UM10800

LPC82x User manual

Rev. 1 — 18 September 2014 User manual

Document information

Info	Content
Keywords	LPC82x, LPC824M201JHI33, LPC822M101JHI33, LPC824M201JDH20, LPC822M101JDH20, LPC82x UM, LPC82x user manual, LPC820
Abstract	LPC82x User manual

Revision history

Rev	Date	Description
1	20140918	Initial revision. LPC82x User manual.

Contact information

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

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

1.1 Introduction

The LPC82x are an ARM Cortex-M0+ based, low-cost 32-bit MCU family operating at CPU frequencies of up to 30 MHz. The LPC82x support up to 32 KB of flash memory and 8 KB of SRAM.

The peripheral complement of the LPC82x includes a CRC engine, four I^2C-bus interfaces, up to three USARTs, up to two SPI interfaces, one multi-rate timer, self-wake-up timer, and state-configurable timer with PWM function (SCTimer/PWM), a DMA, one 12-bit ADC and one analog comparator, function-configurable I/O ports through a switch matrix, an input pattern match engine, and up to 29 general-purpose I/O pins.

Remark: For additional documentation, see Section 35.2 “References”.

1.2 Features

- System:
  - ARM Cortex-M0+ processor (revision r0p1), running at frequencies of up to 30 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.
  - Micro Trace Buffer (MTB)
- Memory:
  - Up to 32 KB on-chip flash programming memory with 64 Byte page write and erase. Code Read Protection (CRP) supported.
  - 8 KB SRAM.
- ROM API support:
  - bootloader.
  - On-chip ROM APIs for ADC, SPI, I2C, USART, power configuration (power profiles) and integer divide.
  - Flash In-Application Programming (IAP) and In-System Programming (ISP).
- Digital peripherals:
  - High-speed GPIO interface connected to the ARM Cortex-M0+ IO bus with up to 32 General-Purpose I/O (GPIO) pins with configurable pull-up/pull-down resistors, programmable open-drain mode, input inverter, and glitch filter. GPIO direction control supports independent set/clear/toggle of individual bits.
  - High-current source output driver (20 mA) on four pins.
  - High-current sink driver (20 mA) on two true open-drain 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.
- DMA with 18 channels and 9 trigger inputs.

- Timers:
  - State Configurable Timer (SCTimer/PWM) with input and output functions (including capture and match) for timing and PWM applications.
  - 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 the IRC, a low-power, low-frequency internal oscillator, or an external clock input in the always-on power domain.
  - 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 1.2 Msamples/s. The ADC supports two independent conversion sequences.
  - Comparator with four input pins and external or internal reference voltage.
- Serial peripherals:
  - Three USART interfaces with pin functions assigned through the switch matrix and one common fractional baud rate generator.
  - Two SPI controllers with pin functions assigned through the switch matrix.
  - Four I^2C-bus interfaces. One I2C supports Fast-mode plus with 1 Mbit/s data rates on two true open-drain pins and listen mode. Three I2Cs support data rates up to 400 kbit/s on standard digital pins.
- Clock generation:
  - 12 MHz internal RC oscillator trimmed to 1.5 % accuracy that can optionally be used as a system clock.
  - Crystal oscillator with an operating range of 1 MHz to 25 MHz.
  - Programmable watchdog oscillator with a frequency range of 9.4 kHz to 2.3 MHz.
  - PLL allows CPU operation up to the maximum CPU rate without the need for a high-frequency crystal. May be run from the system oscillator, the external clock input, or the internal RC oscillator.
  - Clock output function with divider that can reflect all internal clock sources.
- Power control:
  - Integrated PMU (Power Management Unit) to minimize power consumption.
  - Reduced power modes: Sleep mode, Deep-sleep mode, Power-down mode, and Deep power-down mode.
  - Wake-up from Deep-sleep and Power-down modes on activity on USART, SPI, and I2C peripherals.
  - Timer-controlled self-wake-up from Deep power-down mode.
  - Power-On Reset (POR).

- Brownout detect (BOD).

- Unique device serial number for identification.
- Single power supply (1.8 V to 3.6 V).
- Operating temperature range -40 °C to +105 °C.
- Available in a HVQFN33 (5x5) package.

1.3 Ordering options

Table 1. Ordering information

Type number	Package		
	Name	Description	Version
LPC824M201JHI33	HVQFN33	HVQFN: plastic thermal enhanced very thin quad flat package; no leads; 33 terminals; body 5 × 5 × 0.85 mm	n/a
LPC822M101JHI33	HVQFN33	HVQFN: plastic thermal enhanced very thin quad flat package; no leads; 33 terminals; body 5 × 5 × 0.85 mm	n/a
LPC824M201JDH20	TSSOP20	plastic thin shrink small outline package; 20 leads; body width 4.4 mm	SOT360-1
LPC822M101JDH20	TSSOP20	plastic thin shrink small outline package; 20 leads; body width 4.4 mm	SOT360-1

Table 2. Ordering options

Type number	Flash/ KB	SRAM/ KB	USART	I^2C	SPI	ADC channels	Comparator	GPIO	Package
LPC824M201JHI33	32	8	3	4	2	12	Y	29	HVQFN33
LPC822M101JHI33	16	4	3	4	2	12	Y	29	HVQFN33
LPC824M201JDH20	32	8	3	4	2	5	Y	16	TSSOP20
LPC822M101JDH20	16	4	3	4	2	5	y	16	TSSOP20

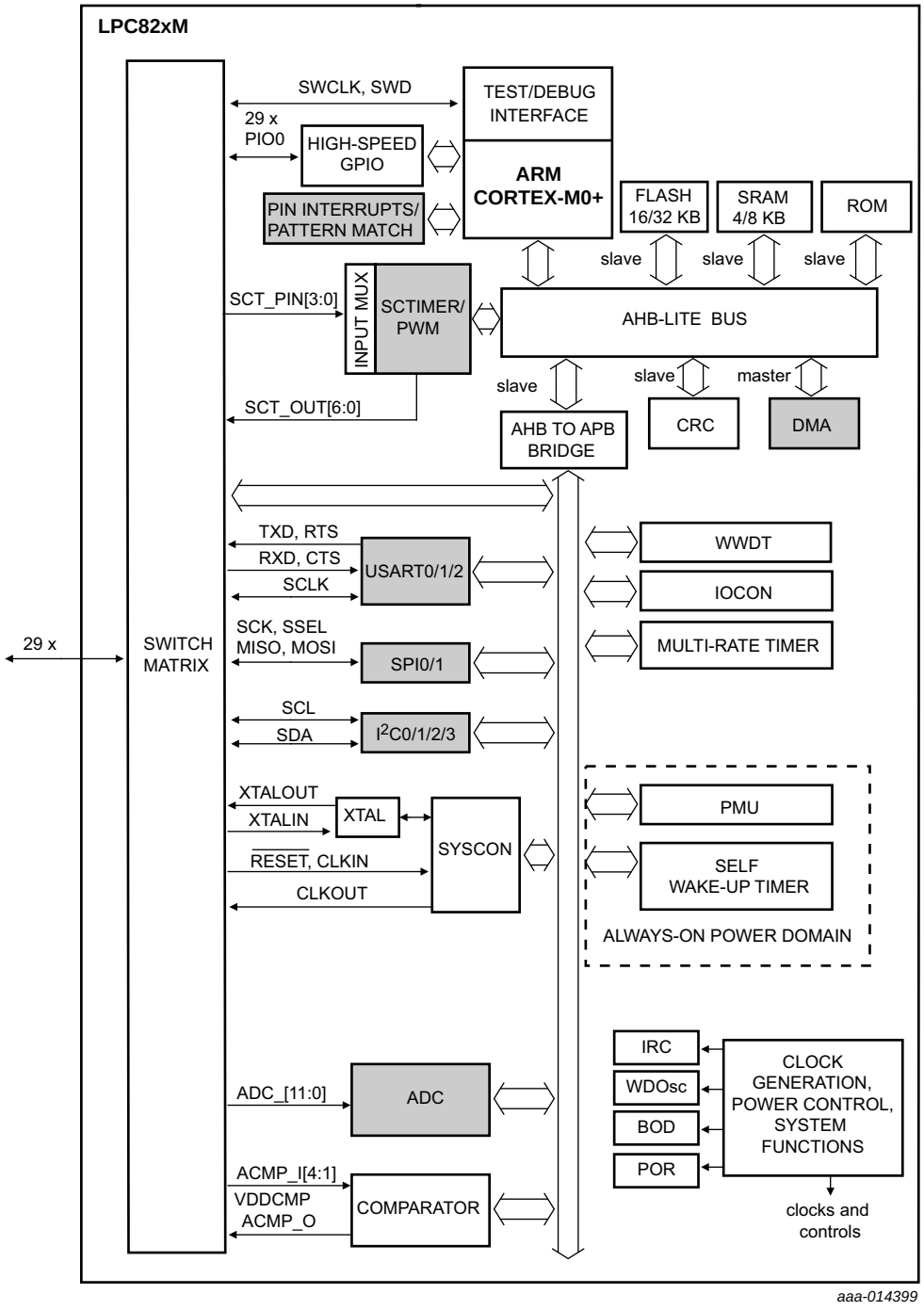
1.4 General description

1.4.1 ARM Cortex-M0+ core configuration

The ARM Cortex-M0+ core runs at an operating frequency of up to 30 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



Grey-shaded blocks show peripherals that can provide hardware triggers for DMA transfers or have DMA request lines.

Fig 1. LPC82x block diagram

2.1 How to read this chapter

The memory mapping is identical for all LPC82x parts. Different LPC82x parts support different flash and SRAM memory sizes.

2.2 General description

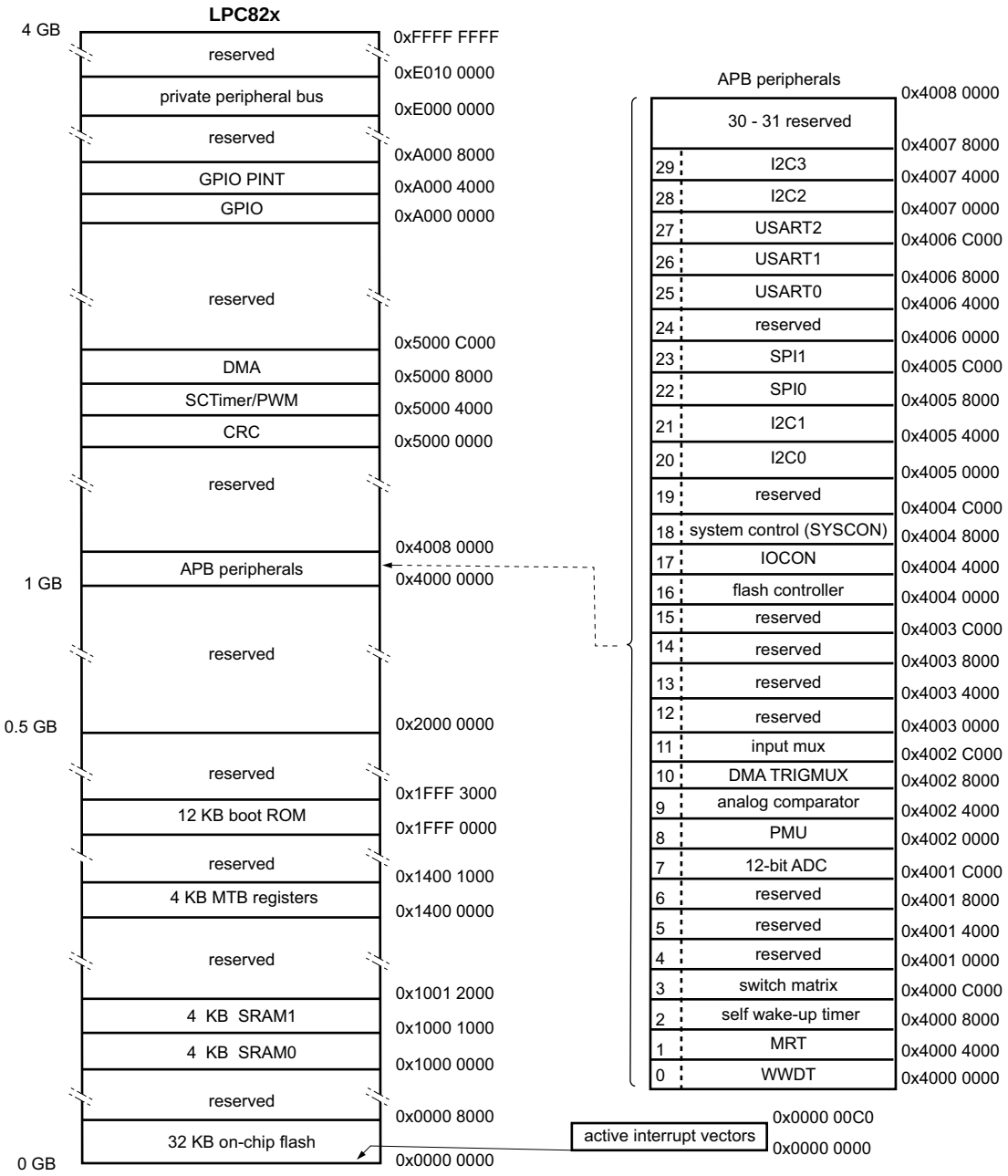
The LPC82x incorporates several distinct memory regions. Figure 2 shows the overall map of the entire address space from the user program viewpoint following reset.

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



The private peripheral bus includes the ARM Cortex-M0+ peripherals such as the NVIC, SysTick, and the core control registers.

Fig 2. LPC82x Memory mapping

2.2.2 Micro Trace Buffer (MTB)

The LPC82x supports the ARM Cortex-M0+ Micro Trace Buffer. See Section 31.5.4.

3.1 How to read this chapter

The bootloader is identical for all parts.

3.2 Features

- 12 KB on-chip boot ROM
- Contains the bootloader with In-System Programming (ISP) facility and the following APIs:
  - In-Application Programming (IAP) of flash memory
  - Power profiles for optimizing power consumption and system performance
  - USART driver
  - ADC driver
  - SPI driver
  - I2C driver
  - Integer divide routines

3.3 Basic configuration

The clock to the ROM is enabled by default. No configuration is required to use the ROM APIs.

3.4 Pin description

When the ISP entry pin is pulled LOW on reset, the part enters ISP mode and the ISP command handler starts up. In ISP mode, pin PIO0_0 is connected to function U0_RXD and pin PIO0_4 is connected to function U0_TXD on the USART0 block.

Table 3. Pin location in ISP mode

ISP entry pin	USART RXD	USART TXD
PIO0_12	PIO0_0	PIO0_4

3.5 General description

3.5.1 Bootloader

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

The bootloader code is executed every time the part is powered on or reset. The bootloader can execute the ISP command handler or the user application code. A LOW level after reset at the ISP entry pin is considered as an external hardware request to start the ISP command handler via USART.

For details on the boot process, see Section 3.6.2 “Boot process”.

Remark: SRAM location 0x1000 0000 to 0x1000 0050 is not used by the bootloader and the memory content in this area is retained during reset. SRAM memory is not retained when the part powers down or enters Deep power-down mode.

Assuming that power supply pins are at their nominal levels when the rising edge on $\overline{\text{RESET}}$ pin is generated, it may take up to 3 ms before the ISP entry pin is sampled and the decision whether to continue with user code or ISP handler is made. The bootloader performs the following steps (see Figure 4):

1. If the watchdog overflow flag is set, the bootloader checks whether a valid user code is present. If the watchdog overflow flag is not set, the ISP entry pin is checked.
2. If there is no request for the ISP command handler execution (ISP entry pin is sampled HIGH after reset), a search is made for a valid user program.
3. If a valid user program is found then the execution control is transferred to it. If a valid user program is not found, the bootloader attempts to load a valid user program via the USART interface.

Remark: The sampling of pin the ISP entry pin can be disabled through programming flash location 0x0000 02FC (see Section 25.5.3 “Code Read Protection (CRP)”).

3.5.2 ROM-based APIs

Once the part has booted, the user can access several APIs located in the boot ROM to access the flash memory, optimize power consumption, and operate the USART and I2C peripherals.

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

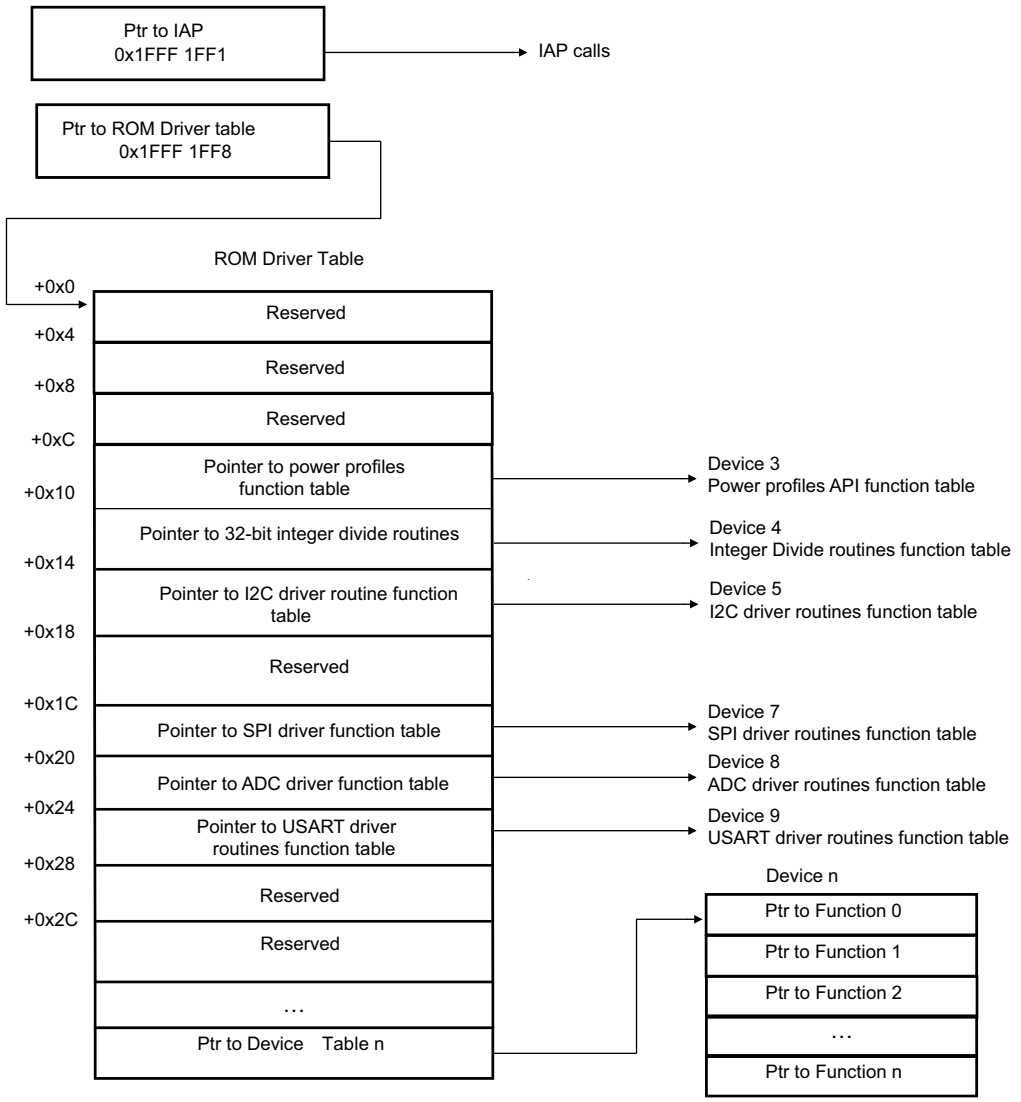


Fig 3. Boot ROM structure

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 PWRD_API_T *pPWRD; /*!< Power API function table base address */
     const ROM_DIV_API_T *divApiBase; /*!< Divider API function table base address */
     const I2CD_API_T  *pI2CD;/*!< I2C driver routines functions table */
     const uint32_t reserved5; /*!< Reserved */
     const SPID_API_T *pSPID; /*!< SPI driver API function table base address */
     const ADCD_API_T *pADCD; /*!< ADC driver API function table base address  */
     const UARTD_API_T *pUARTD; /*!< USART  driver API function table base address */
} LPC_ROM_API_T;

