt Controller (NVIC)

## Chapter 6: LPC804 System configuration (SYSCON)



## Chapter 7: LPC804 FRO API ROM routine



## Chapter 8: LPC804 Switch matrix (SWM)



## Chapter 9: LPC804 I/O configuration (IOCON)

## Chapter 10: LPC804 General Purpose I/O (GPIO)



## Chapter 11: LPC804 Pin interrupts/pattern match engine

## Chapter 12: LPC804 Reduced power modes and power management



## Chapter 13: LPC804 USART0/1



## Chapter 14: LPC804 SPI

## Chapter 15: LPC804 I2C



## Chapter 16: LPC804 Standard counter/timer (CTIMER)



## Chapter 17: LPC804 Windowed Watchdog Timer (WWDT)

## Chapter 18: LPC804 Self-wake-up timer (WKT)



## Chapter 19: LPC804 Multi-Rate Timer (MRT)



## Chapter 20: LPC804 System tick timer (SysTick)



## Chapter 21: LPC804 Capacitive Touch



## Chapter 22: LPC804 12-bit Analog-to-Digital Converter (ADC)

## Chapter 23: LPC804 Digital-to-Analog Converter (DAC)



## Chapter 24: LPC804 Analog comparator



## Chapter 25: Programmable Logic Unit (PLU)



## Chapter 26: LPC804 CRC engine

## Chapter 27: LPC804 ROM API integer divide routines



## Chapter 28: LPC804 Serial Wire Debug (SWD)



## Chapter 29: Supplementary information





For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

**Date of release: 12 July 2018**
**Document identifier: UM11065**