#define ROM_DRIVER_BASE (0x1FFF1FF8UL)
```

Table 4. API calls

API	Description	Reference
Flash IAP	Flash In-Application programming	Table 328
Power profiles API	Configure system clock and power consumption	Table 341
Integer divide routines	32-bit integer divide routines	Table 397
I2C driver	I2C ROM driver	Table 362
SPI driver	SPI ROM driver	Table 354
ADC driver	ADC ROM driver	Table 383
UART driver	USART ROM driver	Table 344

3.6 Functional description

3.6.1 Memory map after any reset

The boot block is 12 KB in size. The boot block is located in the memory region starting from the address 0x1FFF 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 25.7.2 “Memory and interrupt use for ISP and IAP”. The interrupt vectors residing in the boot block of the on-chip flash memory also become active after reset, i.e., the bottom 512 bytes of the boot block are also visible in the memory region starting from the address 0x0000 0000.

3.6.2 Boot process

During the boot process, the bootloader checks if there is valid user code in flash. The criterion for valid user code is as follows:

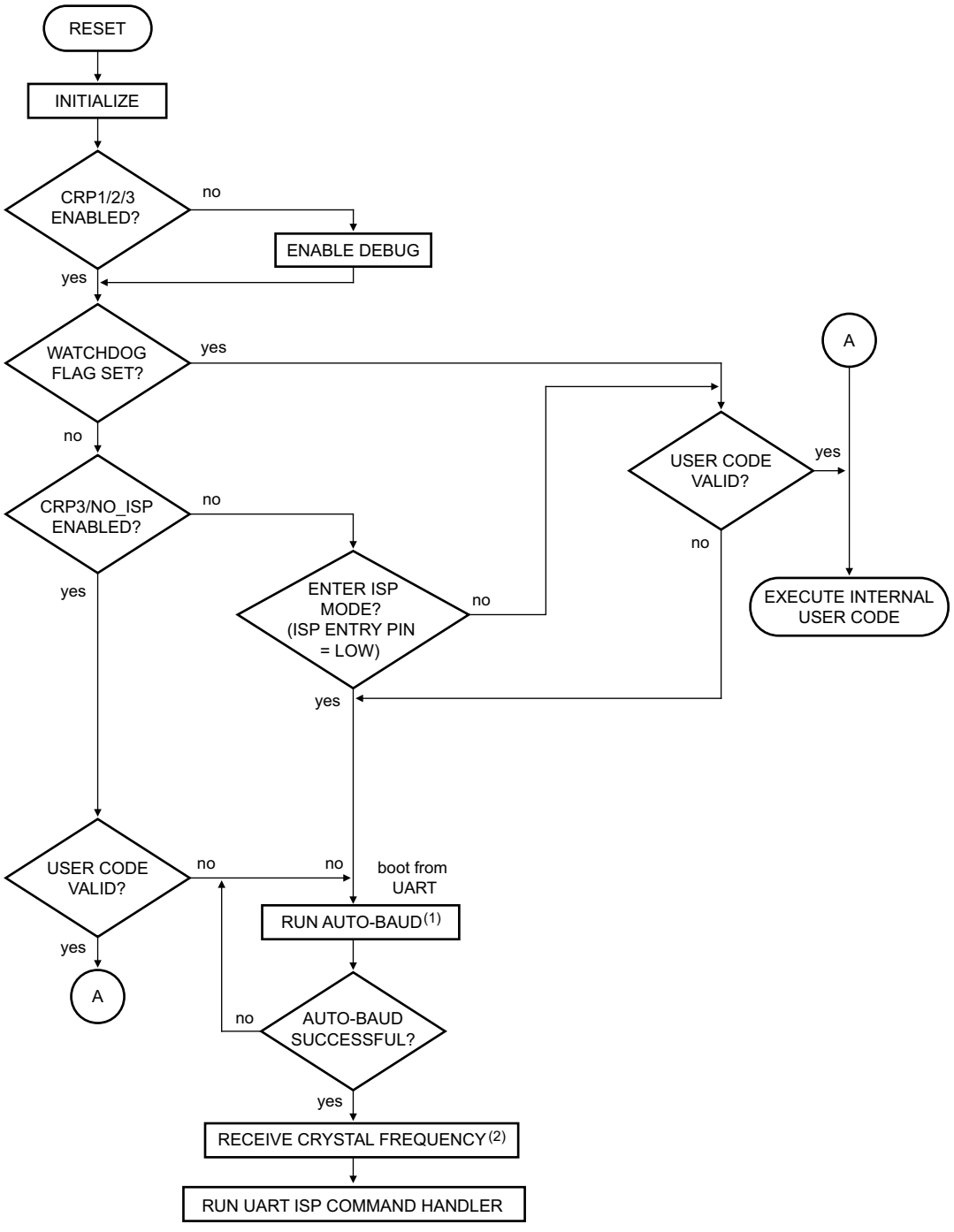
The reserved Cortex-M0+ exception vector location 7 (offset 0x0000 001C in the vector table) should contain the 2’s complement of the check-sum of table entries 0 through 6. This causes the checksum of the first 8 table entries to be 0. The bootloader code checksums the first 8 locations in sector 0 of the flash. If the result is 0, then execution control is transferred to the user code.

If the signature is not valid, the auto-baud routine synchronizes with the host via serial port USART0. 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 (the 12 MHz IRC 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, the host should send the same string ("Synchronized<CR><LF>").

The bootloader 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 bootloader code and, on the LPC800, is ignored. The bootloader configures the part to run at the 12 MHz IRC frequency.

Once the crystal frequency response 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 Table 311 "UART ISP Unlock command".

3.6.3 Boot process flowchart



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

(2) This step is included for backward compatibility and the response is ignored by the bootloader.

Fig 4. Boot process flowchart

4.1 How to read this chapter

The NVIC is identical on all LPC82x parts.

4.2 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.

4.3 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.

4.3.1 Interrupt sources

Table 5 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 5. 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 (SPI0), 0x4005 C00C (SPI1)) bit description”.
1	SPI1_IRQ	SPI1 interrupt	Same as SPI0_IRQ
2	-	Reserved	-
3	UART0_IRQ	USART0 interrupt	See Table 179 “USART Interrupt Enable read and set register (INTENSET, address 0x4006 400C (USART0), 0x4006 800C (USART1), 0x4006C00C (USART2)) bit description”

Table 5. Connection of interrupt sources to the NVIC

Interrupt number	Name	Description	Flags
4	UART1_IRQ	USART1 interrupt	Same as UART0_IRQ
5	UART2_IRQ	USART2 interrupt	Same as UART0_IRQ
6	-	Reserved	-
7	I2C1_IRQ	I2C1 interrupt	See Table 209 “Interrupt Enable Clear register (INTENCLR, address 0x4005 000C (I2C0), 0x4005 400C (I2C1), 0x4007 000C (I2C2), 0x4007 400C (I2C3)) bit description”.
8	I2C0_IRQ	I2C0 interrupt	See Table 209 “Interrupt Enable Clear register (INTENCLR, address 0x4005 000C (I2C0), 0x4005 400C (I2C1), 0x4007 000C (I2C2), 0x4007 400C (I2C3)) bit description”.
9	SCT_IRQ	State configurable timer interrupt	EVFLAG SCT event
10	MRT_IRQ	Multi-rate timer interrupt	Global MRT interrupt. GFLAG0 GFLAG1 GFLAG2 GFLAG3
11	CMP_IRQ	Analog comparator interrupt	COMPEDGE - rising, falling, or both edges can set the bit
12	WDT_IRQ	Windowed watchdog timer interrupt	WARNINT - watchdog warning interrupt
13	BOD_IRQ	BOD interrupts	BODINTVAL - BOD interrupt level
14	FLASH_IRQ	flash interrupt	-
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	DMA_IRQ	DMA interrupt	-
21	I2C2_IRQ	I2C2 interrupt	See Table 209 “Interrupt Enable Clear register (INTENCLR, address 0x4005 000C (I2C0), 0x4005 400C (I2C1), 0x4007 000C (I2C2), 0x4007 400C (I2C3)) bit description”.
22	I2C3_IRQ	I2C3 interrupt	See Table 209 “Interrupt Enable Clear register (INTENCLR, address 0x4005 000C (I2C0), 0x4005 400C (I2C1), 0x4007 000C (I2C2), 0x4007 400C (I2C3)) bit description”.
23	-	Reserved	-
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

Table 5. Connection of interrupt sources to the NVIC

Interrupt number	Name	Description	Flags
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
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

4.3.2 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 5 as NMI using the NMISRC register in the SYSCON block (Table 48). 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.

4.3.3 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).

4.4 Register description

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

Table 6. 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 7
-	-	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 8
-	-	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 9
-	-	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 10
-	-	0x284	Reserved.	0	-
IABR0	RO	0x300	Interrupt Active Bit Register 0. This register allows reading the current interrupt active state for specific peripheral functions.	0	Table 11
-	-	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 12
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 13
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 14
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 15
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 16
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 17
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 18
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 19

4.4.1 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 4.4.2).

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

Bit	Symbol	Description	Reset value
0	ISE_SPI0	Interrupt enable.	0
1	ISE_SPI1	Interrupt enable.	0
2	-	Reserved.	-
3	ISE_UART0	Interrupt enable.	0
4	ISE_UART1	Interrupt enable.	0
5	ISE_UART2	Interrupt enable.	0
6	-	Reserved.	-
7	ISE_I2C1	Interrupt enable.	0
8	ISE_I2C0	Interrupt enable.	0
9	ISE_SCT	Interrupt enable.	0
10	ISE_MRT	Interrupt enable.	0
11	ISE_CMP	Interrupt enable.	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	ISE_SDMA	Interrupt enable.	0
21	ISE_I2C2	Interrupt enable.	0
22	ISE_I2C3	Interrupt enable.	0
23	-	Reserved.	-
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

4.4.2 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 4.4.1).

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 8. Interrupt clear enable register 0 (ICER0, address 0xE000 E180)

Bit	Symbol	Description	Reset value
0	ICE_SPI0	Interrupt disable.	0
1	ICE_SPI1	Interrupt disable.	0
2	-	Reserved.	-
3	ICE_UART0	Interrupt disable.	0
4	ICE_UART1	Interrupt disable.	0
5	ICE_UART2	Interrupt disable.	0
6	-	Reserved.	-
7	ICE_I2C1	Interrupt disable.	0
8	ICE_I2C0	Interrupt disable.	0
9	ICE_SCT	Interrupt disable.	0
10	ICE_MRT	Interrupt disable.	0
11	ICE_CMP	Interrupt disable.	0
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	ICE_SDMA	Interrupt disable.	0
21	ICE_I2C2	Interrupt disable.	0
22	ICE_I2C3	Interrupt disable.	0
23	-	Reserved.	-
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

4.4.3 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 4.4.4).

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 9. 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	ISP_SPI1	Interrupt pending set.	0
2	-	Reserved.	-
3	ISP_UART0	Interrupt pending set.	0
4	ISP_UART1	Interrupt pending set.	0
5	ICE_UART2	Interrupt pending set.	0
6	-	Reserved.	-
7	ISP_I2C1	Interrupt pending set.	0
8	ISP_I2C0	Interrupt pending set.	0
9	ISP_SCT	Interrupt pending set.	0
10	ISP_MRT	Interrupt pending set.	0
11	ISP_CMP	Interrupt pending set.	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	ISP_SDMA	Interrupt pending set.	0
21	ISP_I2C2	Interrupt pending set.	0
22	ISP_I2C3	Interrupt pending set.	0
23	-	Reserved.	-
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 9. 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

4.4.4 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 4.4.3).

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 10. 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	ICP_SPI1	Interrupt pending clear.	0
2	-	Reserved.	-
3	ICP_UART0	Interrupt pending clear.	0
4	ICP_UART1	Interrupt pending clear.	0
5	ICP_UART2	Interrupt pending clear.	0
6	-	Reserved.	-
7	ICP_I2C1	Interrupt pending clear.	0
8	ICP_I2C0	Interrupt pending clear.	0
9	ICP_SCT	Interrupt pending clear.	0
10	ICP_MRT	Interrupt pending clear.	0
11	ICP_CMP	Interrupt pending clear.	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	ISP_ADC_SEQA	Interrupt pending clear.	0
17	ISP_ADC_SEQB	Interrupt pending clear.	0
18	ISP_ADC_THCMP	Interrupt pending clear.	0
19	ISP_ADC_OVR	Interrupt pending clear.	0
20	ISP_SDMA	Interrupt pending clear.	0
21	ISP_I2C2	Interrupt pending clear.	0
22	ISP_I2C3	Interrupt pending clear.	0
23	-	Reserved.	-

Table 10. Interrupt clear pending register 0 register (ICPR0, address 0xE000 E280) bit description *…continued*

Bit	Symbol	Function	Reset value
24	ICP_PININT0	Interrupt pending clear.	0
25	ICP_PININT1	Interrupt pending clear.	0
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

4.4.5 Interrupt Active Bit Register 0

The IABR0 register is a read-only register that allows reading the active state of the peripheral interrupts. Use this register to determine which peripherals are asserting an interrupt to the NVIC and may also be pending if there are enabled.

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

Write — n/a.

Read — 0 indicates that the interrupt is not active, 1 indicates that the interrupt is active.

Table 11. Interrupt Active Bit Register 0 (IABR0, address 0xE000 E300) bit description

Bit	Symbol	Function	Reset value
0	IAB_SPI0	Interrupt active.	0
1	IAB_SPI1	Interrupt active.	0
2	-	Reserved.	-
3	IAB_UART0	Interrupt active.	0
4	IAB_UART1	Interrupt active.	0
5	IAB_UART2	Interrupt active.	0
6	-	Reserved.	-
7	IAB_I2C1	Interrupt active.	0
8	IAB_I2C0	Interrupt active.	0
9	IAB_SCT	Interrupt active.	0
10	IAB_MRT	Interrupt active.	0
11	IAB_CMP	Interrupt active.	0
12	IAB_WDT	Interrupt active.	0
13	IAB_BOD	Interrupt active.	0
14	IAB_FLASH	Interrupt active.	0
15	IAB_WKT	Interrupt active.	0
16	ISP_ADC_SEQA	Interrupt active.	0
17	ISP_ADC_SEQB	Interrupt active.	0
18	ISP_ADC_THCMP	Interrupt active.	0
19	ISP_ADC_OVR	Interrupt active.	0
20	ISP_SDMA	Interrupt active.	0

Table 11. Interrupt Active Bit Register 0 (IABR0, address 0xE000 E300) bit description

Bit	Symbol	Function	Reset value
21	ISP_I2C2	Interrupt active.	0
22	ISP_I2C3	Interrupt active.	0
23	-	Reserved.	-
24	IAB_PININT0	Interrupt active.	0
25	IAB_PININT1	Interrupt active.	0
26	IAB_PININT2	Interrupt active.	0
27	IAB_PININT3	Interrupt active.	0
28	IAB_PININT4	Interrupt active.	0
29	IAB_PININT5	Interrupt active.	0
30	IAB_PININT6	Interrupt active.	0
31	IAB_PININT7	Interrupt active.	0

4.4.6 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 12. 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	IP_SPI1	Interrupt Priority. 0 = highest priority. 3 = lowest priority.
21:16	-	These bits ignore writes, and read as 0.
23:22	-	Reserved.
29:24	-	Reserved.
31:30	IP_UART0	Interrupt Priority. 0 = highest priority. 3 = lowest priority.

4.4.7 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 13. 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	IP_UART2	Interrupt Priority. 0 = highest priority. 3 = lowest priority.
21:16	-	These bits ignore writes, and read as 0.
23:22	-	Reserved.
29:24	IP_I2C1	Interrupt Priority. 0 = highest priority. 3 = lowest priority.
31:30	-	Reserved.

4.4.8 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 14. 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	IP_SCT	Interrupt Priority. 0 = highest priority. 3 = lowest priority.
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	Interrupt Priority. 0 = highest priority. 3 = lowest priority.

4.4.9 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 15. 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.

4.4.10 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 16. 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.

4.4.11 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 17. 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	IP_DMA	Interrupt Priority. 0 = highest priority. 3 = lowest priority.
13:8	-	These bits ignore writes, and read as 0.
15:14	IP_I2C2	Interrupt Priority. 0 = highest priority. 3 = lowest priority.
21:16	-	These bits ignore writes, and read as 0.
23:22	IP_I2C3	Interrupt Priority. 0 = highest priority. 3 = lowest priority.
29:24	-	Reserved.
31:30	-	Reserved.

4.4.12 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 18. 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.

4.4.13 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 19. 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.

5.1 How to read this chapter

The system configuration block is identical for all LPC820 parts.

5.2 Features

- Clock control
  - Configure the system PLL.
  - Configure system oscillator and watchdog oscillator.
  - Enable clocks to individual peripherals and memories.
  - Configure clock output.
  - Configure clock dividers, digital filter clock, and USART baud rate 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.
- MTB trace start and stop.
- Interrupt latency control.
- Select a source for the NMI.
- Calibrate system tick timer.

5.3 Basic configuration

Configure the SYSCON block as follows:

- The SYSCON uses the CLKIN, CLKOUT, $\overline{\text{RESET}}$, and XTALIN/OUT pins. Configure the pin functions through the switch matrix. See Section 5.4.
- No clock configuration is needed. The clock to the SYSCON block is always enabled. By default, the SYSCON block is clocked by the IRC.

5.3.1 Set up the PLL

The PLL creates a stable output clock at a higher frequency than the input clock. If you need a main clock with a frequency higher than the 12 MHz IRC clock, use the PLL to boost the input frequency.

1. Power up the system PLL in the PDRUNCFG register.

   Section 5.6.33 “Power configuration register”
2. Select the PLL input in the SYSPLLCLKSEL register. You have the following input options:

- IRC: 12 MHz internal oscillator.
- System oscillator: External crystal oscillator using the XTALIN/XTALOUT pins.
- External clock input CLKIN. Select this pin through the switch matrix.

Section 5.6.9 “System PLL clock source select register”

3. Update the PLL clock source in the SYSPLLCLKUEN register.

   Section 5.6.10 “System PLL clock source update register”
4. Configure the PLL M and N dividers.

   Section 5.6.3 “System PLL control register”
5. Wait for the PLL to lock by monitoring the PLL lock status.

   Section 5.6.4 “System PLL status register”

5.3.2 Configure the main clock and system clock

The clock source for the registers and memories is derived from main clock. The main clock can be sourced from the IRC at a fixed clock frequency of 12 MHz or from the PLL.

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:
   - IRC: 12 MHz internal oscillator (default).
   - PLL output: You must configure the PLL to use the PLL output.

   Section 5.6.11 “Main clock source select register”
2. Update the main clock source.

   Section 5.6.12 “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 5.6.13 “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 5.6.14 “System clock control register”

5.3.3 Set up the system oscillator using XTALIN and XTALOUT

To use the system oscillator with the LPC800, you need to assign the XTALIN and XTALOUT pins, which connect to the external crystal, through the fixed-pin function in the switch matrix. XTALIN and XTALOUT can only be assigned to pins PIO0_8 and PIO0_9.

1. In the IOCON block, remove the pull-up and pull-down resistors in the IOCON registers for pins PIO0_8 and PIO0_9.
2. In the switch matrix block, enable the 1-bit functions for XTALIN and XTALOUT.
3. In the SYSOSCCTRL register, disable the BYPASS bit and select the oscillator frequency range according to the desired oscillator output clock.

Related registers:

Table 96 “PIO0_8 register (PIO0_8, address 0x4004 4038) bit description”

Table 95 “PIO0_9 register (PIO0_9, address 0x4004 4034) bit description”

Table 79 “Pin enable register 0 (PINENABLE0, address 0x4000 C1C0) bit description”

Table 26 “System oscillator control register (SYSOSCCTRL, address 0x4004 8020) bit description”

5.4 Pin description

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

See Section 7.3.1 “Connect an internal signal to a package pin” to assign the CLKOUT function to a pin.

See Section 7.3.2 to enable the clock input, the oscillator pins, and the external reset input.

Table 20. SYSCON pin description

Function	Direction	Pin	Description	SWM register	Reference
CLKOUT	O	any	CLKOUT clock output.	PINASSIGN8	Table 75
CLKIN	I	PIO0_1/ACMP_I2/CLKIN	External clock input to the system PLL. Disable the ACMP_I2 function in the PINENABLE register.	PINENABLE0	Table 79
XTALIN	I	PIO0_8/XTALIN	Input to the system oscillator.	PINENABLE0	Table 79
XTALOUT	O	PIO0_9/XTALOUT	Output from the system oscillator.	PINENABLE0	Table 79
$\overline{\text{RESET}}$	I	$\overline{\text{RESET}}$/PIO0_5	External reset input	PINENABLE0	Table 79

5.5 General description

5.5.1 Clock generation

The system control block generates all clocks for the chip. Only the low-power oscillator used for wake-up timing is controlled by the PMU. Except for the USART clock and the clock to configure the glitch filters of the digital I/O pins, the clocks to the core and peripherals run at the same frequency. The maximum system clock frequency is 30 MHz. See Figure 5.

Remark: The main clock frequency is limited to 100 MHz.

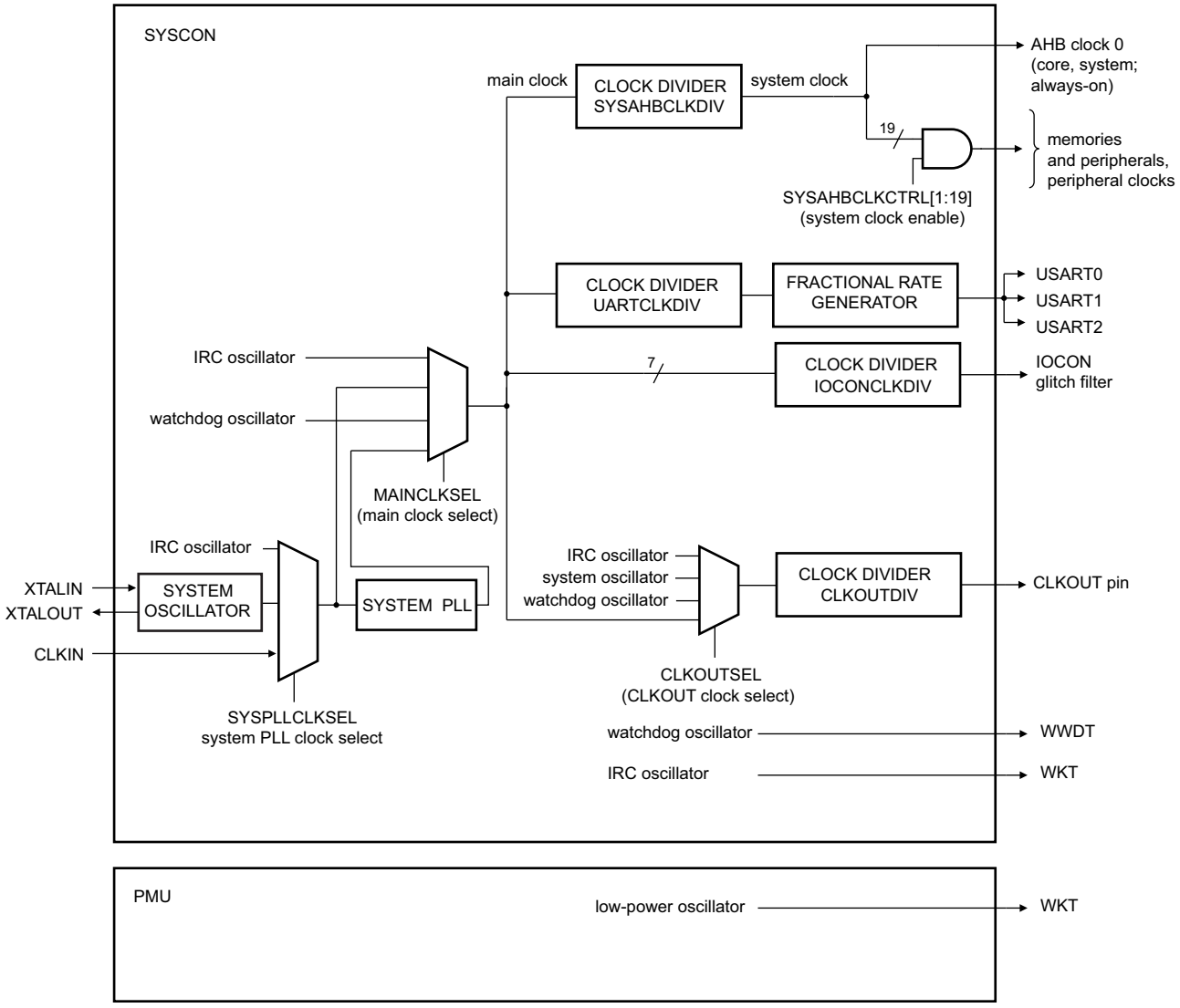


Fig 5. Clock generation

5.5.2 Power control of analog components

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

Section 5.6.31 “Deep-sleep mode configuration register”

Section 5.6.3 “System PLL control register”

Section 5.6.6 “Watchdog oscillator control register”

Section 5.6.5 “System oscillator control register”

5.5.3 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 watchdog 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 5.6.33 “Power configuration register”

Section 5.6.30 “Start logic 1 interrupt wake-up enable register”

5.5.4 Reset and interrupt control

The peripheral reset control register in the system control register allows to assert and release individual peripheral resets. See Table 23.

Up to eight external pin interrupts can be assigned to any digital pin in the system control block (see Section 5.6.28 “Pin interrupt select registers”).

5.6 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 not shown in Table 21 are reserved and should not be written to.

Table 21. Register overview: System configuration (base address 0x4004 8000)

Name	Access	Offset	Description	Reset value	Reset value after boot	Reference
SYSMEMREMAP	R/W	0x000	System memory remap	0x2		Table 22
PRESETCTRL	R/W	0x004	Peripheral reset control	0x0001 FFFF		Table 23
SYSPLLCTRL	R/W	0x008	System PLL control	0		Table 24
SYSPLLSTAT	R	0x00C	System PLL status	0		Table 25
-	-	0x010	Reserved	-		-
-	-	0x014	Reserved	-		-
SYSOSCCTRL	R/W	0x020	System oscillator control	0x000		Table 26
WDTOSCCTRL	R/W	0x024	Watchdog oscillator control	0x0A0		Table 27
IRCCTRL	R/W	0x028	IRC control	0x080		Table 28
-	-	0x02C	Reserved	-		-
SYSRSTSTAT	R/W	0x030	System reset status register	0		Table 29
SYSPLLCLKSEL	R/W	0x040	System PLL clock source select	0		Table 30
SYSPLLCLKUEN	R/W	0x044	System PLL clock source update enable	0		Table 31
MAINCLKSEL	R/W	0x070	Main clock source select	0		Table 32
MAINCLKUEN	R/W	0x074	Main clock source update enable	0		Table 33
SYSAHBCLKDIV	R/W	0x078	System clock divider	1		Table 34
SYSAHBCLKCTRL	R/W	0x080	System clock control	0xDF		Table 35
UARTCLKDIV	R/W	0x094	USART clock divider	0		Table 36

Table 21. Register overview: System configuration (base address 0x4004 8000) *…continued*

Name	Access	Offset	Description	Reset value	Reset value after boot	Reference
-	-	0x098	Reserved	-		-
-	-	0x09C	Reserved	-		-
-	-	0x0A0 - 0x0BC	Reserved	-		-
-	-	0x0CC	Reserved	-		-
CLKOUTSEL	R/W	0x0E0	CLKOUT clock source select	0		Table 37
CLKOUTUEN	R/W	0x0E4	CLKOUT clock source update enable	0		Table 38
CLKOUTDIV	R/W	0x0E8	CLKOUT clock divider	0		Table 39
UARTFRGDIV	R/W	0x0F0	USART1 to USART4 common fractional generator divider value	0		Table 40
UARTFRGMULT	R/W	0x0F4	USART1 to USART4 common fractional generator multiplier value	0		Table 41
EXTTRACECMD	R/W	0x0FC	External trace buffer command register	0		Table 42
PIOPORCAP0	R	0x100	POR captured PIO status 0	user dependent		Table 43
-	-	0x104	Reserved	-		-
IOCONCLKDIV6	R/W	0x134	Peripheral clock 6 to the IOCON block for programmable glitch filter	0		Table 44
IOCONCLKDIV5	R/W	0x138	Peripheral clock 5 to the IOCON block for programmable glitch filter	0		Table 44
IOCONCLKDIV4	R/W	0x13C	Peripheral clock 4 to the IOCON block for programmable glitch filter	0		Table 44
IOCONCLKDIV3	R/W	0x140	Peripheral clock 3 to the IOCON block for programmable glitch filter	0		Table 44
IOCONCLKDIV2	R/W	0x144	Peripheral clock 2 to the IOCON block for programmable glitch filter	0		Table 44
IOCONCLKDIV1	R/W	0x148	Peripheral clock 1 to the IOCON block for programmable glitch filter	0		Table 44
IOCONCLKDIV0	R/W	0x14C	Peripheral clock 0 to the IOCON block for programmable glitch filter	0		Table 44
BODCTRL	R/W	0x150	Brown-Out Detect	0		Table 45
SYSTCKCAL	R/W	0x154	System tick counter calibration	0		Table 46
-	R/W	0x168	Reserved	-		-
IRQLATENCY	R/W	0x170	IQR delay. Allows trade-off between interrupt latency and determinism.	0x0000 0010		Table 47
NMISRC	R/W	0x174	NMI Source Control	0		Table 48
PINTSEL0	R/W	0x178	GPIO Pin Interrupt Select register 0	0		Table 49
PINTSEL1	R/W	0x17C	GPIO Pin Interrupt Select register 1	0		Table 49
PINTSEL2	R/W	0x180	GPIO Pin Interrupt Select register 2	0		Table 49
PINTSEL3	R/W	0x184	GPIO Pin Interrupt Select register 3	0		Table 49
PINTSEL4	R/W	0x188	GPIO Pin Interrupt Select register 4	0		Table 49
PINTSEL5	R/W	0x18C	GPIO Pin Interrupt Select register 5	0		Table 49
PINTSEL6	R/W	0x190	GPIO Pin Interrupt Select register 6	0		Table 49

Table 21. Register overview: System configuration (base address 0x4004 8000) *…continued*

Name	Access	Offset	Description	Reset value	Reset value after boot	Reference
PINTSEL7	R/W	0x194	GPIO Pin Interrupt Select register 7	0		Table 49
STARTERP0	R/W	0x204	Start logic 0 pin wake-up enable register	0		Table 50
STARTERP1	R/W	0x214	Start logic 1 interrupt wake-up enable register	0		Table 51
PDSLEEPCFG	R/W	0x230	Power-down states in deep-sleep mode	0xFFFF		Table 52
PDAWAKECFG	R/W	0x234	Power-down states for wake-up from deep-sleep	0xEDF0		Table 53
PDRUNCFG	R/W	0x238	Power configuration register	0xEDF0		Table 54
DEVICE_ID	R	0x3F8	Device ID	part dependent		Table 55

5.6.1 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 22. 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	-

5.6.2 Peripheral reset control register

The PRESETCTRL 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 23. Peripheral reset control register (PRESETCTRL, address 0x4004 8004) bit description

Bit	Symbol	Value	Description	Reset value
0	SPI0_RST_N		SPI0 reset control	1
		0	Assert the SPI0 reset.	
		1	Clear the SPI0 reset.	

Table 23. Peripheral reset control register (PRESETCTRL, address 0x4004 8004) bit description

Bit	Symbol	Value	Description	Reset value
1	SPI1_RST_N		SPI1 reset control	1
		0	Assert the SPI1 reset.	
		1	Clear the SPI1 reset.	
2	UARTFRG_RST_N		USART fractional baud rate generator (UARTFRG) reset control	1
		0	Assert the UARTFRG reset.	
		1	Clear the UARTFRG reset.	
3	UART0_RST_N		USART0 reset control	1
		0	Assert the USART0 reset.	
		1	Clear the USART0 reset.	
4	UART1_RST_N		USART1 reset control	1
		0	Assert the USART reset.	
		1	Clear the USART1 reset.	
5	UART2_RST_N		USART2 reset control	1
		0	Assert the USART2 reset.	
		1	Clear the USART2 reset.	
6	I2C0_RST_N		I2C0 reset control	1
		0	Assert the I2C0 reset.	
		1	Clear the I2C0 reset.	
7	MRT_RST_N		Multi-rate timer (MRT) reset control	1
		0	Assert the MRT reset.	
		1	Clear the MRT reset.	
8	SCT_RST_N		SCT reset control	1
		0	Assert the SCT reset.	
		1	Clear the SCT reset.	
9	WKT_RST_N		Self-wake-up timer (WKT) reset control	1
		0	Assert the WKT reset.	
		1	Clear the WKT reset.	
10	GPIO_RST_N		GPIO and GPIO pin interrupt reset control	1
		0	Assert the GPIO reset.	
		1	Clear the GPIO reset.	
11	FLASH_RST_N		Flash controller reset control	1
		0	Assert the flash controller reset.	
		1	Clear the flash controller reset.	
12	ACMP_RST_N		Analog comparator reset control	1
		0	Assert the analog comparator reset.	
		1	Clear the analog comparator controller reset.	
13	-	-	Reserved	-

Table 23. Peripheral reset control register (PRESETCTRL, address 0x4004 8004) bit description

Bit	Symbol	Value	Description	Reset value
14	I2C1_RST_N		I2C1 reset control	1
		0	Assert the I2C1 reset.	
		1	Clear the I2C1 reset.	
15	I2C2_RST_N		I2C2 reset control	1
		0	Assert the I2C2 reset.	
		1	Clear the I2C2 reset.	
16	I2C3_RST_N		I2C3 reset control	1
		0	Assert the I2C3 reset.	
		1	Clear the I2C3 reset.	
31:17	-	-	Reserved	-

5.6.3 System PLL control register

This register connects and enables the system PLL and configures the PLL multiplier and divider values. The PLL accepts an input frequency from 10 MHz to 25 MHz from various clock sources. The input frequency is multiplied to a higher frequency and then divided down to provide the actual clock used by the CPU, peripherals, and memories. The PLL can produce a clock up to the maximum allowed for the CPU.

Remark: The divider values for P and M must be selected so that the PLL output clock frequency FCLKOUT is lower than 100 MHz.

Table 24. System PLL control register (SYSPLLCTRL, address 0x4004 8008) bit description

Bit	Symbol	Value	Description	Reset value
4:0	MSEL		Feedback divider value. The division value M is the programmed MSEL value + 1. 00000: Division ratio M = 1 to 11111: Division ratio M = 32	0
6:5	PSEL		Post divider ratio P. The division ratio is 2 × P.	0
		0x0	P = 1	
		0x1	P = 2	
		0x2	P = 4	
		0x3	P = 8	
31:7	-	-	Reserved. Do not write ones to reserved bits.	-

5.6.4 System PLL status register

This register is a Read-only register and supplies the PLL lock status (see Section 5.7.4.1).

Table 25. System PLL status register (SYSPLLSTAT, address 0x4004 800C) bit description

Bit	Symbol	Value	Description	Reset value
0	LOCK		PLL lock status	0
		0	PLL not locked	
		1	PLL locked	
31:1	-	-	Reserved	-

5.6.5 System oscillator control register

This register configures the frequency range for the system oscillator. The system oscillator itself is powered on or off in the PDRUNCFG register. See Table 54.

Table 26. System oscillator control register (SYSOSCCTRL, address 0x4004 8020) bit description

Bit	Symbol	Value	Description	Reset value
0	BYPASS		Bypass system oscillator	0x0
		0	Disabled. Oscillator is not bypassed.	
		1	Enabled. PLL input (sys_osc_clk) is fed directly from the XTALIN pin bypassing the oscillator. Use this mode when using an external clock source instead of the crystal oscillator.	
1	FREQRANGE		Determines oscillator frequency range.	0x0
		0	1 - 20 MHz frequency range.	
		1	15 - 25 MHz frequency range	
31:2	-	-	Reserved	0x00

5.6.6 Watchdog oscillator control register

This register configures the watchdog oscillator. The oscillator consists of an analog and a digital part. The analog part contains the oscillator function and generates an analog clock (Fclkana). With the digital part, the analog output clock (Fclkana) can be divided to the required output clock frequency wdt_osc_clk. The analog output frequency (Fclkana) can be adjusted with the FREQSEL bits between 600 kHz and 4.6 MHz. With the digital part Fclkana will be divided (divider ratios = 2, 4,...,64) to wdt_osc_clk using the DIVSEL bits.

The output clock frequency of the watchdog oscillator can be calculated as wdt_osc_clk = Fclkana/(2 × (1 + DIVSEL)) = 9.3 kHz to 2.3 MHz (nominal values).

Remark: Any setting of the FREQSEL bits will yield a Fclkana value within ±40% of the listed frequency value. The watchdog oscillator is the clock source with the lowest power consumption. If accurate timing is required, use the IRC or system oscillator.

Remark: The frequency of the watchdog oscillator is undefined after reset. The watchdog oscillator frequency must be programmed by writing to the WDTOSCCTRL register before using the watchdog oscillator.

Table 27. Watchdog oscillator control register (WDTOSCCTRL, address 0x4004 8024) bit description

Bit	Symbol	Value	Description	Reset value
4:0	DIVSEL		Select divider for Fclkana. wdt_osc_clk = Fclkana/ (2 × (1 + DIVSEL)) 00000: 2 × (1 + DIVSEL) = 2 00001: 2 × (1 + DIVSEL) = 4 to 11111: 2 × (1 + DIVSEL) = 64	0
8:5	FREQSEL		Select watchdog oscillator analog output frequency (Fclkana).	0x00
		0x1	0.6 MHz	
		0x2	1.05 MHz	
		0x3	1.4 MHz	
		0x4	1.75 MHz	
		0x5	2.1 MHz	
		0x6	2.4 MHz	
		0x7	2.7 MHz	
		0x8	3.0 MHz	
		0x9	3.25 MHz	
		0xA	3.5 MHz	
		0xB	3.75 MHz	
		0xC	4.0 MHz	
		0xD	4.2 MHz	
		0xE	4.4 MHz	
		0xF	4.6 MHz	
31:9	-	-	Reserved	0x00

5.6.7 Internal resonant crystal control register

This register can be used to re-trim the on-chip 12 MHz oscillator. Note that the factory-preset trim value is written to this register by the boot code on start-up.

Table 28. Internal resonant crystal control register (IRCCTRL, address 0x4004 8028) bit description

Bit	Symbol	Description	Reset value
7:0	TRIM	Trim value	0x80, then flash will reprogram
31:8	-	Reserved	0x00

5.6.8 System reset status register

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

The reset value given in Table 29 applies to the POR reset.

Table 29. System reset status register (SYSRSTSTAT, address 0x4004 8030) bit description

Bit	Symbol	Value	Description	Reset value
0	POR		POR reset status	0
		0	No POR detected	
		1	POR detected. Writing a one clears this reset.	
1	EXTRST		Status of the external 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	BOD		Status of the Brown-out detect reset	0
		0	No BOD reset detected	
		1	BOD reset detected. Writing a one clears this reset.	
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	-

5.6.9 System PLL clock source select register

This register selects the clock source for the system PLL. The SYSPLLCLKUEN register (see Section 5.6.10) must be toggled from LOW to HIGH for the update to take effect.

Table 30. System PLL clock source select register (SYSPLLCLKSEL, address 0x4004 8040) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SEL		System PLL clock source	0
		0x0	IRC	
		0x1	Crystal Oscillator (SYSOSC)	
		0x2	Reserved.	
		0x3	CLKIN. External clock input.	
31:2	-	-	Reserved	-

5.6.10 System PLL clock source update register

This register updates the clock source of the system PLL with the new input clock after the SYSPLLCLKSEL register has been written to. In order for the update to take effect, first write a zero to the SYSPLLUEN register and then write a one to SYSPLLUEN.

Table 31. System PLL clock source update enable register (SYSPLLCLKUEN, address 0x4004 8044) bit description

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

5.6.11 Main clock source select register

This register selects the main system clock, which can be the system PLL (sys_pllclkout), or the watchdog oscillator, or the IRC oscillator. The main system clock clocks the core, the peripherals, and the memories.

Bit 0 of the MAINCLKUEN register (see Section 5.6.12) must be toggled from 0 to 1 for the update to take effect.

Table 32. Main clock source select register (MAINCLKSEL, address 0x4004 8070) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SEL		Clock source for main clock	0
		0x0	IRC Oscillator	
		0x1	PLL input	
		0x2	Watchdog oscillator	
		0x3	PLL output	
31:2	-	-	Reserved	-

5.6.12 Main clock source update enable register

This 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 33. Main clock source update enable register (MAINCLKUEN, address 0x4004 8074) 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	-

5.6.13 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 34. System clock divider register (SYSAHBCLKDIV, address 0x4004 8078) bit description

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

5.6.14 System clock control register

The SYSAHBCLKCTRL 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 35. System clock control register (SYSAHBCLKCTRL, 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_1		Enables clock for SRAM0 and SRAM1.	1
		0	Disable	
		1	Enable	
3	FLASHREG		Enables clock for flash register interface.	1
		0	Disable	
		1	Enable	
4	FLASH		Enables clock for flash.	1
		0	Disable	
		1	Enable	
5	I2C0		Enables clock for I2C0.	0
		0	Disable	
		1	Enable	
6	GPIO		Enables clock for GPIO port registers and GPIO pin interrupt registers.	1
		0	Disable	
		1	Enable	

Table 35. System clock control register (SYSAHBCLKCTRL, address 0x4004 8080) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
7	SWM		Enables clock for switch matrix.	1
		0	Disable	
		1	Enable	
8	SCT		Enables clock for state configurable timer SCTimer/PWM.	0
		0	Disable	
		1	Enable	
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	SPI1		Enables clock for SPI1.	
		0	Disable	
		1	Enable	
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	UART2		Enables clock for USART2.	0
		0	Disable	
		1	Enable	
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	

Table 35. System clock control register (SYSAHBCLKCTRL, address 0x4004 8080) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
20	-	-	Reserved	-
21	I2C1		Enables clock to I2C1.	0
		0	Disable	
		1	Enable	
22	I2C2		Enables clock to I2C2.	0
		0	Disable	
		1	Enable	
23	I2C3		Enables clock to I2C3.	0
		0	Disable	
		1	Enable	
24	ADC		Enables clock to ADC.	0
		0	Disable	
		1	Enable	
25	-	-	Reserved	-
26	MTB		Enables clock to micro-trace buffer control registers. Turn on this clock when using the micro-trace buffer for debug purposes.	0
		0	Disable	
		1	Enable	
28:27	-	-	Reserved	-
29	DMA		Enables clock to DMA.	0
		0	Disable	
		1	Enable	
31:30	-	-	Reserved	-

5.6.15 USART clock divider register

This register configures the clock for the fractional baud rate generator and all USARTs. The UART clock can be disabled by setting the DIV field to zero (this is the default setting).

Table 36. USART clock divider register (UARTCLKDIV, address 0x4004 8094) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	USART fractional baud rate generator clock divider values. 0: Clock disabled. 1: Divide by 1. to 255: Divide by 255.	0
31:8	-	Reserved	-

5.6.16 CLKOUT clock source select register

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

Bit 0 of the CLKOUTUEN register (see Section 5.6.17) must be toggled from 0 to 1 for the update to take effect.

Table 37. CLKOUT clock source select register (CLKOUTSEL, address 0x4004 80E0) bit description

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

5.6.17 CLKOUT clock source update enable register

This register updates the clock source of the CLKOUT pin with the new clock after the CLKOUTSEL register has been written to. In order for the update to take effect at the input of the CLKOUT pin, first write a zero to bit 0 of this register, then write a one.

Table 38. CLKOUT clock source update enable register (CLKOUTUEN, address 0x4004 80E4) bit description

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

5.6.18 CLKOUT clock divider register

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

Table 39. CLKOUT clock divider registers (CLKOUTDIV, address 0x4004 80E8) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	CLKOUT clock divider values 0: Disable CLKOUT clock divider. 1: Divide by 1. to 255: Divide by 255.	0
31:8	-	Reserved	-

5.6.19 USART fractional generator divider value register

All USART peripherals share a common clock U_PCLK, which can be adjusted by a fractional divider:

U_PCLK = UARTCLKDIV/(1 + MULT/DIV).

UARTCLKDIV is the USART clock configured in the UARTCLKDIV register.

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

1. 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 U_PCLK.
2. The MULT value of the fractional divider is programmed in the UARTFRGMULT register. See Table 41.

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 40. USART fractional generator divider value register (UARTFRGDIV, address 0x4004 80F0) bit description

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

5.6.20 USART fractional generator multiplier value register

All USART peripherals share a common clock U_PCLK, which can be adjusted by a fractional divider:

U_PCLK = UARTCLKDIV/(1 + MULT/DIV).

UARTCLKDIV is the USART clock configured in the UARTCLKDIV register.

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

1. The DIV denominator of the fractional divider value is programmed in the UARTFRGDIV register. See Table 40.
2. 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.

See also:

Section 13.3.1 “Configure the USART clock and baud rate”

Section 13.7.1 “Clocking and baud rates”

Table 41. USART fractional generator multiplier value register (UARTFRGMULT, address 0x4004 80F4) bit description

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

5.6.21 External trace buffer command register

This register works in conjunction with the MTB master register to start and stop tracing. Also see Section 31.5.4.

Table 42. External trace buffer command register (EXTTRACECMD, address 0x4004 80FC) bit description

Bit	Symbol	Description	Reset value
0	START	Trace start command. Writing a one to this bit sets the TSTART signal to the MTB to HIGH and starts tracing if the TSTARTEN bit in the MTB master register is set to one as well.	0
1	STOP	Trace stop command. Writing a one to this bit sets the TSTOP signal in the MTB to HIGH and stops tracing if the TSTOPEN bit in the MTB master register is set to one as well.	0
31:2	-	Reserved	0

5.6.22 POR captured PIO 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 43. POR captured PIO status register 0 (PIOPORCAP0, address 0x4004 8100) bit description

Bit	Symbol	Description	Reset value
17:0	PIOSTAT	State of PIO0_17 through PIO0_0 at power-on reset	Implementation dependent
31:18	-	Reserved.	-

5.6.23 IOCON glitch filter clock divider registers 6 to 0

These registers individually configure the seven peripheral input clocks (IOCONFILTR_PCLK) to the IOCON programmable glitch filter. The clocks can be shut down by setting the DIV bits to 0x0.

Table 44. IOCON glitch filter clock divider registers 6 to 0 (IOCONCLKDIV[6:0], address 0x4004 8134 (IOCONCLKDIV6) to 0x004 814C (IOCONFILTCLKDIV0)) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	IOCON glitch filter clock divider values 0: Disable IOCONFILTR_PCLK. 1: Divide by 1. to 255: Divide by 255.	0
31:8	-	Reserved	0x00

5.6.24 BOD control register

The BOD control register selects four separate threshold values for sending a BOD interrupt to the NVIC and for forced reset. Reset and interrupt threshold values listed in Table 45 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 LPC800 data sheet for the BOD reset and interrupt levels.

Table 45. BOD control register (BODCTRL, address 0x4004 8150) bit description

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

5.6.25 System tick counter calibration register

This register determines the value of the SYST_CALIB register.

Table 46. 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	-

5.6.26 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 (e.g. 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 47. 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	-

5.6.27 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 5. For a description of the NMI functionality, see Section 4.3.2.

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 48. 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 5 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.

5.6.28 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 28 for pins PIO0_0 to PIO0_28 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 5).

To use the selected pins for pin interrupts or the pattern match engine, see Section 10.5.2 “Pattern match engine”.

Table 49. 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_28 correspond to numbers 0 to 28).	0
31:6	-	Reserved	-

5.6.29 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 5 “Connection of interrupt sources to the NVIC”.

Table 50. 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	

Table 50. Start logic 0 pin wake-up enable register 0 (STARTERP0, address 0x4004 8204) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
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	-

5.6.30 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 5 “Connection of interrupt sources to the NVIC”.

Table 51. 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	
1	SPI1		SPI1 interrupt wake-up	0
		0	Disabled	
		1	Enabled	
2	-		Reserved	-
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	

Table 51. Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
5	USART2		USART2 interrupt wake-up. Configure USART in synchronous slave mode.	0
		0	Disabled	
		1	Enabled	
6	-		Reserved	-
7	I2C1		I2C1 interrupt wake-up.	0
		0	Disabled	
		1	Enabled	
8	I2C0		I2C0 interrupt wake-up.	0
		0	Disabled	
		1	Enabled	
11:9	-		Reserved	-
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	
20:16	-		Reserved.	-
21	I2C2		I2C2 interrupt wake-up.	0
		0	Disabled	
		1	Enabled	
22	I2C3		I2C3 interrupt wake-up.	0
		0	Disabled	
		1	Enabled	
31:23	-		Reserved.	-

5.6.31 Deep-sleep mode configuration register

The bits in this register (BOD_PD and WDTOSC_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 watchdog oscillator, which can be configured to remain running through this register. The WDTOSC_PD value written to the PDSLEEPCFG register is overwritten if the LOCK bit in the WWDT MOD register (see Table 253) is set. See Section 17.5.3 for details.

Table 52. 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	WDTOSC_PD		Watchdog 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 watchdog oscillator is always running.	1
		0	Powered	
		1	Powered down	
15:7	-		Reserved	0b111111111
31:16	-	-	Reserved	0

5.6.32 Wake-up configuration register

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

Table 53. Wake-up configuration register (PDAWAKECFG, address 0x4004 8234) bit description

Bit	Symbol	Value	Description	Reset value
0	IRCOUT_PD		IRC oscillator output wake-up configuration	0
		0	Powered	
		1	Powered down	
1	IRC_PD		IRC 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	SYSOSC_PD		Crystal oscillator wake-up configuration	1
		0	Powered	
		1	Powered down	

Table 53. Wake-up configuration register (PDAWAKECFG, address 0x4004 8234) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
6	WDTOSC_PD		Watchdog 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 watchdog oscillator is always running.	1
		0	Powered	
		1	Powered down	
7	SYSPLL_PD		System PLL wake-up configuration	1
		0	Powered	
		1	Powered down	
11:8	-		Reserved. Always write these bits as 0b1101	0b1101
14:12	-		Reserved. Always write these bits as 0b110	0b110
15	ACMP		Analog comparator wake-up configuration	1
		0	Powered	
		1	Powered down	
31:16	-	-	Reserved	0

5.6.33 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 IRC.

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

The system oscillator requires typically 500 µs to start up after the SYSOSC_PD bit has been changed from 1 to 0. There is no hardware flag to monitor the state of the system oscillator. Therefore, add a software delay of about 500 µs before using the system oscillator after power-up.

Table 54. Power configuration register (PDRUNCFG, address 0x4004 8238) bit description

Bit	Symbol	Value	Description	Reset value
0	IRCOUT_PD		IRC oscillator output power	0
		0	Powered	
		1	Powered down	
1	IRC_PD		IRC oscillator power down	0
		0	Powered	
		1	Powered down	
2	FLASH_PD		Flash power down	0
		0	Powered	
		1	Powered down	

Table 54. Power configuration register (PDRUNCFG, address 0x4004 8238) bit description

Bit	Symbol	Value	Description	Reset value
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	SYSOSC_PD		Crystal oscillator power down. After power-up, add a software delay of approximately 500 µs before using.	1
		0	Powered	
		1	Powered down	
6	WDTOSC_PD		Watchdog 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 watchdog oscillator is always running.	1
		0	Powered	
		1	Powered down	
7	SYSPLL_PD		System PLL power down	1
		0	Powered	
		1	Powered down	
11:8	-		Reserved. Always write these bits as 0b1101	0b1101
14:12	-		Reserved. Always write these bits as 0b110	0b110
15	ACMP		Analog comparator power down	1
		0	Powered	
		1	Powered down	
31:16	-	-	Reserved	0

5.6.34 Device ID register

This 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 322).

Table 55. Device ID register (DEVICE_ID, address 0x4004 83F8) bit description

Bit	Symbol	Description	Reset value
31:0	DEVICEID	0x0000 8241 = LPC824M201JHI33 0x0000 8221 = LPC822M101JHI33 0x0000 8242 = LPC824M201JDH20 0x0000 8222 = LPC822M101JDH20	part-dependent

5.7 Functional description

5.7.1 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 IRC causing reset to remain asserted until the external Reset is de-asserted, the oscillator is running, and the flash controller has completed its initialization.

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

1. The IRC starts up. After the IRC-start-up time (maximum of 6 μs on power-up), the IRC provides a stable clock output.
2. The flash is powered up. This takes approximately 100 μs. Then the flash initialization sequence is started, which takes about 250 cycles.
3. The boot code in the ROM starts. The boot code performs the boot tasks and may jump to the flash.

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

5.7.2 Start-up behavior

See Figure 6 for the start-up timing after reset. The IRC is the default clock at Reset and provides a clean system clock shortly after the supply voltage reaches the threshold value of 1.8 V.

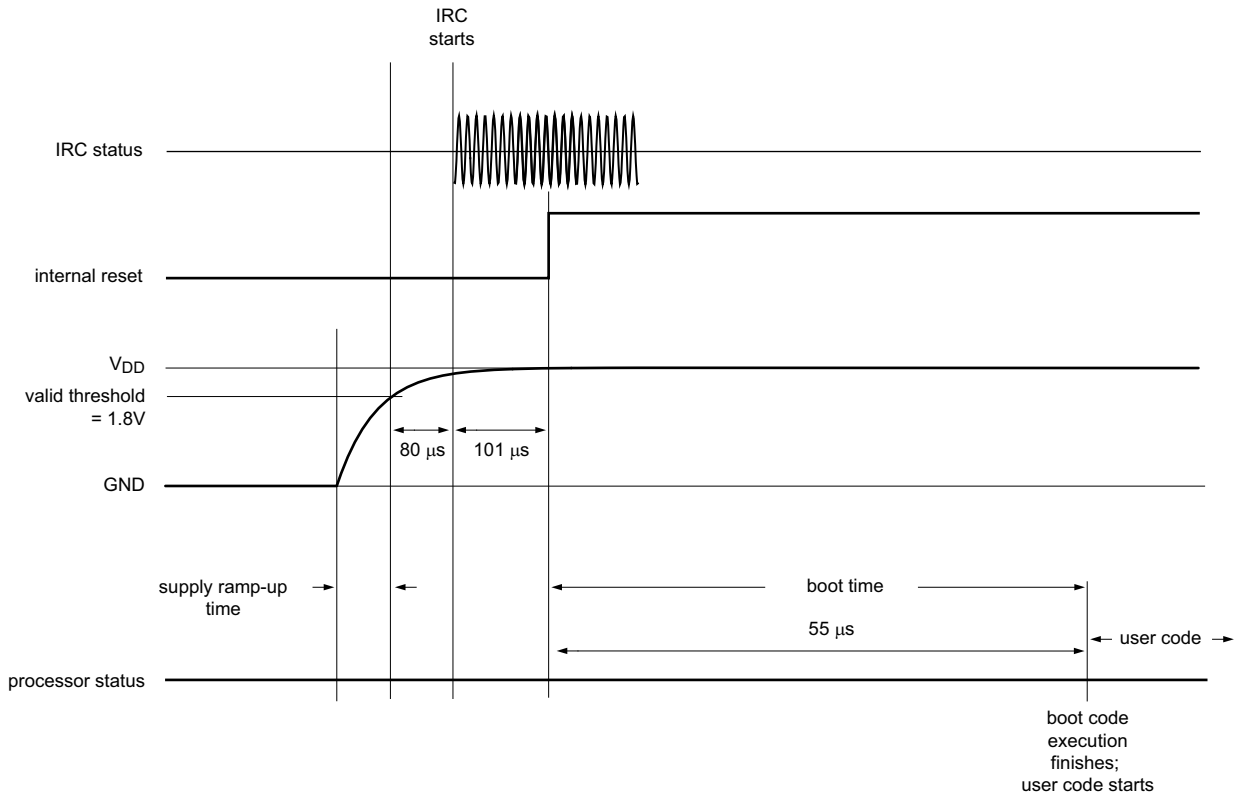


Fig 6. Start-up timing

5.7.3 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 45).

The interrupt signal can be enabled for interrupt in the Interrupt Enable Register in the NVIC (see Table 6) 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 51) 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.

5.7.4 System PLL functional description

The LPC82X uses the system PLL to create the clocks for the core and peripherals.

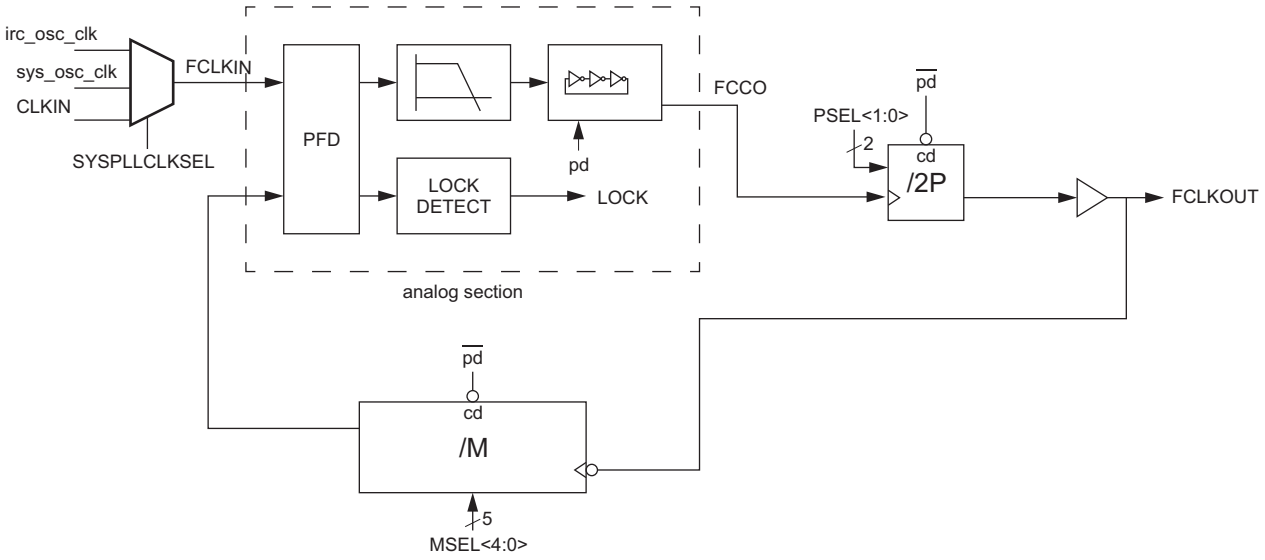


Fig 7. System PLL block diagram

The block diagram of this PLL is shown in Figure 7. The input frequency range is 10 MHz to 25 MHz. The input clock is fed directly to the Phase-Frequency Detector (PFD). This block compares the phase and frequency of its inputs, and generates a control signal when phase and/ or frequency do not match. The loop filter filters these control signals and drives the current controlled oscillator (CCO), which generates the main clock and optionally two additional phases. The CCO frequency range is 156 MHz to 320 MHz. These clocks are either divided by 2xP by the programmable post divider to create the output clocks, or are sent directly to the outputs. The main output clock is then divided by M by the programmable feedback divider to generate the feedback clock. The output signal of the phase-frequency detector is also monitored by the lock detector, to signal when the PLL has locked on to the input clock.

Remark: The divider values for P and M must be selected so that the PLL output clock frequency FCLKOUT is lower than 100 MHz because the main clock is limited to a maximum frequency of 100 MHz

5.7.4.1 Lock detector

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

5.7.4.2 Power-down control

To reduce the power consumption when the PLL clock is not needed, a PLL Power-down mode has been incorporated. This mode is enabled by setting the SYSPLL_PD bit to one in the Power-down configuration register (Table 54). In this mode, the internal current

reference will be turned off, the oscillator and the phase-frequency detector will be stopped and the dividers will enter a reset state. While in PLL Power-down mode, the lock output will be low to indicate that the PLL is not in lock. When the PLL Power-down mode is terminated by setting the SYSPLL_PD bit to zero, the PLL will resume its normal operation and will make the lock signal high once it has regained lock on the input clock.

5.7.4.3 Divider ratio programming

5.7.4.3.1 Post divider

The division ratio of the post divider is controlled by the PSEL bits. The division ratio is two times the value of P selected by PSEL bits as shown in Table 24. This guarantees an output clock with a 50% duty cycle.

5.7.4.3.2 Feedback divider

The feedback divider's division ratio is controlled by the MSEL bits. The division ratio between the PLL's output clock and the input clock is the decimal value on MSEL bits plus one, as specified in Table 24.

5.7.4.3.3 Changing the divider values

Changing the divider ratio while the PLL is running is not recommended. As there is no way to synchronize the change of the MSEL and PSEL values with the dividers, the risk exists that the counter will read in an undefined value, which could lead to unwanted spikes or drops in the frequency of the output clock. The recommended way of changing between divider settings is to power down the PLL, adjust the divider settings and then let the PLL start up again.

5.7.4.4 Frequency selection

The PLL frequency equations use the following parameters (also see Figure 7):

Table 56. PLL frequency parameters

Parameter	System PLL
FCLKIN	Frequency of sys_pllclkin (input clock to the system PLL) from the SYSPLLCLKSEL multiplexer (see Section 5.6.9).
FCCO	Frequency of the Current Controlled Oscillator (CCO); 156 to 320 MHz.
FCLKOUT	Frequency of sys_pllclkout. This is the PLL output frequency and must be < 100 MHz.
P	System PLL post divider ratio; PSEL bits in SYSPLLCTRL (see Section 5.6.3).
M	System PLL feedback divider register; MSEL bits in SYSPLLCTRL (see Section 5.6.3).

5.7.4.4.1 Normal mode

In this mode the post divider is enabled, giving a 50% duty cycle clock with the following frequency relations:

$$Fclkout = M \times Fclkin = (FCCO)/(2 \times P) \quad (1)$$

To select the appropriate values for M and P, it is recommended to follow these steps:

1. Specify the input clock frequency Fclkin.
2. Calculate M to obtain the desired output frequency Fclkout with $M = F_{clkout} / F_{clkin}$.
3. Find a value so that $FCCO = 2 \times P \times F_{clkout}$.
4. Verify that all frequencies and divider values conform to the limits specified in Table 24.

Remark: The divider values for P and M must be selected so that the PLL output clock frequency FCLKOUT is lower than 100 MHz.

Table 57 shows how to configure the PLL for a 12 MHz crystal oscillator using the SYSPLLCTRL register (Table 24). The main clock is equivalent to the system clock if the system clock divider SYSAHBCLKDIV is set to one (see Table 34).

Table 57. PLL configuration examples

PLL input clock sys_pllclkin (Fclkin)	Main clock (Fclkout)	MSEL bits Table 24	M divider value	PSEL bits Table 24	P divider value	FCCO frequency	SYSAHBCLKDIV	System clock
12 MHz	60 MHz	00100 (binary)	5	01 (binary)	2	240 MHz	2	30 MHz
12 MHz	24 MHz	00001(binary)	2	10 (binary)	4	192 MHz	1	24 MHz

5.7.4.4.2 PLL Power-down mode

In this mode, the internal current reference will be turned off, the oscillator and the phase-frequency detector will be stopped and the dividers will enter a reset state. While in PLL Power-down mode, the lock output will be low, to indicate that the PLL is not in lock. When the PLL Power-down mode is terminated by SYSPLL_PD bit to zero in the Power-down configuration register (Table 54), the PLL will resume its normal operation and will make the lock signal high once it has regained lock on the input clock.

6.1 How to read this chapter

The LPC82x provides an on-chip API in the boot ROM to optimize power consumption in active and sleep modes. See Section 26.1.

Read this chapter to configure the reduced power modes Deep-sleep mode, Power-down mode, and Deep power-down mode.

Remark: The external clock input WKTCLKIN on the wake-up timer is not available on the TSSOP20 package.

6.2 Features

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

6.3 Basic configuration

The PMU is always on as long as V_{DD} is present.

If using the WAKEUP function, disable the hysteresis for the WAKEUP pad in the DPDCTRL register when the supply voltage VDD is below 2.2 V. See Table 63.

If using the WKTCLKIN function, disable the hysteresis for that pin in the DPDCTRL register. See Table 63.

6.3.1 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.

6.3.1.1 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 allows to put the ARM core into sleep mode or the entire system in Deep-sleep or Power-down mode. To set the low power state with SLEEPDEEP = 1 to either deep-sleep or power-down or to enter the Deep power-down mode, use the PCON register (Table 61).

Table 58. 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: 0 = do not sleep when returning to Thread mode. 1 = enter sleep, or deep sleep, on return from an ISR to Thread mode. Setting this bit to 1 enables an interrupt driven application to avoid returning to an empty main application.	0
2	SLEEPDEEP	Controls whether the processor uses sleep or deep-sleep as its low power mode: 0 = sleep 1 = deep sleep.	0
3	-	Reserved.	0
4	SEVONPEND	Send Event on Pending bit: 0 = only enabled interrupts or events can wake-up the processor, disabled interrupts are excluded 1 = enabled events and all interrupts, including disabled interrupts, can wake up the processor. 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. The processor also wakes up on execution of an `SEV` instruction.	0
31:5	-	Reserved.	0

6.4 Pin description

In Deep power-down only the WAKEUP pin PIO0_4 and the self-wake-up timer clock input WKTCLKIN on pin PIO0_28 are functional (if enabled). The WAKEUP function can be disabled in the DPDCTRL register to lower the power consumption even more. In this case, enable the self-wake-up timer to provide an internal wake-up signal. See Section 6.6.3 “Deep power-down control register”.

Remark: When entering Deep power-down mode, an external pull-up resistor is required on the WAKEUP pin to hold it HIGH. In addition, pull the RESET pin HIGH to prevent it from floating while in Deep power-down mode.

6.5 General description

Power on the LPC800 is controlled by the PMU, by 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 IRC to provide a clock signal that can be shut down cleanly.

   a. In Deep-sleep mode, the peripherals receive no internal clocks. The flash is in standby mode. The 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.

   b. In Power-down mode, the peripherals receive no internal clocks. The internal SRAM memory and all peripheral registers as well as the processor maintain their internal states. The flash memory is powered down. The WWDT, WKT, and BOD can remain active to wake up the system on an interrupt.

3. Deep power-down mode:

   For maximal power savings, the entire system is shut down except for the general purpose registers in the PMU and the self-wake-up timer. Only the general purpose registers in the PMU maintain their internal states. The part can wake up on a pulse on the WAKEUP pin or when the self-wake-up timer times out. On wake-up, the part reboots.

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

6.5.1 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 53 “Wake-up configuration register (PDAWAKECFG, address 0x4004 8234) bit description”.
- To use external interrupts for wake-up: Table 50 “Start logic 0 pin wake-up enable register 0 (STARTERP0, address 0x4004 8204) bit description” and Table 49 “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 51 “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 59 “Wake-up sources for reduced power modes”

Table 59. 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. • Enable interrupt in BODCTRL register. • BOD powered in PDSLEEPCFG register.
	BOD reset	• Enable reset in BODCTRL register. • BOD powered in PDSLEEPCFG register.
	WWDT interrupt	• Enable interrupt in NVIC and STARTERP1 registers. • WWDT running. Enable WWDT in WWDT MOD register and feed. • Enable interrupt in WWDT MOD register. • WDOsc powered in PDSLEEPCFG register.
	WWDT reset	• WWDT running. • Enable reset in WWDT MOD register. • WDOsc powered in PDSLEEPCFG register.
	Self-Wake-up Timer (WKT) time-out	• Enable interrupt in NVIC and STARTERP1 registers. • Enable low-power oscillator in the DPDCTRL register in the PCON block. • Select low-power clock for WKT clock in the WKT CTRL register. • 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. • Enable USART/I2C/SPI interrupts. • Provide an external clock signal to the peripheral. • Configure the USART in synchronous slave mode and I2C and SPI in slave mode.
Deep power-down	WAKEUP pin PIO0_4	Enable the WAKEUP function in the DPDCTRL register in the PMU.
	WKT time-out	• Enable the low-power oscillator in the DPDCTRL register in the PMU. • Enable the low-power oscillator to keep running in Deep power-down mode in the DPDCTRL register in the PMU. • Select low-power clock for WKT clock in the WKT CTRL register. • Start WKT by writing a time-out value to the WKT COUNT register.

6.6 Register description

Table 60. Register overview: PMU (base address 0x4002 0000)

Name	Access	Address offset	Description	Reset value	Reference
PCON	R/W	0x000	Power control register	0x0	Table 61
GPREG0	R/W	0x004	General purpose register 0	0x0	Table 62
GPREG1	R/W	0x008	General purpose register 1	0x0	Table 62
GPREG2	R/W	0x00C	General purpose register 2	0x0	Table 62
GPREG3	R/W	0x010	General purpose register 3	0x0	Table 62
DPDCTRL	R/W	0x014	Deep power-down control register. Also includes bits for general purpose storage.	0x0	Table 63

6.6.1 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 61. Power control register (PCON, address 0x4002 0000) bit description

Bit	Symbol	Value	Description	Reset value
2:0	PM		Power mode	000
		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. ARM WFI will enter Deep-power down mode (ARM Cortex-M0+ core powered-down).	
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. 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. Write: No effect.	
		1	Low power mode. Read: Sleep, Deep-sleep or Power-down mode entered. 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. Write: No effect.	0
		1	Deep power-down. Read: Deep power-down mode entered. Write: Clear the Deep power-down flag.	
31:12	-	-	Reserved. Do not write ones to this bit.	0

6.6.2 General purpose registers 0 to 3

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 62. General purpose registers 0 to 3 (GPREG[0:3], address 0x4002 0004 (GPREG0) to 0x4002 0010 (GPREG3)) bit description

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

6.6.3 Deep power-down control register

The Deep power-down control register controls the low-power oscillator that can be used by the self-wake-up timer to wake up from Deep power-down mode. In addition, this register configures the functionality of the WAKEUP pin (pin PIO0_4).

The bits in the register not used for deep power-down control (bits 31:4) can be used for storing additional data which are retained in Deep power-down mode in the same way as registers GPREG0 to GPREG3.

Remark: If there is a possibility that the external voltage applied on pin V_{DD} drops below 2.2 V during Deep power-down, the hysteresis of the WAKEUP input pin has to be disabled in this register before entering Deep power-down mode in order for the chip to wake up.

Remark: Enabling the low-power oscillator in Deep power-down mode increases the power consumption. Only enable this oscillator if you need the self-wake-up timer to wake up the part from Deep power-down mode. You may need the self-wake-up timer if the wake-up pin is used for other purposes and the wake-up function is not available.

Table 63. Deep power down control register (DPDCTRL, address 0x4002 0014) bit description

Bit	Symbol	Value	Description	Reset value
0	WAKEUPHYS		WAKEUP pin hysteresis enable	0
		0	Disabled. Hysteresis for WAKEUP pin disabled.	
		1	Enabled. Hysteresis for WAKEUP pin enabled.	
1	WAKEPAD_ DISABLE		WAKEUP pin disable. Setting this bit disables the wake-up pin, so it can be used for other purposes. **Remark:** Never set this bit if you intend to use a pin to wake up the part from Deep power-down mode. You can only disable the wake-up pin if the self-wake-up timer is enabled and configured. **Remark:** Setting this bit is not necessary if Deep power-down mode is not used.	0
		0	Enabled. The wake-up function is enabled on pin PIO0_4.	
		1	Disabled. Setting this bit disables the wake-up function on pin PIO0_4.	
2	LPOSCEN		Enable the low-power oscillator for use with the 10 kHz self-wake-up timer clock. You must set this bit if the CLKSEL bit in the self-wake-up timer CTRL bit is set. Do not enable the low-power oscillator if the self-wake-up timer is clocked by the divided IRC or the external clock input.	0
		0	Disabled.	
		1	Enabled.	

Table 63. Deep power down control register (DPDCTRL, address 0x4002 0014) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
3	LPOSCDPDEN		Enable the low-power oscillator in Deep power-down mode. Setting this bit causes the low-power oscillator to remain running during Deep power-down mode provided that bit 2 in this register is set as well. You must set this bit for the self-wake-up timer to be able to wake up the part from Deep power-down mode. **Remark:** Do not set this bit unless you use the self-wake-up timer with the low-power oscillator clock source to wake up from Deep power-down mode.	0
		0	Disabled.	
		1	Enabled.	
4	WAKEUPCLKHYS		External clock input for the self-wake-up timer WKTCLKIN hysteresis enable.	0
		0	Disabled. Hysteresis for WAKEUP clock pin disabled.	
		1	Enabled. Hysteresis for WAKEUP clock pin enabled.	
5	WAKECLKPAD_DISABLE		Disable the external clock input for the self-wake-up timer. Setting this bit enables the self-wake-up timer clock pin WKTCLKLIN. To minimize power consumption, especially in deep power-down mode, disable this clock input when not using the external clock option for the self-wake-up timer.	0
		0	Disabled. Setting this bit disables external clock input on pin PIO0_28.	
		1	Enabled. The external clock input for the self-wake-up timer is enabled on pin PIO0_28.	
31:6	-		Data retained during Deep power-down mode.	0x0

6.7 Functional description

6.7.1 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 64. Peripheral configuration in reduced power modes

Peripheral	Sleep mode	Deep-sleep mode	Power-down mode	Deep power-down mode
IRC	software configurable	on	off	off
IRC output	software configurable	off	off	off
Flash	software configurable	on	off	off
BOD	software configurable	software configurable	software configurable	off
PLL	software configurable	off	off	off
SysOsc	software configurable	off	off	off

Table 64. Peripheral configuration in reduced power modes

Peripheral	Sleep mode	Deep-sleep mode	Power-down mode	Deep power-down mode
WDosc/WWDT	software configurable	software configurable	software configurable	off
Digital peripherals	software configurable	off	off	off
WKT/low-power oscillator	software configurable	software configurable	software configurable	software configurable

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

6.7.2 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 watchdog 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”.

6.7.3 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.

6.7.3.1 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 35).
- The power to various analog blocks (PLL, oscillators, the BOD circuit, and the flash block) can be controlled at any time individually through the PDRUNCFG register (Table 54 “Power configuration register (PDRUNCFG, address 0x4004 8238) bit description”).
- The clock source for the system clock can be selected from the IRC (default), the system oscillator, or the watchdog oscillator (see Figure 5 and related registers).
- The system clock frequency can be selected by the SYSPLLCTRL (Table 24) and the SYSAHBCLKDIV register (Table 34).
- The USART and CLKOUT use individual peripheral clocks with their own clock dividers. The peripheral clocks can be shut down through the corresponding clock divider registers.

6.7.4 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.

6.7.4.1 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.

6.7.4.2 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 58).
3. Use the ARM Cortex-M0+ Wait-For-Interrupt (WFI) instruction.

6.7.4.3 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.

6.7.5 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 watchdog 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 watchdog oscillator is selected. The IRC 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.

6.7.5.1 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 52) register:

- The watchdog 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.

6.7.5.2 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 61).
2. Select the power configuration in Deep-sleep mode in the PDSLEEPCFG (Table 52) register.
3. Select the power configuration after wake-up in the PDAWAKECFG (Table 53) 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 50, Table 51) and in the NVIC.
5. Select the IRC as the main clock. See Table 32.
6. Write one to the SLEEPDEEP bit in the ARM Cortex-M0+ SCR register (Table 58).
7. Use the ARM WFI instruction.

6.7.5.3 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 49. Each pin interrupt must also be enabled in the STARTERP0 register (Table 50) 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 51). 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 45).
- WWDT signal, if the watchdog oscillator is enabled in the PDSLEEPCFG register:
  - WWDT interrupt using the interrupt wake-up register 1 (Table 51). 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 watchdog 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 I2C. See Section 15.3.3.
- Via any of the SPI blocks. See Section 14.3.1.

6.7.6 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 watchdog 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 watchdog oscillator is selected. The IRC 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.

6.7.6.1 Power configuration in Power-down mode

Power consumption in Power-down mode can be configured by the power configuration setting in the PDSLEEPCFG (Table 52) register in the same way as for Deep-sleep mode (see Section 6.7.5.1):

- The watchdog 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.

6.7.6.2 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 61).
2. Select the power configuration in Power-down mode in the PDSLEEPCFG (Table 52) register.
3. Select the power configuration after wake-up in the PDAWAKECFG (Table 53) 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 50, Table 51) and in the NVIC.
5. Select the IRC as the main clock. See Table 32.
6. Write one to the SLEEPDEEP bit in the ARM Cortex-M0+ SCR register (Table 58).
7. Use the ARM WFI instruction.

6.7.6.3 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 49. Each pin interrupt must also be enabled in the STARTERP0 register (Table 50) 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 51). 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 45).
- WWDT signal, if the watchdog oscillator is enabled in the PDSLEEPCFG register:

- WWDT interrupt using the interrupt wake-up register 1 (Table 51). 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 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 I2C. See Section 15.3.3.
- Via any of the SPI blocks. See Section 14.3.1.

6.7.7 Deep power-down mode

In Deep power-down mode, power and clocks are shut off to the entire chip with the exception of the WAKEUP pin and the self-wake-up timer.

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 WAKEUP pin. In this mode, you must pull the $\overline{\text{RESET}}$ pin HIGH externally.

Remark: Setting bit 3 in the PCON register (Table 61) prevents the part from entering Deep-power down mode.

6.7.7.1 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 WAKEUP pin and the self-wake-up timer are powered.

6.7.7.2 Programming Deep power-down mode using the WAKEUP pin:

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

1. Pull the WAKEUP pin externally HIGH.
2. Ensure that bit 3 in the PCON register (Table 61) is cleared.
3. Write 0x3 to the PM bits in the PCON register (see Table 61).
4. Store data to be retained in the general purpose registers (Section 6.6.2).
5. Write one to the SLEEPDEEP bit in the ARM Cortex-M0+ SCR register (Table 58).
6. Use the ARM WFI instruction.

6.7.7.3 Wake-up from Deep power-down mode using the WAKEUP pin:

Pulling the WAKEUP pin LOW wakes up the LPC800 from Deep power-down, and the part goes through the entire reset process.

1. On the WAKEUP pin, 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 DPDCTRL and GPREG0 to GPREG3 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 61) 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 61).
4. (Optional) Read the stored data in the general purpose registers (Section 6.6.2).
5. Set up the PMU for the next Deep power-down cycle.

Remark: The $\overline{\text{RESET}}$ pin has no functionality in Deep power-down mode.

6.7.7.4 Programming Deep power-down mode using the self-wake-up timer:

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

1. Enable the low-power oscillator to run in Deep power-down mode by setting bits 2 and 3 in the DPDCTRL register to 1 (see Table 63)
2. Ensure that bit 3 in the PCON register (Table 61) is cleared.
3. Write 0x3 to the PM bits in the PCON register (see Table 61).
4. Store data to be retained in the general purpose registers (Section 6.6.2).
5. Write one to the SLEEPDEEP bit in the ARM Cortex-M0+ SCR register.
6. Start the self-wake-up timer by writing a value to the WKT COUNT register (Table 262).
7. Use the ARM WFI instruction.

6.7.7.5 Wake-up from Deep power-down mode using the self-wake-up timer:

The part goes through the entire reset process when the self-wake-up timer times out:

1. When the WKT count reaches 0, the following happens:
   - 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 DPDCTRL and GPREG0 to GPREG3 registers and PCON are in their reset state.
2. Once the chip has booted, read the deep power-down flag in the PCON register (Table 61) 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 61).
4. (Optional) Read the stored data in the general purpose registers (Section 6.6.2).
5. Set up the PMU for the next Deep power-down cycle.

Remark: The $\overline{\text{RESET}}$ pin has no functionality in Deep power-down mode.

7.1 How to read this chapter

The switch matrix is identical for all LPC82x parts.

7.2 Features

- Flexible assignment of digital peripheral functions to pins
- Enable/disable of analog functions

7.3 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.

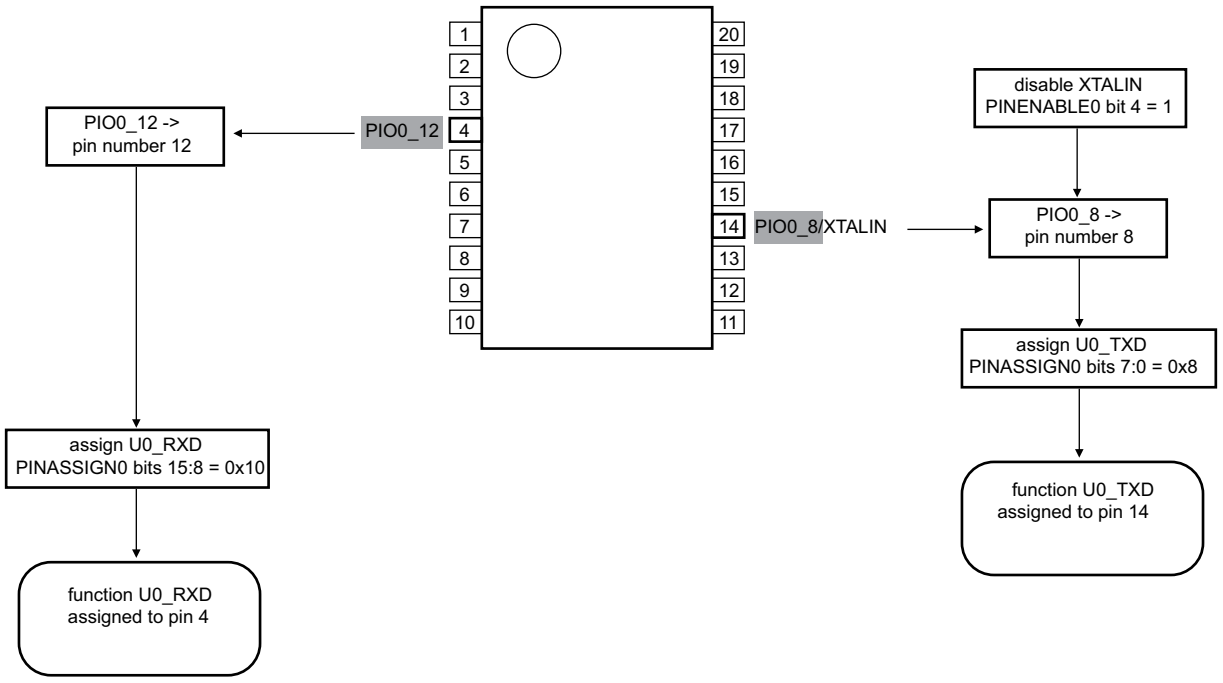
Before activating a peripheral or enabling its interrupt, use the switch matrix to connect the peripheral to external pins.

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.

7.3.1 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 8. 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 65 or in the data sheet.
2. Use the LPC800 data sheet to decide which pin x on the LPC800 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 register.
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.

7.3.2 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 LPC800 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.

7.3.3 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.

7.4 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, digital filter, 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 crystal oscillator pins (XTALIN/XTALOUT) or 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.

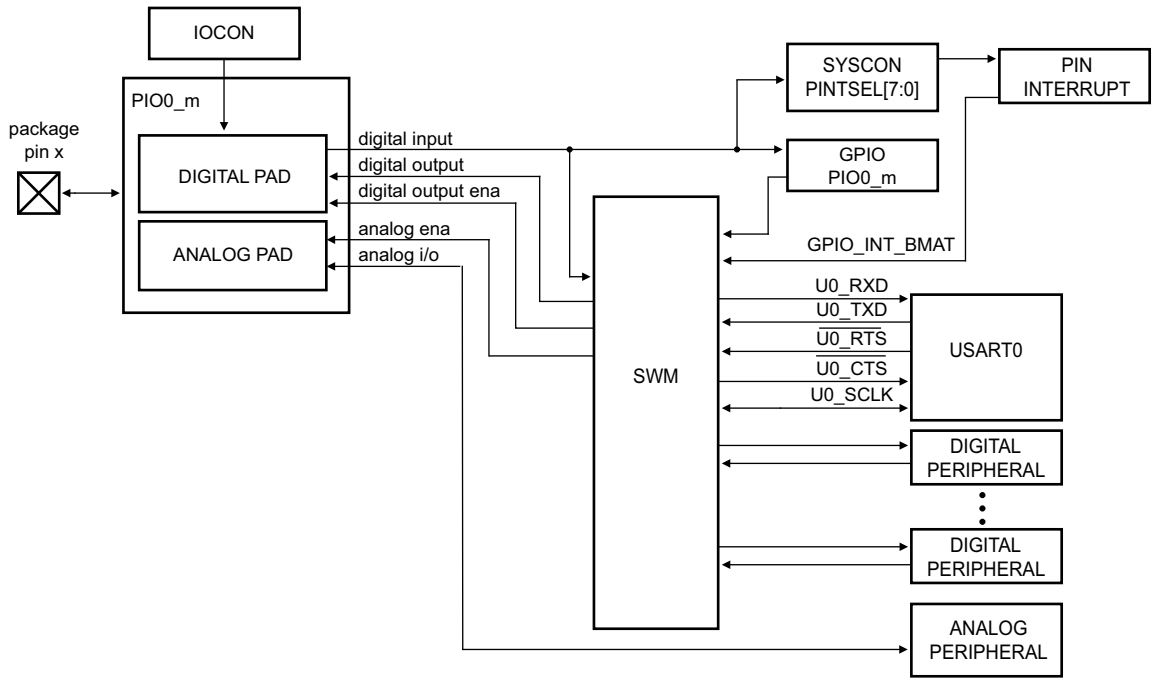


Fig 9. Functional diagram of the switch matrix

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 9). 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.

  Example:

  You can enable the CLKIN input in the PINENABLE0 register on pin PIO0_1 and also assign one ore more SCT inputs to pin PIO0_1 through the PINASSIGN registers to feed the CLKIN into the SCT.

  You can send the input on one pin to all SCT inputs to use as an SCT abort signal.
- 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 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.

7.4.1 Movable functions

Table 65. Movable functions (assign to pins PIO0_0 to PIO0_28 through switch matrix)

Function name	Type	Description	SWM Pin assign register	Reference
U0_TXD	O	Transmitter output for USART0.	PINASSIGN0	Table 67
U0_RXD	I	Receiver input for USART0.	PINASSIGN0	Table 67
$\overline{\text{U0\_RTS}}$	O	Request To Send output for USART0.	PINASSIGN0	Table 67
$\overline{\text{U0\_CTS}}$	I	Clear To Send input for USART0.	PINASSIGN0	Table 67
U0_SCLK	I/O	Serial clock input/output for USART0 in synchronous mode.	PINASSIGN1	Table 68
U1_TXD	O	Transmitter output for USART1.	PINASSIGN1	Table 68
U1_RXD	I	Receiver input for USART1.	PINASSIGN1	Table 68
$\overline{\text{U1\_RTS}}$	O	Request To Send output for USART1.	PINASSIGN1	Table 68
$\overline{\text{U1\_CTS}}$	I	Clear To Send input for USART1.	PINASSIGN2	Table 69
U1_SCLK	I/O	Serial clock input/output for USART1 in synchronous mode.	PINASSIGN2	Table 69
U2_TXD	O	Transmitter output for USART2.	PINASSIGN2	Table 69
U2_RXD	I	Receiver input for USART2.	PINASSIGN2	Table 69
$\overline{\text{U2\_RTS}}$	O	Request To Send output for USART1.	PINASSIGN3	Table 70
$\overline{\text{U2\_CTS}}$	I	Clear To Send input for USART1.	PINASSIGN3	Table 70
U2_SCLK	I/O	Serial clock input/output for USART1 in synchronous mode.	PINASSIGN3	Table 70
SPI0_SCK	I/O	Serial clock for SPI0.	PINASSIGN3	Table 70
SPI0_MOSI	I/O	Master Out Slave In for SPI0.	PINASSIGN4	Table 71
SPI0_MISO	I/O	Master In Slave Out for SPI0.	PINASSIGN4	Table 71
SPI0_SSEL0	I/O	Slave select 0 for SPI0.	PINASSIGN4	Table 71
SPI0_SSEL1	I/O	Slave select 0 for SPI1.	PINASSIGN4	Table 71
SPI0_SSEL2	I/O	Slave select 0 for SPI2.	PINASSIGN5	Table 72
SPI0_SSEL3	I/O	Slave select 0 for SPI3.	PINASSIGN5	Table 72
SPI1_SCK	I/O	Serial clock for SPI1.	PINASSIGN5	Table 72
SPI1_MOSI	I/O	Master Out Slave In for SPI1.	PINASSIGN5	Table 72
SPI1_MISO	I/O	Master In Slave Out for SPI1.	PINASSIGN6	Table 73
SPI1_SSEL0	I/O	Slave select 0 for SPI1.	PINASSIGN6	Table 73
SPI1_SSEL1	I/O	Slave select 1 for SPI1.	PINASSIGN6	Table 73
SCT_PIN0	I	Pin input 0 to the SCT input multiplexer.	PINASSIGN6	Table 73
SCT_PIN1	I	Pin input 1 to the SCT input multiplexer.	PINASSIGN7	Table 74
SCT_PIN2	I	Pin input 2 to the SCT input multiplexer.	PINASSIGN7	Table 74
SCT_PIN3	I	Pin input 3 to the SCT input multiplexer.	PINASSIGN7	Table 74
SCT_OUT0	O	SCT output 0.	PINASSIGN7	Table 74
SCT_OUT1	O	SCT output 1.	PINASSIGN8	Table 75

Table 65. Movable functions (assign to pins PIO0_0 to PIO0_28 through switch matrix)

Function name	Type	Description	SWM Pin assign register	Reference
SCT_OUT2	O	SCT output 2.	PINASSIGN8	Table 75
SCT_OUT3	O	SCT output 3.	PINASSIGN8	Table 75
SCT_OUT4	O	SCT output 4.	PINASSIGN8	Table 75
SCT_OUT5	O	SCT output 5.	PINASSIGN9	Table 76
I2C1_SDA	I/O	I^2C1-bus data input/output.	PINASSIGN9	Table 76
I2C1_SCL	I/O	I^2C1-bus clock input/output.	PINASSIGN9	Table 76
I2C2_SDA	I/O	I^2C2-bus data input/output.	PINASSIGN9	Table 76
I2C2_SCL	I/O	I^2C2-bus clock input/output.	PINASSIGN10	Table 77
I2C3_SDA	I/O	I^2C3-bus data input/output.	PINASSIGN10	Table 77
I2C3_SCL	I/O	I^2C3-bus clock input/output.	PINASSIGN10	Table 77
ADC_PINTRIG0	I	ADC external pin trigger input 0.	PINASSIGN10	Table 77
ADC_PINTRIG1	I	ADC external pin trigger input 1.	PINASSIGN11	Table 78
ACMP_O	O	Analog comparator output.	PINASSIGN11	Table 78
CLKOUT	O	Clock output.	PINASSIGN11	Table 78
GPIO_INT_BMAT	O	Output of the pattern match engine.	PINASSIGN11	Table 78

7.4.2 Switch matrix register interface

The switch matrix consists of two blocks of pin-assignment registers PINASSIGN and PINENABLE. Every function has an assigned field (1-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.

GPIOs range from PIO0_0 to PIO0_28 and, for assignment through the pin-assignment registers, are numbered 0 to 28.

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

1. **Movable functions** (PINASSIGN0 to 11):

   All movable functions are digital functions. Assign movable functions to pin numbers through the 8 bits of the PINASSIGN 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 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.
- If a fixed-pin analog function is selected, its assigned pin cannot be used for any other function.

7.5 Register description

Table 66. 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 67
PINASSIGN1	R/W	0x004	Pin assign register 1. Assign movable functions U0_SCLK, U1_TXD, U1_RXD, U1_RTS.	0xFFFF FFFF	Table 68
PINASSIGN2	R/W	0x008	Pin assign register 2. Assign movable functions U1_CTS, U1_SCLK, U2_TXD, U2_RXD.	0xFFFF FFFF	Table 69
PINASSIGN3	R/W	0x00C	Pin assign register 3. Assign movable function U2_RTS, U2_CTS, U2_SCLK, SPI0_SCK.	0xFFFF FFFF	Table 70
PINASSIGN4	R/W	0x010	Pin assign register 4. Assign movable functions SPI0_MOSI, SPI0_MISO, SPI0_SSEL0, SPI0_SSEL1.	0xFFFF FFFF	Table 71
PINASSIGN5	R/W	0x014	Pin assign register 5. Assign movable functions SPI0_SSEL2, SPI0_SSEL3, SPI1_SCK, SPI1_MOSI	0xFFFF FFFF	Table 72
PINASSIGN6	R/W	0x018	Pin assign register 6. Assign movable functions SPI1_MISO, SPI1_SSEL0, SPI1_SSEL1, SCT0_IN0.	0xFFFF FFFF	Table 73
PINASSIGN7	R/W	0x01C	Pin assign register 7. Assign movable functions SCT_IN1, SCT_IN2, SCT_IN3, SCT_OUT0.	0xFFFF FFFF	Table 74
PINASSIGN8	R/W	0x020	Pin assign register 8. Assign movable functions SCT_OUT1, SCT_OUT2, SCT_OUT3, SCT_OUT4.	0xFFFF FFFF	Table 75
PINASSIGN9	R/W	0x024	Pin assign register 9. Assign movable functions SCT_OUT5, I2C1_SDA, I2C1_SCL, I2C2_SDA.	0xFFFF FFFF	Table 76

Table 66. Register overview: Switch matrix (base address 0x4000 C000) *…continued*

Name	Access	Offset	Description	Reset value	Reference
PINASSIGN10	R/W	0x028	Pin assign register 10. Assign movable functions I2C2_SCL, I2C3_SDA, I2C3_SCL, ADC_PINTRIG0.	0xFFFF FFFF	Table 77
PINASSIGN11	R/W	0x02C	Pin assign register 11. Assign movable functions ADC_PINTRIG1, ACMP_O, CLKOUT, GPIO_INT_BMAT	0xFFFF FFFF	Table 78
PINENABLE0	R/W	0x1C0	Pin enable register 0. Enables fixed-pin functions ACMP_I0, ACMP_I1, SWCLK, SWDIO, XTALIN, XTALOUT, RESET, CLKIN, VDDCMP.	0xFFFF FECF	Table 79

7.5.1 Pin assign register 0

Table 67. 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: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	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: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	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: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	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: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF

7.5.2 Pin assign register 1

Table 68. 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: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	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: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	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: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
31:24	U1_RTS_O	U1_RTS function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF

7.5.3 Pin assign register 2

Table 69. Pin assign register 2 (PINASSIGN2, address 0x4000 C008) bit description

Bit	Symbol	Description	Reset value
7:0	U1_CTS_I	U1_CTS function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
15:8	U1_SCLK_IO	U1_SCLK function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
23:16	U2_TXD_O	U2_TXD function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
31:24	U2_RXD_I	U2_RXD function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF

7.5.4 Pin assign register 3

Table 70. Pin assign register 3 (PINASSIGN3, address 0x4000 C00C) bit description

Bit	Symbol	Description	Reset value
7:0	U2_RTS_O	U2_RTS function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
15:8	U2_CTS_I	U2_CTS function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
23:16	U2_SCLK_IO	U2_SCLK function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
31:24	SPI0_SCK_IO	SPI0_SCK function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF

7.5.5 Pin assign register 4

Table 71. Pin assign register 4 (PINASSIGN4, address 0x4000 C010) bit description

Bit	Symbol	Description	Reset value
7:0	SPI0_MOSI_IO	SPI0_MOSI function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF

Table 71. Pin assign register 4 (PINASSIGN4, address 0x4000 C010) bit description

Bit	Symbol	Description	Reset value
15:8	SPI0_MISO_IO	SPI0_MISIO function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
23:16	SPI0_SSEL0_IO	SPI0_SSEL0 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
31:24	SPI0_SSEL1_IO	SPI0_SSEL1 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF

7.5.6 Pin assign register 5

Table 72. Pin assign register 5 (PINASSIGN5, address 0x4000 C014) bit description

Bit	Symbol	Description	Reset value
7:0	SPI0_SSEL2_IO	SPI0_SSEL2 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
15:8	SPI0_SSEL3_IO	SPI0_SSEL3 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
23:16	SPI1_SCK_IO	SPI1_SCK function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
31:24	SPI1_MOSI_IO	SPI1_MOSI function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF

7.5.7 Pin assign register 6

Table 73. Pin assign register 6 (PINASSIGN6, address 0x4000 C018) bit description

Bit	Symbol	Description	Reset value
7:0	SPI1_MISO_IO	SPI1_MISO function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
15:8	SPI1_SSEL0_IO	SPI1_SSEL0 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
23:16	SPI1_SSEL1_IO	SPI1_SSEL1 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
31:24	SCT_PIN0_I	SCT_PIN0 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF

7.5.8 Pin assign register 7

Table 74. Pin assign register 7 (PINASSIGN7, address 0x4000 C01C) bit description

Bit	Symbol	Description	Reset value
7:0	SCT_PIN1_I	SCT_PIN1 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
15:8	SCT_PIN2_I	SCT_PIN2 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
23:16	SCT_PIN3_I	SCT_PIN3 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
31:24	SCT_OUT0_O	SCT_OUT0 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF

7.5.9 Pin assign register 8

Table 75. Pin assign register 8 (PINASSIGN8, address 0x4000 C020) bit description

Bit	Symbol	Description	Reset value
7:0	SCT_OUT1_O	SCT_OUT1 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
15:8	SCT_OUT2_O	SCT_OUT2 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
23:16	SCT_OUT3_O	SCT_OUT3 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
31:24	SCT_OUT4_O	SCT_OUT4 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF

7.5.10 Pin assign register 9

Table 76. Pin assign register 9 (PINASSIGN9, address 0x4000 C024) bit description

Bit	Symbol	Description	Reset value
7:0	SCT_OUT5_O	SCT_OUT5 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF

Table 76. Pin assign register 9 (PINASSIGN9, address 0x4000 C024) 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: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	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: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
31:24	I2C2_SDA_IO	I2C1_SDA function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF

7.5.11 Pin assign register 10

Table 77. Pin assign register 10 (PINASSIGN10, address 0x4000 C028) bit description

Bit	Symbol	Description	Reset value
7:0	I2C2_SCL_IO	I2C1_SCL function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
15:8	I2C3_SDA_IO	I2C3_SDA function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
23:16	I2C3_SCL_IO	I2C3_SCL function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
31:24	ADC_PINTRIG0_I	ADC_PINTRIG0 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF

7.5.12 Pin assign register 11

Table 78. Pin assign register 11 (PINASSIGN11, address 0x4000 C02C) bit description

Bit	Symbol	Description	Reset value
7:0	ADC_PINTRIG1_I	ADC_PINTRIG1 function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
15:8	ACMP_O_O	ACMP_O function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
23:16	CLKOUT_O	CLKOUT function assignment. The value is the pin number to be assigned to this function. The following pins are available: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF
31:24	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: PIO0_0 (= 0) to PIO0_28 (= 0x1C).	0xFF

7.5.13 PINENABLE 0

Table 79. 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_23.	
		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	XTALIN		XTALIN function select.	1
		0	XTALIN enabled on pin PIO0_8.	
		1	XTALIN disabled.	
7	XTALOUT		XTALOUT function select.	1
		0	XTALOUT enabled on pin PIO0_9.	
		1	XTALOUT disabled.	
8	RESETN		RESETN function select.	0
		0	RESETN enabled on pin PIO0_5.	
		1	RESETN disabled.	
9	CLKIN		CLKIN function select.	1
		0	CLKIN enabled on pin PIO0_1.	
		1	CLKIN disabled.	
10	VDDCMP		VDDCMP function select.	1
		0	VDDCMP enabled on pin PIO0_6.	
		1	VDDCMP disabled.	
11	I2C0_SDA		I2C0_SDA function select.	1
		0	I2C0_SDA enabled on pin PIO0_11.	
		1	I2C0_SDA disabled.	
12	I2C0_SCL		I2C0_SCL function select.	1
		0	I2C0_SCL enabled on pin PIO0_10.	
		1	I2C0_SCL disabled.	

Table 79. Pin enable register 0 (PINENABLE0, address 0x4000 C1C0) bit description

Bit	Symbol	Value	Description	Reset value
13	ADC_0		ADC_0 function select.	1
		0	ADC_0 enabled on pin PIO0_7.	
		1	ADC_0 disabled.	
14	ADC_1		ADC_1 function select.	1
		0	ADC_1 enabled on pin PIO0_6.	
		1	ADC_1 disabled.	
15	ADC_2		ADC_2 function select.	1
		0	ADC_2 enabled on pin PIO0_14.	
		1	ADC_2 disabled.	
16	ADC_3		ADC_3 function select.	1
		0	ADC_3 enabled on pin PIO0_23.	
		1	ADC_3 disabled.	
17	ADC_4		ADC_4 function select.	1
		0	ADC_4 enabled on pin PIO0_22.	
		1	ADC_4 disabled.	
18	ADC_5		ADC_5 function select.	1
		0	ADC_5 enabled on pin PIO0_21.	
		1	ADC_5 disabled.	
19	ADC_6		ADC_6 function select.	1
		0	ADC_6 enabled on pin PIO0_20.	
		1	ADC_6 disabled.	
20	ADC_7		ADC_7 function select.	1
		0	ADC_7 enabled on pin PIO0_19.	
		1	ADC_7 disabled.	
21	ADC_8		ADC_8 function select.	1
		0	ADC_8 enabled on pin PIO0_18.	
		1	ADC_8 disabled.	
22	ADC_9		ADC_9 function select.	1
		0	ADC_9 enabled on pin PIO0_17.	
		1	ADC_9 disabled.	
23	ADC_10		ADC_10 function select.	1
		0	ADC_10 enabled on pin PIO0_13.	
		1	ADC_10 disabled.	
24	ADC_11		ADC_11 function select.	1
		0	ADC_11 enabled on pin PIO0_4.	
		1	ADC_11 disabled.	
31:25	-		Reserved.	1

8.1 How to read this chapter

The IOCON block is identical for all LPC82x parts. Registers for pins that are not available on a specific package are reserved.

Table 80. Pinout summary

Package	Pins/configuration registers available
TSSOP20	PIO0_0 to PIO0_5; PIO0_8 to PIO0_15; PIO0_17; PIO0_23
HVQFN33	PIO0_0 to PIO0_28

8.2 Features

The following electrical properties are configurable for each pin:

- Pull-up/pull-down resistor
- Open-drain mode
- Hysteresis
- Digital glitch filter with programmable time constant
- Analog mode (for a subset of pins, see the LPC82x data sheet)

The true open-drain pins PIO0_10 and PIO0_11 can be configured for different I2C-bus speeds.

8.3 Basic configuration

Enable the clock to the IOCON in the SYSAHBCLKCTRL register (Table 35, bit 18). Once the pins are configured, you can disable the IOCON clock to conserve power.

Remark: If the open-drain pins PIO0_10 and PIO0_11 are not available on the package, prevent the pins from internally floating as follows: Set bits 10 and 11 in the GPIO DIR0 register to 1 to enable the output driver and write 1 to bits 10 and 11 in the GPIO CLR0 register to drive the outputs LOW internally.

8.4 General description

8.4.1 Pin configuration

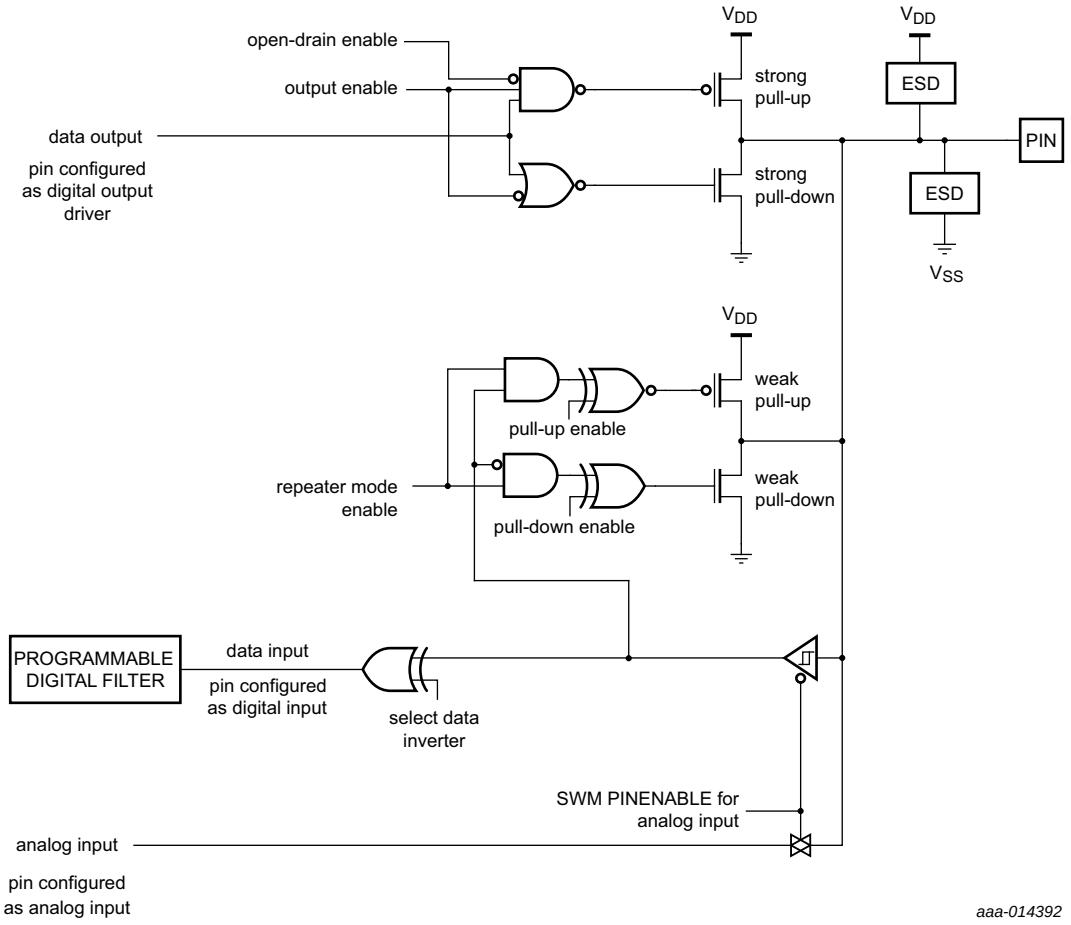


Fig 10. Pin configuration

8.4.2 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 65 “Movable functions (assign to pins PIO0_0 to PIO0_28 through switch matrix)”

8.4.3 Pin mode

The MODE bit in the IOCON register allows enabling or disabling an on-chip pull-up resistor for each pin. By default all pull-up resistors are enabled except for the I^2C-bus pins PIO0_10 and PIO0_11, which do not have a programmable pull-up resistor.

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.

8.4.4 Open-drain mode

An open-drain mode can be enabled for all digital I/O pins that are not the I2C-bus 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 $V_{DD} = 0$.

8.4.5 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.

8.4.6 I^2C-bus mode

The I^2C-bus pins PIO0_10 and PIO0_11 can be programmed to support a true open-drain mode independently of whether the I2C function is selected or another digital function. If the I^2C function is selected, all three I^2C modes, Standard mode, Fast-mode, and Fast-mode plus, are supported. A digital glitch filter can be configured for all functions. Pins PIO0_10 and PIO0_11 operate as high-current sink drivers (20 mA) independently of the programmed function.

Remark: Pins PIO0_10 and PIO0_11 are 5 V tolerant when $V_{DD} = 0$ and when V_{DD} is at operating voltage level.

8.4.7 Programmable digital filter

All GPIO pins are equipped with a programmable, digital glitch filter. The filter rejects input pulses with a selectable duration of shorter than one, two, or three cycles of a filter clock (S_MODE = 1, 2, or 3). For each individual pin, the filter clock can be selected from one of seven peripheral clocks PCLK0 to 6, which are derived from the main clock using the IOCONCLKDIV0 to 6 registers. The filter can also be bypassed entirely.

Any input pulses of duration T_{pulse} of either polarity will be rejected if:
$T_{pulse} < T_{PCLKn} \times S\_MODE$

Input pulses of one filter clock cycle longer may also be rejected:

$T_{pulse} = T_{PCLKn} \times (S\_MODE + 1)$

Remark: The filtering effect is accomplished by requiring that the input signal be stable for (S_MODE +1) successive edges of the filter clock before being passed on to the chip. Enabling the filter results in delaying the signal to the internal logic and should be done only if specifically required by an application. For high-speed or time critical functions ensure that the filter is bypassed.

If the delay of the input signal must be minimized, select a faster PCLK and a higher sample mode (S_MODE) to minimize the effect of the potential extra clock cycle.

If the sensitivity to noise spikes must be minimized, select a slower PCLK and lower sample mode.

Related registers and links:

Table 44 “IOCON glitch filter clock divider registers 6 to 0 (IOCONCLKDIV[6:0], address 0x4004 8134 (IOCONCLKDIV6) to 0x004 814C (IOCONFILTCLKDIV0)) bit description”

8.5 Register description

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

Remark: See Table 82 for the IOCON register map ordered by pin name.

Table 81. 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 0090	Table 83
PIO0_13	R/W	0x004	I/O configuration for pin PIO0_13/ADC_10	0x0000 0090	Table 84
PIO0_12	R/W	0x008	I/O configuration for pin PIO0_12	0x0000 0090	Table 85
PIO0_5	R/W	0x00C	I/O configuration for pin PIO0_5/$\overline{\text{RESET}}$	0x0000 0090	Table 86
PIO0_4	R/W	0x010	I/O configuration for pin PIO0_4/ADC_11/TRSTN/WAKEUP	0x0000 0090	Table 87
PIO0_3	R/W	0x014	I/O configuration for pin PIO0_3/SWCLK	0x0000 0090	Table 88
PIO0_2	R/W	0x018	I/O configuration for pin PIO0_2/SWDIO	0x0000 0090	Table 89
PIO0_11	R/W	0x01C	I/O configuration for pin PIO0_11. This is the pin configuration for the true open-drain pin.	0x0000 0080	Table 90
PIO0_10	R/W	0x020	I/O configuration for pin PIO0_10. This is the pin configuration for the true open-drain pin.	0x0000 0080	Table 91
PIO0_16	R/W	0x024	I/O configuration for pin PIO0_16	0x0000 0090	Table 92
PIO0_15	R/W	0x028	I/O configuration for pin PIO0_15	0x0000 0090	Table 93
PIO0_1	R/W	0x02C	I/O configuration for pin PIO0_1/ACMP_I1/CLKIN	0x0000 0090	Table 94
-	-	0x030	Reserved	-	-
PIO0_9	R/W	0x034	I/O configuration for pin PIO0_9/XTALOUT	0x0000 0090	Table 95
PIO0_8	R/W	0x038	I/O configuration for pin PIO0_8/XTALIN	0x0000 0090	Table 96
PIO0_7	R/W	0x03C	I/O configuration for pin PIO0_7/ADC_0	0x0000 0090	Table 97
PIO0_6	R/W	0x040	I/O configuration for pin PIO0_6/ADC_1/VDDCMP	0x0000 0090	Table 98
PIO0_0	R/W	0x044	I/O configuration for pin PIO0_0/ACMP_I0	0x0000 0090	Table 99

Table 81. Register overview: I/O configuration (base address 0x4004 4000)

Name	Access	Address offset	Description	Reset value	Reference
PIO0_14	R/W	0x048	I/O configuration for pin PIO0_14/ ACMP_I3/ADC_2	0x0000 0090	Table 100
-	-	0x04C	Reserved.	-	-
PIO0_28	R/W	0x050	I/O configuration for pin PIO0_28	0x0000 0090	Table 101
PIO0_27	R/W	0x054	I/O configuration for pin PIO0_27	0x0000 0090	Table 102
PIO0_26	R/W	0x058	I/O configuration for pin PIO0_26	0x0000 0090	Table 103
PIO0_25	R/W	0x05C	I/O configuration for pin PIO0_25	0x0000 0090	Table 104
PIO0_24	R/W	0x060	I/O configuration for pin PIO0_24	0x0000 0090	Table 105
PIO0_23	R/W	0x064	I/O configuration for pin PIO0_23/ADC_3/ACMP_I4	0x0000 0090	Table 106
PIO0_22	R/W	0x068	I/O configuration for pin PIO0_22/ADC_4	0x0000 0090	Table 107
PIO0_21	R/W	0x06C	I/O configuration for pin PIO0_21/ADC_5	0x0000 0090	Table 108
PIO0_20	R/W	0x070	I/O configuration for pin PIO0_20/ADC_6	0x0000 0090	Table 109
PIO0_19	R/W	0x074	I/O configuration for pin PIO0_19/ADC_7	0x0000 0090	Table 110
PIO0_18	R/W	0x078	I/O configuration for pin PIO0_18/ADC_8	0x0000 0090	Table 111

Table 82. I/O configuration registers ordered by pin name

Name	Address offset	True open-drain	Analog[1]	Digital filter	High-drive output	Reference
PIO0_0	0x044	no	yes	yes	no	Table 99
PIO0_1	0x02C	no	yes	yes	no	Table 94
PIO0_2	0x018	no	no	yes	yes	Table 89
PIO0_3	0x014	no	no	yes	yes	Table 88
PIO0_4	0x010	no	yes	yes	no	Table 87
PIO0_5	0x00C	no	no	yes	no	Table 86
PIO0_6	0x040	no	yes	yes	no	Table 98
PIO0_7	0x03C	no	yes	yes	no	Table 97
PIO0_8	0x038	no	yes	yes	no	Table 96
PIO0_9	0x034	no	yes	yes	no	Table 95
PIO0_10	0x020	yes	no	yes	no	Table 91
PIO0_11	0x01C	yes	no	yes	no	Table 90
PIO0_12	0x008	no	no	yes	yes	Table 85
PIO0_13	0x004	no	yes	yes	no	Table 84
PIO0_14	0x048	no	yes	yes	no	Table 100
PIO0_15	0x028	no	no	yes	no	Table 93
PIO0_16	0x024	no	no	yes	yes	Table 92
PIO0_17	0x000	no	yes	yes	no	Table 83

Table 82. I/O configuration registers ordered by pin name

Name	Address offset	True open-drain	Analog[1]	Digital filter	High-drive output	Reference
PIO0_18	0x078	no	yes	yes	no	Table 111
PIO0_19	0x074	no	yes	yes	no	Table 110
PIO0_20	0x070	no	yes	yes	no	Table 109
PIO0_21	0x06C	no	yes	yes	no	Table 108
PIO0_22	0x068	no	yes	yes	no	Table 107
PIO0_23	0x064	no	yes	yes	no	Table 106
PIO0_24	0x060	no	no	yes	no	Table 105
PIO0_25	0x05C	no	no	yes	no	Table 104
PIO0_26	0x058	no	no	yes	no	Table 103
PIO0_27	0x054	no	no	yes	no	Table 102
PIO0_28	0x050	no	no	yes	no	Table 101

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

8.5.1 PIO0_17 register

Table 83. 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.	0
		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. **Remark:** This is not a true open-drain mode.	

Table 83. PIO0_17 register (PIO0_17, address 0x4004 4000) bit description

Bit	Symbol	Value	Description	Reset value
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.2 PIO0_13 register

Table 84. 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.	0
		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 84. PIO0_13 register (PIO0_13, address 0x4004 4004) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.3 PIO0_12 register

Table 85. 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.	0
		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).	

Table 85. PIO0_12 register (PIO0_12, address 0x4004 4008) bit description *...continued*

Bit	Symbol	Value	Description	Reset value
9:7	-	-	Reserved.	0b001
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.4 PIO0_5 register

Table 86. 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.	0
		0	Disable.	
		1	Enable.	

Table 86. PIO0_5 register (PIO0_5, address 0x4004 400C) 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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.5 PIO0_4 register

Table 87. 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 87. PIO0_4 register (PIO0_4, address 0x4004 4010) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
5	HYS		Hysteresis.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.6 PIO0_3 register

Table 88. PIO0_3 register (PIO0_3, address 0x4004 4014) bit description

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

Table 88. PIO0_3 register (PIO0_3, address 0x4004 4014) 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.7 PIO0_2 register

Table 89. 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.8 PIO0_11 register

Table 90. PIO0_11 register (PIO0_11, address 0x4004 401C) bit description

Bit	Symbol	Value	Description	Reset value
5:0	-		Reserved.	0
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).	
7	-		Reserved.	1
9:8	I2CMODE		Selects I2C mode. Select Standard mode (I2CMODE = 00, default) or Standard I/O functionality (I2CMODE = 01) if the pin function is GPIO.	00
		0x0	Standard mode/ Fast-mode I2C.	
		0x1	Standard GPIO functionality. Requires external pull-up for GPIO output function.	
		0x2	Fast-mode Plus I2C	
		0x3	Reserved.	
10	-	-	Reserved.	-
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	-

8.5.9 PIO0_10 register

Table 91. PIO0_10 register (PIO0_10, address 0x4004 4020) bit description

Bit	Symbol	Value	Description	Reset value
5:0	-		Reserved.	0
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).	
7	-		Reserved.	1
9:8	I2CMODE		Selects I2C mode. Select Standard mode (I2CMODE = 00, default) or Standard I/O functionality (I2CMODE = 01) if the pin function is GPIO.	00
		0x0	Standard mode/ Fast-mode I2C.	
		0x1	Standard GPIO functionality. Requires external pull-up for GPIO output function.	
		0x2	Fast-mode Plus I2C	
		0x3	Reserved.	
10	-	-	Reserved.	-
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	-

8.5.10 PIO0_16 register

Table 92. 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.11 PIO0_15 register

Table 93. 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.12 PIO0_1 register

Table 94. 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.13 PIO0_9 register

Table 95. 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).	10
		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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.14 PIO0_8 register

Table 96. PIO0_8 register (PIO0_8, address 0x4004 4038) 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.15 PIO0_7 register

Table 97. 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.16 PIO0_6 register

Table 98. PIO0_6 register (PIO0_6, 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).	10
		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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.17 PIO0_0 register

Table 99. 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.18 PIO0_14 register

Table 100. 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.19 PIO0_28 register

Table 101. PIO0_28 register (PIO0_28, 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.20 PIO0_27 register

Table 102. PIO0_27 register (PIO0_27, 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.21 PIO0_26 register

Table 103. PIO0_26 register (PIO0_26, 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.22 PIO0_25 register

Table 104. PIO0_25 register (PIO0_25, 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.23 PIO0_24 register

Table 105. PIO0_24 register (PIO0_24, 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.24 PIO0_23 register

Table 106. PIO0_23 register (PIO0_23, address 0x4004 4064) 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.25 PIO0_22 register

Table 107. PIO0_22 register (PIO0_22, 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.26 PIO0_21 register

Table 108. PIO0_21 register (PIO0_21, 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.27 PIO0_20 register

Table 109. PIO0_20 register (PIO0_20, 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.28 PIO0_19 register

Table 110. PIO0_19 register (PIO0_19, 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

8.5.29 PIO0_18 register

Table 111. PIO0_18 register (PIO0_18, address 0x4004 4078) 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.	0
		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. **Remark:** This is not a true open-drain mode.	
12:11	S_MODE		Digital filter sample mode.	0
		0x0	Bypass input filter.	
		0x1	1 clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	2 clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	3 clock cycles. Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved.	0
		0x0	IOCONCLKDIV0.	
		0x1	IOCONCLKDIV1.	
		0x2	IOCONCLKDIV2.	
		0x3	IOCONCLKDIV3.	
		0x4	IOCONCLKDIV4.	
		0x5	IOCONCLKDIV5.	
		0x6	IOCONCLKDIV6.	
31:16	-	-	Reserved.	0

9.1 How to read this chapter

All GPIO registers refer to 32 pins on each port. Depending on the package type, not all pins are available, and the corresponding bits in the GPIO registers are reserved.

Table 112. GPIO pins available

Package	GPIO Port 0
TSSOP20	PIO0_0 to PIO0_5; PIO0_8 to PIO0_15; PIO0_17; PIO0_23
HVQFN33	PIO0_0 to PIO0_28

9.2 Basic configuration

For the GPIO port registers, enable the clock to the GPIO port in the SYSAHBCLKCTRL register (Table 35).

9.3 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.

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

9.5 Register description

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

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

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

Table 113. Register overview: GPIO port (base address 0xA000 0000)

Name	Access	Address offset	Description	Reset value	Width	Reference
B0 to B28	R/W	0x0000 to 0x001C	Byte pin registers port 0; pins PIO0_0 to PIO0_28	ext	byte (8 bit)	Table 114
W0 to W28	R/W	0x1000 to 0x1074	Word pin registers port 0	ext	word (32 bit)	Table 115
DIR0	R/W	0x2000	Direction registers port 0	0	word (32 bit)	Table 116
MASK0	R/W	0x2080	Mask register port 0	0	word (32 bit)	Table 117
PIN0	R/W	0x2100	Port pin register port 0	ext	word (32 bit)	Table 118
MPIN0	R/W	0x2180	Masked port register port 0	ext	word (32 bit)	Table 119
SET0	R/W	0x2200	Write: Set register for port 0 Read: output bits for port 0	0	word (32 bit)	Table 120
CLR0	WO	0x2280	Clear port 0	-	word (32 bit)	Table 121
NOT0	WO	0x2300	Toggle port 0	-	word (32 bit)	Table 122
DIRSET0	WO	0x2380	Set pin direction bits for port 0.	0	word (32 bit)	Table 123
DIRCLR0	WO	0x2400	Clear pin direction bits for port 0.	-	word (32 bit)	Table 124
DIRNOT0	WO	0x2480	Toggle pin direction bits for port 0.	-	word (32 bit)	Table 125

9.5.1 GPIO port byte pin registers

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

Table 114. GPIO port byte pin registers (B[0:28], addresses 0xA000 0000 (B0) to 0xA000 001C (B28)) bit description

Bit	Symbol	Description	Reset value	Access
0	PBYTE	Read: state of the pin PIO0_n, regardless of direction, masking, or alternate function, except that pins configured as analog I/O always read as 0. One register for each port pin: n = pin 0 to 28. Write: loads the pin's output bit.	ext	R/W
7:1		Reserved (0 on read, ignored on write)	0	-

9.5.2 GPIO port word pin registers

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

Table 115. GPIO port word pin registers (W[0:28], addresses 0xA000 1000 (W0) to 0xA000 1074 (W28)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	PWORD	Read 0: pin PIOm_n is LOW. Write 0: clear output bit. Read 0xFFFF FFFF: pin PIOm_n is HIGH. Write any value 0x0000 0001 to 0xFFFF FFFF: set output bit. **Remark:** Only 0 or 0xFFFF FFFF can be read. Writing any value other than 0 will set the output bit. One register for each port pin: n = pin 0 to 28.	ext	R/W

9.5.3 GPIO port direction registers

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

Table 116. GPIO direction port register (DIR0, address 0xA000 2000) bit description

Bit	Symbol	Description	Reset value	Access
28:0	DIRP	Selects pin direction for pin PIO0_n (bit 0 = PIO0_0, bit 1 = PIO0_1, ..., bit 28 = PIO0_28). 0 = input. 1 = output.	0	R/W
31:29	-	Reserved.	0	-

9.5.4 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 117. GPIO mask port register (MASK0, address 0xA000 2080) bit description

Bit	Symbol	Description	Reset value	Access
28:0	MASKP	Controls which bits corresponding to PIO0_n are active in the MPORT register (bit 0 = PIO0_0, bit 1 = PIO0_1, ..., bit 28 = PIO0_28). 0 = Read MPORT: pin state; write MPORT: load output bit. 1 = Read MPORT: 0; write MPORT: output bit not affected.	0	R/W
31:29	-	Reserved.	0	-

9.5.5 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 118. GPIO port pin register (PIN0, address 0xA000 2100) bit description

Bit	Symbol	Description	Reset value	Access
28:0	PORT	Reads pin states or loads output bits (bit 0 = PIO0_0, bit 1 = PIO0_1, ..., bit 38 = PIO0_28). 0 = Read: pin is low; write: clear output bit. 1 = Read: pin is high; write: set output bit.	ext	R/W
31:29	-	Reserved.	0	-

9.5.6 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 119. GPIO masked port pin register (MPIN0, address 0xA000 2180) bit description

Bit	Symbol	Description	Reset value	Access
28:0	MPORTP	Masked port register (bit 0 = PIO0_0, bit 1 =PIO0_1, ..., bit 28 = PIO0_28). 0 = Read: pin is LOW and/or the corresponding bit in the MASK register is 1; write: clear output bit if the corresponding bit in the MASK register is 0. 1 = Read: pin is HIGH and the corresponding bit in the MASK register is 0; write: set output bit if the corresponding bit in the MASK register is 0.	ext	R/W
31:29	-	Reserved.	0	-

9.5.7 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 120. GPIO port set register (SET0, address 0xA000 2200) bit description

Bit	Symbol	Description	Reset value	Access
28:0	SETP	Read or set output bits (bit 0 = PIO0_0, bit 1 =PIO0_1, ..., bit 28 = PIO0_28). 0 = Read: output bit: write: no operation. 1 = Read: output bit; write: set output bit.	0	R/W
31:29	-	Reserved.	0	-

9.5.8 GPIO port clear registers

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

Table 121. GPIO port clear register (CLR0, address 0xA000 2280) bit description

Bit	Symbol	Description	Reset value	Access
28:0	CLRP	Clear output bits (bit 0 = PIO0_0, bit 1 =PIO0_1, ..., bit 28 = PIO0_28). 0 = No operation. 1 = Clear output bit.	NA	WO
31:29	-	Reserved.	0	-

9.5.9 GPIO port toggle registers

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

Table 122. GPIO port toggle register (NOT0, address 0xA000 2300) bit description

Bit	Symbol	Description	Reset value	Access
28:0	NOTP	Toggle output bits (bit 0 = PIO0_0, bit 1 =PIO0_1, ..., bit 28 = PIO0_28). 0 = no operation. 1 = Toggle output bit.	NA	WO
31:29	-	Reserved.	0	-

9.5.10 GPIO port direction set registers

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

Table 123. GPIO port direction set register (DIRSET0, address 0xA000 2380) bit description

Bit	Symbol	Description	Reset value	Access
28:0	DIRSETP	Set direction bits (bit 0 = PIO0_0, bit 1 = PIO0_1, ..., bit 28 = PIO0_28). 0 = No operation. 1 = Set direction bit.	0	WO
31:29	-	Reserved.	0	-

9.5.11 GPIO port direction clear registers

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

Table 124. GPIO port direction clear register (DIRCLR0, 0xA000 2400) bit description

Bit	Symbol	Description	Reset value	Access
28:0	DIRCLRP	Clear direction bits (bit 0 = PIO0_0, bit 1 =PIO0_1, ..., bit 28 = PIO0_28). 0 = No operation. 1 = Clear direction bit.	NA	WO
31:29	-	Reserved.	0	-

9.5.12 GPIO port direction toggle registers

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

Table 125. GPIO port direction toggle register (DIRNOT0, address 0xA000 2480) bit description

Bit	Symbol	Description	Reset value	Access
28:0	DIRNOTP	Toggle direction bits (bit 0 = PIO0_0, bit 1 =PIO0_1, ..., bit 28 = PIO0_28). 0 = no operation. 1 = Toggle direction bit.	NA	WO
31:29	-	Reserved.	0	-

9.6 Functional description

9.6.1 Reading pin state

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

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

9.6.2 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 9.6.1 returns the state of pins, regardless of their direction or alternate functions.

9.6.3 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.

9.6.4 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.

9.6.5 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.

10.1 How to read this chapter

The pin interrupt generator and the pattern match engine are available on all LPC82x parts.

10.2 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.

10.3 Basic configuration

- Pin interrupts:
  - Select up to eight external interrupt pins from all GPIO port pins in the SYSCON block (Table 49). 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 35, bit 6).
  - 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 50).
  - 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 49). 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 35, bit 6).
- 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 (PINASSIGN11, Table 78).

10.3.1 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 LPC800 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 5.6.28 “Pin interrupt select registers” in the SYSCON block for the PINTSEL registers.

10.4 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 5.6.28 “Pin interrupt select registers”.

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

See Section 7.3.1 “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 126. Pin interrupt/pattern match engine pin description

Function	Direction	Pin	Description	SWM register	Reference
GPIO_INT_BMAT	O	any	GPIO pattern match output	PINASSIGN11	Table 78

10.5 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.

10.5.1 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 49). The external interrupt pins are connected to eight individual interrupts in the NVIC and are created based on rising or falling edges or on the input level on the pin.

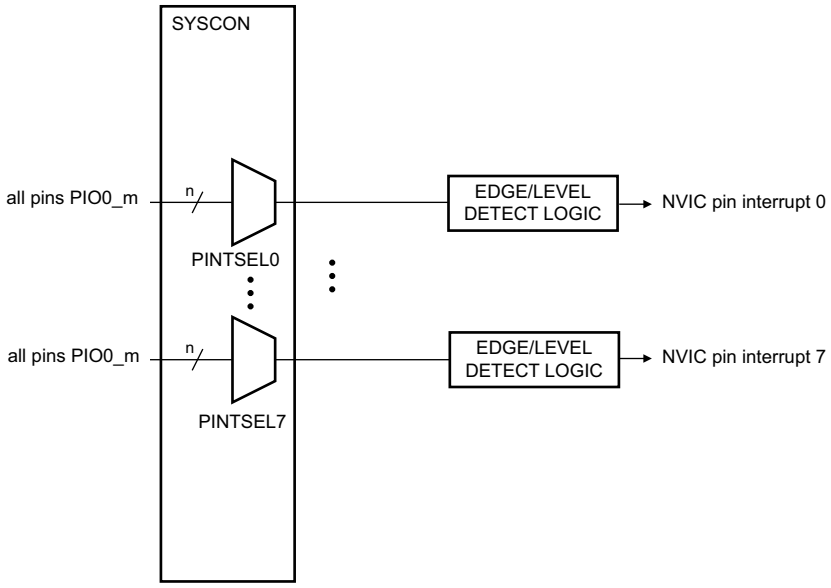
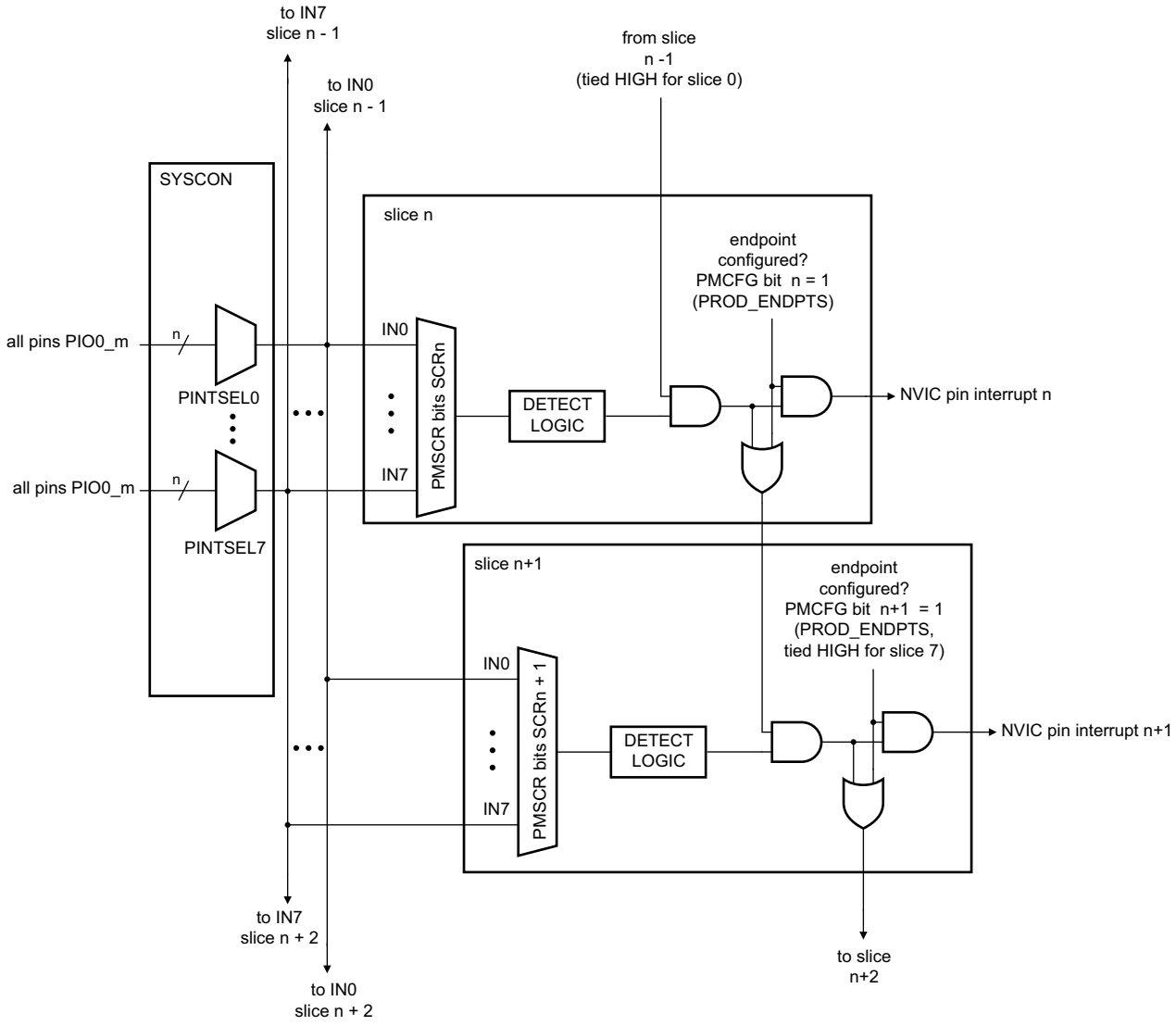


Fig 11. Pin interrupt connections

10.5.2 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 13 for the detect logic block.

Fig 12. 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 13 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 10.7.3 for details.

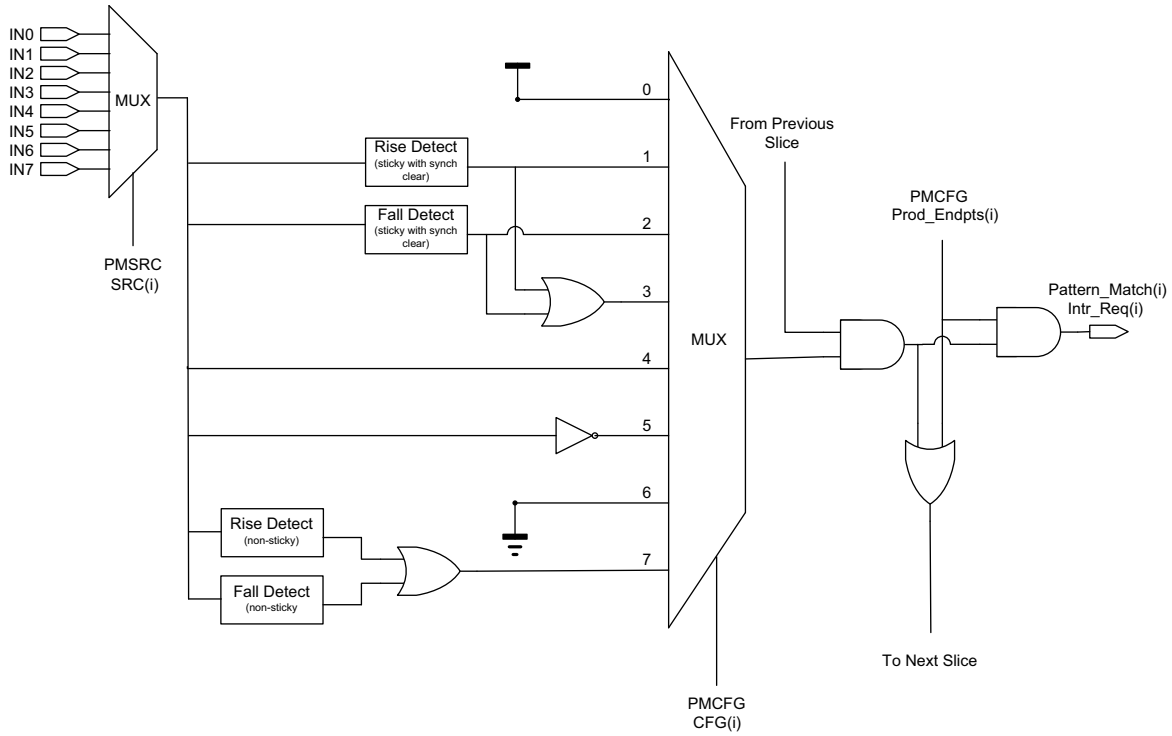


Fig 13. Pattern match bit slice with detect logic

10.5.2.1 Inputs and outputs of the pattern match engine

The connections between the pins and the pattern match engine are shown in Figure 12. 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 in order to use the pins for wake-up.

10.5.2.2 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 13).

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

Example:

Assume the expression: (IN0)~(IN1)(IN3)^ + (IN1)(IN2) + (IN0)~(IN3)~(IN4) is specified through the registers PMSRC (Table 139) and PMCFG (Table 140). 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 10.7.2

10.6 Register description

Table 127. 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 128
IENR	R/W	0x004	Pin interrupt level or rising edge interrupt enable register	0	Table 129
SIENR	WO	0x008	Pin interrupt level or rising edge interrupt set register	NA	Table 130
CIENR	WO	0x00C	Pin interrupt level (rising edge interrupt) clear register	NA	Table 131
IENF	R/W	0x010	Pin interrupt active level or falling edge interrupt enable register	0	Table 132
SIENF	WO	0x014	Pin interrupt active level or falling edge interrupt set register	NA	Table 133
CIENF	WO	0x018	Pin interrupt active level or falling edge interrupt clear register	NA	Table 134
RISE	R/W	0x01C	Pin interrupt rising edge register	0	Table 135
FALL	R/W	0x020	Pin interrupt falling edge register	0	Table 136
IST	R/W	0x024	Pin interrupt status register	0	Table 137
PMCTRL	R/W	0x028	Pattern match interrupt control register	0	Table 138
PMSRC	R/W	0x02C	Pattern match interrupt bit-slice source register	0	Table 139
PMCFG	R/W	0x030	Pattern match interrupt bit slice configuration register	0	Table 140

10.6.1 Pin interrupt mode register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 5.6.28), one bit in the ISEL register determines whether the interrupt is edge or level sensitive.

Table 128. 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. 0 = Edge sensitive 1 = Level sensitive	0	R/W
31:8	-	Reserved.	-	-

10.6.2 Pin interrupt level or rising edge interrupt enable register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 5.6.28), 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 129. 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. 0 = Disable rising edge or level interrupt. 1 = Enable rising edge or level interrupt.	0	R/W
31:8	-	Reserved.	-	-

10.6.3 Pin interrupt level or rising edge interrupt set register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 5.6.28), 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 130. 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. 0 = No operation. 1 = Enable rising edge or level interrupt.	NA	WO
31:8	-	Reserved.	-	-

10.6.4 Pin interrupt level or rising edge interrupt clear register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 5.6.28), 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 131. 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. 0 = No operation. 1 = Disable rising edge or level interrupt.	NA	WO
31:8	-	Reserved.	-	-

10.6.5 Pin interrupt active level or falling edge interrupt enable register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 5.6.28), 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 132. 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. 0 = Disable falling edge interrupt or set active interrupt level LOW. 1 = Enable falling edge interrupt enabled or set active interrupt level HIGH.	0	R/W
31:8	-	Reserved.	-	-

10.6.6 Pin interrupt active level or falling edge interrupt set register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 5.6.28), 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 133. 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. 0 = No operation. 1 = Select HIGH-active interrupt or enable falling edge interrupt.	NA	WO
31:8	-	Reserved.	-	-

10.6.7 Pin interrupt active level or falling edge interrupt clear register

For each of the 8 pin interrupts selected in the PINTSELn registers (see Section 5.6.28), 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 134. 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. 0 = No operation. 1 = LOW-active interrupt selected or falling edge interrupt disabled.	NA	WO
31:8	-	Reserved.	-	-

10.6.8 Pin interrupt rising edge register

This register contains ones for pin interrupts selected in the PINTSELn registers (see Section 5.6.28) 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 135. 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. Read 0: No rising edge has been detected on this pin since Reset or the last time a one was written to this bit. Write 0: no operation. Read 1: a rising edge has been detected since Reset or the last time a one was written to this bit. Write 1: clear rising edge detection for this pin.	0	R/W
31:8	-	Reserved.	-	-

10.6.9 Pin interrupt falling edge register

This register contains ones for pin interrupts selected in the PINTSELn registers (see Section 5.6.28) 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 136. 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. Read 0: No falling edge has been detected on this pin since Reset or the last time a one was written to this bit. Write 0: no operation. Read 1: a falling edge has been detected since Reset or the last time a one was written to this bit. Write 1: clear falling edge detection for this pin.	0	R/W
31:8	-	Reserved.	-	-

10.6.10 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 137. 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. Read 0: interrupt is not being requested for this interrupt pin. Write 0: no operation. Read 1: interrupt is being requested for this interrupt pin. Write 1 (edge-sensitive): clear rising- and falling-edge detection for this pin. Write 1 (level-sensitive): switch the active level for this pin (in the IENF register).	0	R/W
31:8	-	Reserved.	-	-

10.6.11 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 138. 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

10.6.12 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 5.6.28. 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 139. 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 139. 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 139. 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 139. 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 139. 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.	

10.6.13 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 140. 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 140. 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 140. 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 140. 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 140. 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 140. 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.	

10.7 Functional description

10.7.1 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 141.

Table 141. 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.

10.7.2 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 139):
  - 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 140):
  - 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 138):

– 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.

10.7.3 Pattern match engine edge detect examples

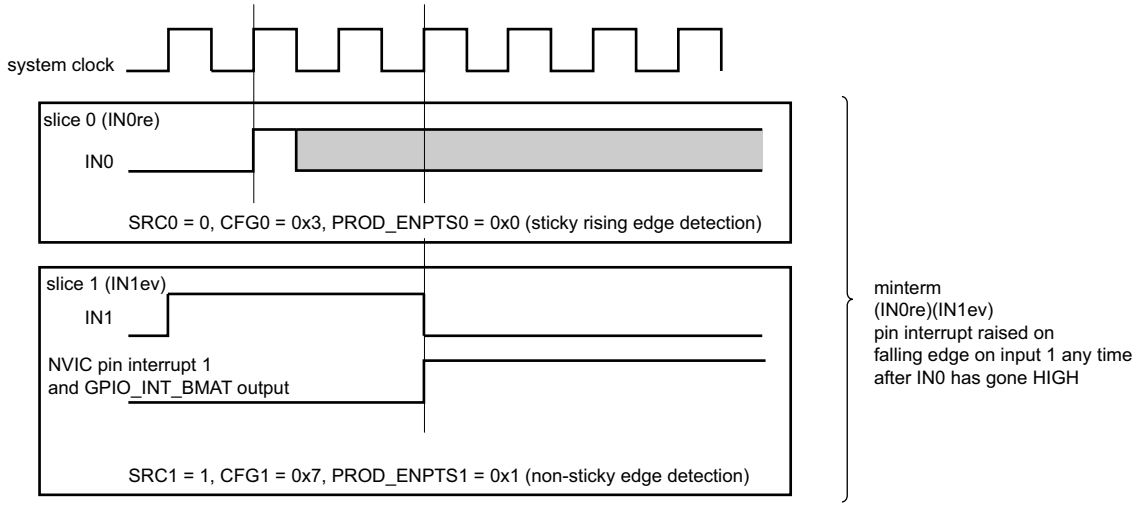


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 14. Pattern match engine examples: sticky edge detect

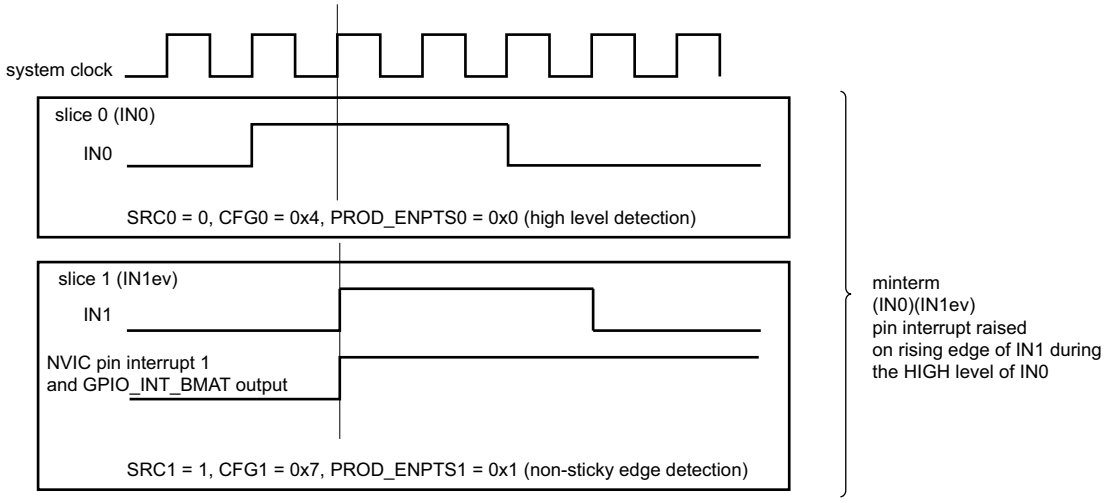


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 15. Pattern match engine examples: Windowed non-sticky edge detect evaluates as true

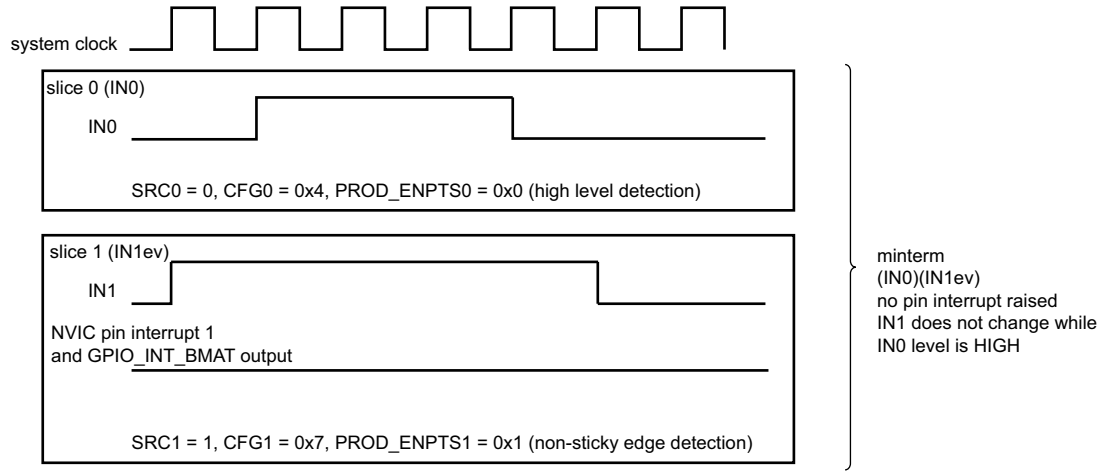


Figure shows pattern match functionality only and accurate timing is not implied. Inputs (INn) are shown synchronized to the system clock for simplicity.

Fig 16. Pattern match engine examples: Windowed non-sticky edge detect evaluates as false

11.1 How to read this chapter

SCT input multiplexing and DMA input multiplexing is available for all parts.

11.2 Features

- Configures the inputs to the SCTimer/PWM.
- Configures the inputs to the DMA triggers.

11.3 Basic configuration

- In the SYSAHBCLKCTRL register, enable the SCT clock to write to the INPUT MUX registers. See Table 35.
- In the SYSAHBCLKCTRL register, enable the DMA clock to write to the DMA TRIGMUX registers. See Table 35.

11.4 Pin description

The input multiplexer has no dedicated pins. However, external pins can be selected as inputs to the SCT input multiplexer. Multiplexer inputs from external pins are assigned through the switch matrix to pins.

Table 142. INPUT MUX pin description

Pin functions	Peripheral	Input mux reference
SCT_PIN0, SCT_PIN1, SCT_PIN2, SCT_PIN3	SCT0	Table 147

11.5 General description

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

The input multiplexing makes it possible to design complex event-driven processes without CPU intervention by connecting peripherals like the SCT and the ADC or the SCT and the analog comparator.

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

11.5.1 SCT input multiplexing

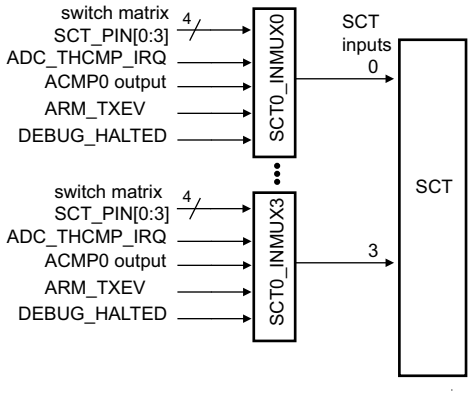


Fig 17. SCT input multiplexing

11.5.2 DMA trigger input multiplexing

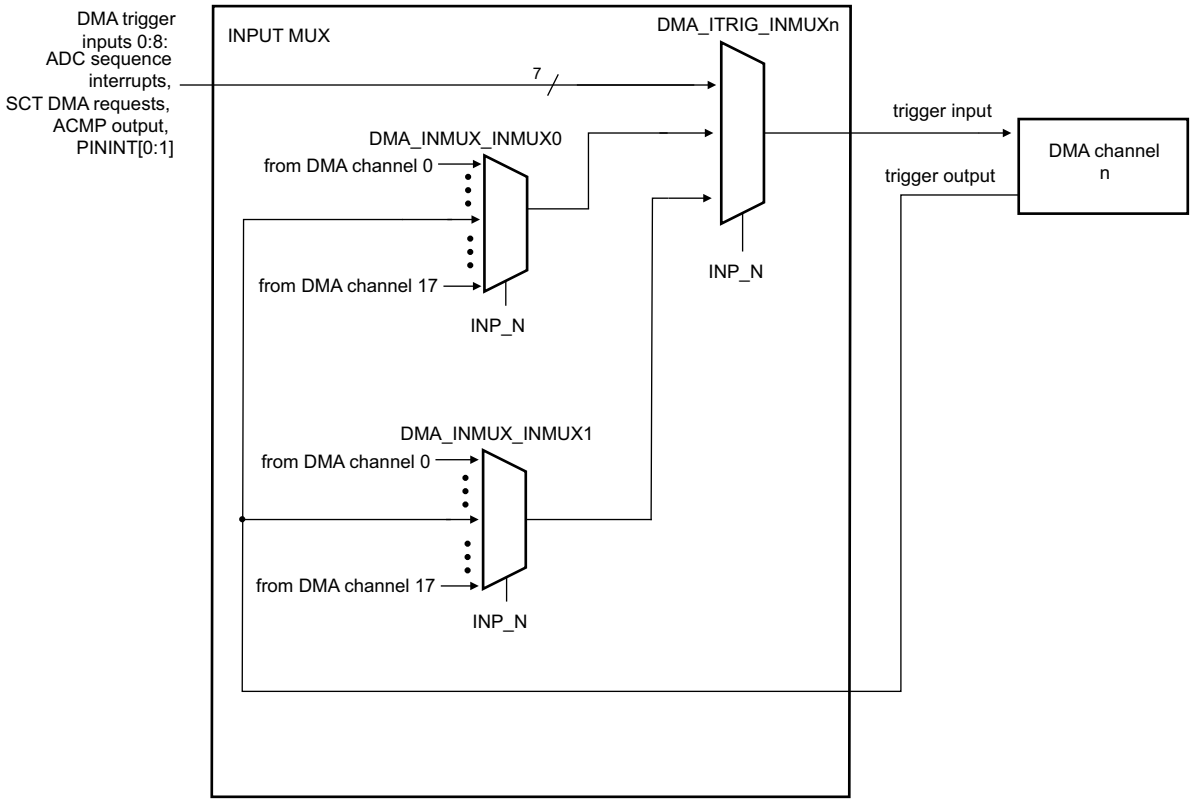


Fig 18. DMA trigger multiplexing

11.6 Register description

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

All address offsets not shown in Table 144 are reserved and should not be written to.

Table 143. Register overview: Input multiplexing (base address 0x4002 8000)

Name	Access	Offset	Description	Reset value	Reference
DMA_ITRIG_INMUX0	R/W	0x000	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145
DMA_ITRIG_INMUX1	R/W	0x004	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145
DMA_ITRIG_INMUX2	R/W	0x008	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145
DMA_ITRIG_INMUX3	R/W	0x00C	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145
DMA_ITRIG_INMUX4	R/W	0x010	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145
DMA_ITRIG_INMUX5	R/W	0x014	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145
DMA_ITRIG_INMUX6	R/W	0x018	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145
DMA_ITRIG_INMUX7	R/W	0x01C	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145
DMA_ITRIG_INMUX8	R/W	0x020	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145
DMA_ITRIG_INMUX9	R/W	0x024	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145
DMA_ITRIG_INMUX10	R/W	0x028	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145
DMA_ITRIG_INMUX11	R/W	0x02C	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145
DMA_ITRIG_INMUX12	R/W	0x030	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145
DMA_ITRIG_INMUX13	R/W	0x034	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145

Table 143. Register overview: Input multiplexing (base address 0x4002 8000) *…continued*

Name	Access	Offset	Description	Reset value	Reference
DMA_ITRIG_INMUX14	R/W	0x038	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145
DMA_ITRIG_INMUX15	R/W	0x03C	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145
DMA_ITRIG_INMUX16	R/W	0x040	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145
DMA_ITRIG_INMUX17	R/W	0x044	Input mux register for trigger inputs 0 to 23 connected to DMA channel 0. Selects from ADC, SCT, ACMP, pin interrupts, and DMA requests.	0x0F	Table 145

Table 144. Register overview: Input multiplexing (base address 0x4002 C000)

Name	Access	Offset	Description	Reset value	Reference
DMA_INMUX_INMUX0	R/W	0x000	Input mux register for DMA trigger input 20. Selects from 18 DMA trigger outputs.	0x1F	Table 146
DMA_INMUX_INMUX1	R/W	0x004	Input mux register for DMA trigger input 21. Selects from 18 DMA trigger outputs.	0x1F	Table 146
SCT0_INMUX0	R/W	0x020	Input mux register for SCT input 0	0x0F	Table 147
SCT0_INMUX1	R/W	0x024	Input mux register for SCT input 1	0x0F	Table 147
SCT0_INMUX2	R/W	0x028	Input mux register for SCT input 2	0x0F	Table 147
SCT0_INMUX3	R/W	0x02C	Input mux register for SCT input 3	0x0F	Table 147

11.6.1 DMA input trigger input mux registers 0 to 17

With the DMA input trigger input mux registers you can select one trigger input for each of the 18 DMA channels from multiple internal sources.

By default, none of the triggers are selected.

Table 145. DMA input trigger Input mux registers 0 to 17 (DMA_ITRIG_INMUX[0:17], address 0x4002 80E0 (DMA_ITRIG_INMUX0) to 0x4002 8124 (DMA_ITRIG_INMUX17)) bit description

Bit	Symbol	Value	Description	Reset value
3:0	INP		Trigger input number (decimal value) for DMA channel n (n = 0 to 8). All other values are reserved.	0x0F
		0x0	ADC_SEQA_IRQ	
		0x1	ADC_SEQB_IRQ	
		0x2	SCT_DMA0	
		0x3	SCT_DMA1	
		0x4	ACMP_O	
		0x5	PININT0	
		0x6	PININT1	

Table 145. DMA input trigger Input mux registers 0 to 17 (DMA_ITRIG_INMUX[0:17], address 0x4002 80E0 (DMA_ITRIG_INMUX0) to 0x4002 8124 (DMA_ITRIG_INMUX17)) bit description

Bit	Symbol	Value	Description	Reset value
		0x7	DMA trigger mux 0. (DMA_INMUX_INMUX0).	
		0x8	DMA trigger mux 1. (DMA_INMUX_INMUX1)	
31:4	-		Reserved.	-

11.6.2 DMA trigger input mux input registers 0 to 1

This register provides a multiplexer for inputs 7 to 8 of each DMA trigger input mux register DMA_ITRIG_INMUX. These inputs can be selected from the 18 trigger outputs generated by the DMA (one trigger output per channel).

By default, none of the triggers are selected.

Table 146. DMA input trigger input mux input registers 0 to 1 (DMA_INMUX_INMUX[0:1], address 0x4002 C000 (DMA_INMUX_INMUX0) to 0x4002 C004 (DMA_INMUX_INMUX1)) bit description

Bit	Symbol	Description	Reset value
4:0	INP	DMA trigger output number (decimal value) for DMA channel n (n = 0 to 17).	0x1F
31:5	-	Reserved.	-

11.6.3 SCT input mux registers 0 to 3

With the SCT0 Input mux registers you can select one input source for each SCT input from 8 external and internal sources.

The output of SCT Input mux register 0 selects the source for SCT0 input 0, the output of SCT0 Input mux register 1 selects the source for SCT0 input 1, and so forth up to SCT0 Input mux register 3, which selects the input for SCT0 input 3.

The value to be programmed in this register is the input number ranging from 0 for pin function SCT_IN0 to 7 for the DEBUG_HALTED signal from the ARM CoreSight debug signal.

Inputs 0 to 3 are connected to external pins through the switch matrix.

Table 147. SCT input mux registers 0 to 3 (SCT0_INMUX[0:3], address 0x4002 C020 (SCT0_INMUX0) to 0x4002 C02C (SCT0_INMUX3)) bit description

Bit	Symbol	Value	Description	Reset value
3:0	INP_N		Input number (decimal value) to SCT0 inputs 0 to 3.	0x0F
		0x0	SCT_PIN0. Assign to pin using the switch matrix.	
		0x1	SCT_PIN1. Assign to pin using the switch matrix.	
		0x2	SCT_PIN2. Assign to pin using the switch matrix.	
		0x3	SCT_PIN3. Assign to pin using the switch matrix.	
		0x4	ADC_THCMP_IRQ	
		0x5	ACMP_O	

Table 147. SCT input mux registers 0 to 3 (SCT0_INMUX[0:3], address 0x4002 C020 (SCT0_INMUX0) to 0x4002 C02C (SCT0_INMUX3)) bit description

Bit	Symbol	Value	Description	Reset value
		0x6	ARM_TXEV	
		0x7	DEBUG_HALTED	
31:4	-		Reserved.	-

12.1 How to read this chapter

The DMA controller is available on all parts.

12.2 Features

- 18 channels supported with 18 channels connected to peripheral request inputs and outputs of the USART, SPI, and I2C peripherals.
- DMA operations can be triggered by on- or off-chip events. Each DMA channel can select one trigger input from 9 sources.
- Priority is user selectable for each channel.
- Continuous priority arbitration.
- Address cache with four entries.
- Efficient use of data bus.
- Supports single transfers up to 1,024 words.
- Address increment options allow packing and/or unpacking data.

12.3 Basic configuration

Configure the DMA as follows:

- Use the SYSAHBCLKCTRL register (Table 35) to enable the clock to the DMA registers interface.
- The DMA interrupt is connected to slot #20 in the NVIC.
- Each DMA channel has one DMA request line associated and can also select one of nine input triggers through the input multiplexer registers DMA_ITRIG_INMUX[0:17].
- Trigger outputs are connected to DMA_INMUX_INMUX[0:3] as inputs to DMA triggers.

For details on the trigger input and output multiplexing, see Section 11.5.2 “DMA trigger input multiplexing”.

12.3.1 Hardware triggers

Each DMA channel can use one trigger that is independent of the request input for this channel. The trigger input is selected in the DMA_ITRIG_INMUX registers. There are 20 possible internal trigger sources for each channel with each trigger signal issued by the output of a peripheral. In addition, the DMA trigger output can be routed to the trigger input of another channel through the trigger input multiplexing. See Section 11.5.2 “DMA trigger input multiplexing”.

See Table 148 for the connection of input multiplexers to DMA channels.

See Table 145 for a list of possible trigger input sources.

12.3.2 Trigger outputs

Each channel of the DMA controller provides a trigger output. This allows the possibility of using the trigger outputs as a trigger source to a different channel in order to support complex transfers on selected peripherals. This kind of transfer can, for example, use more than one peripheral DMA request. An example use would be to input data to a holding buffer from one peripheral, and then output the data to another peripheral, with both transfers being paced by the appropriate peripheral DMA request. This kind of operation is called “chained operation” or “channel chaining”.

12.3.3 DMA requests

DMA requests are directly connected to the peripherals. Each channel supports one DMA request line and one trigger input which is multiplexed to many possible input sources.

For each trigger multiplexer DMA_ITRIG_INMUXn, the following sources are supported:

- ADC sequence A interrupt ADC_SEQA_IRQ
- ADC sequence B interrupt ADC_SEQB_IRQ
- SCT DMA request 0 SCT_DMA0
- SCT DMA request 1 SCT_DMA1
- GPIO pin interrupt 0 (PININT0)
- GPIO pin interrupt 1 (PININT1)
- Two choices of one of the DMA output triggers

Table 148. DMA requests

DMA channel #	Request input	DMA trigger multiplexer
0	USART0_RX_DMA	DMA_ITRIG_INMUX0
1	USART0_TX_DMA	DMA_ITRIG_INMUX1
2	USART1_RX_DMA	DMA_ITRIG_INMUX2
3	USART1_TX_DMA	DMA_ITRIG_INMUX3
4	USART2_RX_DMA	DMA_ITRIG_INMUX4
5	USART2_TX_DMA	DMA_ITRIG_INMUX5
6	SPI0_RX_DMA	DMA_ITRIG_INMUX6
7	SPI0_TX_DMA	DMA_ITRIG_INMUX7
8	SPI1_RX_DMA	DMA_ITRIG_INMUX8
9	SPI1_TX_DMA	DMA_ITRIG_INMUX9
10	I2C0_SLV_DMA	DMA_ITRIG_INMUX10
11	I2C0_MST_DMA	DMA_ITRIG_INMUX11
12	I2C1_SLV_DMA	DMA_ITRIG_INMUX12
13	I2C1_MST_DMA	DMA_ITRIG_INMUX13
14	I2C2_SLV_DMA	DMA_ITRIG_INMUX14
15	I2C2_MST_DMA	DMA_ITRIG_INMUX15
16	I2C3_SLV_DMA	DMA_ITRIG_INMUX16
17	I2C3_MST_DMA	DMA_ITRIG_INMUX17

12.3.4 DMA in sleep mode

The DMA can operate and access all SRAM blocks in sleep mode.

12.4 Pin description

The DMA controller has no configurable pins.

12.5 General description

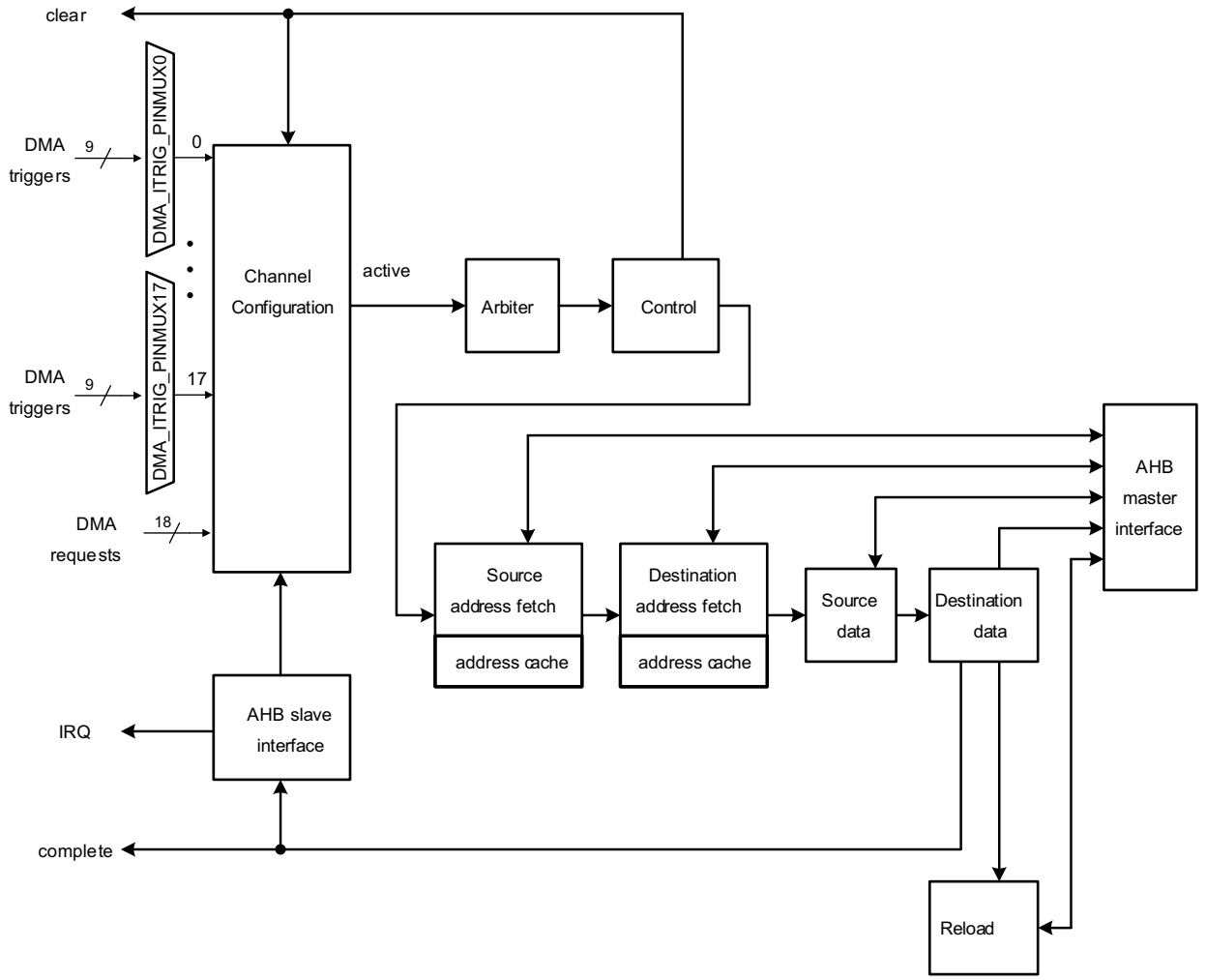


Fig 19. DMA block diagram

12.5.1 DMA requests and triggers

An operation on a DMA channel can be initiated by either a DMA request or a trigger event. DMA requests come from peripherals and specifically indicate when a peripheral either needs input data to be read from it, or that output data may be sent to it. DMA requests are created by the UART, SPI, and I2C peripherals.

A trigger initiates a DMA operation and can be a signal from an unrelated peripheral. Peripherals that generate triggers are the SCT, the ADC, and the analog comparator. In addition, the DMA triggers also create an trigger output that can trigger DMA transactions on another channel. Triggers can be used to send a character or a string to a UART or other serial output at a fixed time interval or when an event occurs.

A DMA channel using a trigger can respond by moving data from any memory address to any other memory address. This can include fixed peripheral data registers, or incrementing through RAM buffers. The amount of data moved by a single trigger event can range from a single transfer to many transfers. A transfer that is started by a trigger can still be paced using the channel's DMA request. This allows sending a string to a serial peripheral, for instance, without overrunning the peripheral's transmit buffer.

Each trigger input to the DMA has a corresponding output that can be used as a trigger input to another channel. The trigger outputs appear in the trigger source list for each channel and can be selected through the DMA_INMUX registers as inputs to other channels.

12.5.2 DMA Modes

The DMA controller doesn't really have separate operating modes, but there are ways of using the DMA controller that have commonly used terminology in the industry.

Once the DMA controller is set up for operation, using any specific DMA channel requires initializing the registers associated with that channel (see Table 148), and supplying at least the channel descriptor, which is located somewhere in memory, typically in on-chip SRAM (see Section 12.6.3). The channel descriptor is shown in Table 149.

Table 149. Channel descriptor

Offset	Description
+ 0x0	Reserved
+ 0x4	Source data end address
+ 0x8	Destination end address
+ 0xC	Link to next descriptor

The source and destination end addresses, as well as the link to the next descriptor are just memory addresses that can point to any valid address on the device. The starting address for both source and destination data is the specified end address minus the transfer length (XFERCOUNT * the address increment as defined by SRCINC and DSTINC). The link to the next descriptor is used only if it is a linked transfer.

After the channel has had a sufficient number of DMA requests and/or triggers, depending on its configuration, the initial descriptor will be exhausted. At that point, if the transfer configuration directs it, the channel descriptor will be reloaded with data from memory pointed to by the "Link to next descriptor" entry of the initial channel descriptor. Descriptors loaded in this manner look slightly different the channel descriptor, as shown in Table 150. The difference is that a new transfer configuration is specified in the reload descriptor instead of being written to the XFERCFG register for that channel.

This process repeats as each descriptor is exhausted as long as reload is selected in the transfer configuration for each new descriptor.

Table 150. Reload descriptors

Offset	Description
+ 0x0	Transfer configuration.
+ 0x4	Source end address. This points to the address of the last entry of the source address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size.
+ 0x8	Destination end address. This points to the address of the last entry of the destination address range if the address is incremented. The address to be used in the transfer is calculated from the end address, data width, and transfer size.
+ 0xC	Link to next descriptor. If used, this address must be aligned to a multiple of 16 bytes (i.e., the size of a descriptor).

12.5.3 Single buffer

This generally applies to memory to memory moves, and peripheral DMA that occurs only occasionally and is set up for each transfer. For this kind of operation, only the initial channel descriptor shown in Table 151 is needed.

Table 151. Channel descriptor for a single transfer

Offset	Description
+ 0x0	Reserved
+ 0x4	Source data end address
+ 0x8	Destination data end address
+ 0xC	(not used)

This case is identified by the Reload bit in the XFERCFG register = 0. When the DMA channel receives a DMA request or trigger (depending on how it is configured), it performs one or more transfers as configured, then stops. Once the channel descriptor is exhausted, additional DMA requests or triggers will have no effect until the channel configuration is updated by software.

12.5.4 Ping-Pong

Ping-pong is a special case of a linked transfer. It is described separately because it is typically used more frequently than more complicated versions of linked transfers.

A ping-pong transfer uses two buffers alternately. At any one time, one buffer is being loaded or unloaded by DMA operations. The other buffer has the opposite operation being handled by software, readying the buffer for use when the buffer currently being used by the DMA controller is full or empty. Table 152 shows an example of descriptors for ping-pong from a peripheral to two buffers in memory.

Table 152. Example descriptors for ping-pong operation: peripheral to buffer

Channel Descriptor			Descriptor B			Descriptor A	
+ 0x0	(not used)		+ 0x0	Buffer B transfer configuration		+ 0x0	Buffer A transfer configuration
+ 0x4	Peripheral data end address		+ 0x4	Peripheral data end address		+ 0x4	Peripheral data end address
+ 0x8	Buffer A memory end address		+ 0x8	Buffer B memory end address		+ 0x8	Buffer A memory end address
+ 0xC	Address of descriptor B		+ 0xC	Address of descriptor A		+ 0xC	Address of descriptor B

In this example, the channel descriptor is used first, with a first buffer in memory called buffer A. The configuration of the DMA channel must have been set to indicate a reload. Similarly, both descriptor A and descriptor B must also specify reload. When the channel descriptor is exhausted, descriptor B is loaded using the link to descriptor B, and a transfer interrupt informs the CPU that buffer A is available.

Descriptor B is then used until it is also exhausted, when descriptor A is loaded using the link to descriptor A contained in descriptor B. Then a transfer interrupt informs the CPU that buffer B is available for processing. The process repeats when descriptor A is exhausted, alternately using each of the 2 memory buffers.

12.5.5 Linked transfers (linked list)

A linked transfer can use any number of descriptors to define a complicated transfer. This can be configured such that a single transfer, a portion of a transfer, one whole descriptor, or an entire structure of links can be initiated by a single DMA request or trigger.

An example of a linked transfer could start out like the example for a ping-pong transfer (Table 152). The difference would be that descriptor B would not link back to descriptor A, but would continue on to another different descriptor. This could continue as long as desired, and can be ended anywhere, or linked back to any point to repeat a sequence of descriptors. Of course, any descriptor not currently in use can be altered by software as well.

12.5.6 Address alignment for data transfers

Transfers of 16 bit width require an address alignment to a multiple of 2 bytes. Transfers of 32 bit width require an address alignment to a multiple of 4 bytes. Transfers of 8 bit width can be at any address.

12.6 Register description

The DMA registers are grouped into DMA control, interrupt and status registers and DMA channel registers. DMA transfers are controlled by a set of three registers per channel, the CFG[0:20], CTRLSTAT[0:20], and XFERCFG[0:20] registers.

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

Table 153. Register overview: DMA controller (base address 0x5000 8000)

Name	Access	Address offset	Description	Reset Value	Reference
Global control and status registers					
CTRL	R/W	0x000	DMA control.	0	Table 154
INTSTAT	RO	0x004	Interrupt status.	0	Table 155
SRAMBASE	R/W	0x008	SRAM address of the channel configuration table.	0	Table 156
Shared registers					
ENABLESET0	RO/W1	0x020	Channel Enable read and Set for all DMA channels.	0	Table 158
ENABLECLR0	W1	0x028	Channel Enable Clear for all DMA channels.	NA	Table 159
ACTIVE0	RO	0x030	Channel Active status for all DMA channels.	0	Table 160
BUSY0	RO	0x038	Channel Busy status for all DMA channels.	0	Table 161
ERRINT0	RO/W1	0x040	Error Interrupt status for all DMA channels.	0	Table 162
INTENSET0	RO/W1	0x048	Interrupt Enable read and Set for all DMA channels.	0	Table 163
INTENCLR0	W1	0x050	Interrupt Enable Clear for all DMA channels.	NA	Table 164
INTA0	RO/W1	0x058	Interrupt A status for all DMA channels.	0	Table 165
INTB0	RO/W1	0x060	Interrupt B status for all DMA channels.	0	Table 166
SETVALID0	W1	0x068	Set ValidPending control bits for all DMA channels.	NA	Table 167
SETTRIG0	W1	0x070	Set Trigger control bits for all DMA channels.	NA	Table 168
ABORT0	W1	0x078	Channel Abort control for all DMA channels.	NA	Table 169
Channel0 registers					
CFG0	R/W	0x400	Configuration register for DMA channel 0.		Table 170
CTLSTAT0	RO	0x404	Control and status register for DMA channel 0.		Table 172
XFERCFG0	R/W	0x408	Transfer configuration register for DMA channel 0.		Table 173
Channel1 registers					
CFG1	R/W	0x410	Configuration register for DMA channel 1.		Table 170
CTLSTAT1	RO	0x414	Control and status register for DMA channel 1.		Table 172
XFERCFG1	R/W	0x418	Transfer configuration register for DMA channel 1.		Table 173
Channel2 registers					
CFG2	R/W	0x420	Configuration register for DMA channel 2.		Table 170
CTLSTAT2	RO	0x424	Control and status register for DMA channel 2.		Table 172
XFERCFG2	R/W	0x428	Transfer configuration register for DMA channel 2.		Table 173
Channel3 registers					
CFG3	R/W	0x430	Configuration register for DMA channel 3.		Table 170
CTLSTAT3	RO	0x434	Control and status register for DMA channel 3.		Table 172
XFERCFG3	R/W	0x438	Transfer configuration register for DMA channel 3.		Table 173
Channel4 registers					
CFG4	R/W	0x440	Configuration register for DMA channel 4.		Table 170
CTLSTAT4	RO	0x444	Control and status register for DMA channel 4.		Table 172
XFERCFG4	R/W	0x448	Transfer configuration register for DMA channel 4.		Table 173
Channel5 registers					
CFG5	R/W	0x450	Configuration register for DMA channel 5.		Table 170
CTLSTAT5	RO	0x454	Control and status register for DMA channel 5.		Table 172

Table 153. Register overview: DMA controller (base address 0x5000 8000)

Name	Access	Address offset	Description	Reset Value	Reference
XFERCFG5	R/W	0x458	Transfer configuration register for DMA channel 5.		Table 173
Channel6 registers					
CFG6	R/W	0x460	Configuration register for DMA channel 6.		Table 170
CTLSTAT6	RO	0x464	Control and status register for DMA channel 6.		Table 172
XFERCFG6	R/W	0x468	Transfer configuration register for DMA channel 6.		Table 173
Channel7 registers					
CFG7	R/W	0x470	Configuration register for DMA channel 7.		Table 170
CTLSTAT7	RO	0x474	Control and status register for DMA channel 7.		Table 172
XFERCFG7	R/W	0x478	Transfer configuration register for DMA channel 7.		Table 173
Channel8 registers					
CFG8	R/W	0x480	Configuration register for DMA channel 8.		Table 170
CTLSTAT8	RO	0x484	Control and status register for DMA channel 8.		Table 172
XFERCFG8	R/W	0x488	Transfer configuration register for DMA channel 8.		Table 173
Channel9 registers					
CFG9	R/W	0x490	Configuration register for DMA channel 9.		Table 170
CTLSTAT9	RO	0x494	Control and status register for DMA channel 9.		Table 172
XFERCFG9	R/W	0x498	Transfer configuration register for DMA channel 9.		Table 173
Channel10 registers					
CFG10	R/W	0x4A0	Configuration register for DMA channel 10.		Table 170
CTLSTAT10	RO	0x4A4	Control and status register for DMA channel 10.		Table 172
XFERCFG10	R/W	0x4A8	Transfer configuration register for DMA channel 10.		Table 173
Channel11 registers					
CFG11	R/W	0x4B0	Configuration register for DMA channel 11.		Table 170
CTLSTAT11	RO	0x4B4	Control and status register for DMA channel 11.		Table 172
XFERCFG11	R/W	0x4B8	Transfer configuration register for DMA channel 11.		Table 173
Channel12 registers					
CFG12	R/W	0x4C0	Configuration register for DMA channel 12.		Table 170
CTLSTAT12	RO	0x4C4	Control and status register for DMA channel 12.		Table 172
XFERCFG12	R/W	0x4C8	Transfer configuration register for DMA channel 12.		Table 173
Channel13 registers					
CFG13	R/W	0x4D0	Configuration register for DMA channel 13.		Table 170
CTLSTAT13	RO	0x4D4	Control and status register for DMA channel 13.		Table 172
XFERCFG13	R/W	0x4D8	Transfer configuration register for DMA channel 13.		Table 173
Channel14 registers					
CFG14	R/W	0x4E0	Configuration register for DMA channel 14.		Table 170
CTLSTAT14	RO	0x4E4	Control and status register for DMA channel 14.		Table 172
XFERCFG14	R/W	0x4E8	Transfer configuration register for DMA channel 14.		Table 173
Channel15 registers					
CFG15	R/W	0x4F0	Configuration register for DMA channel 15.		Table 170
CTLSTAT15	RO	0x4F4	Control and status register for DMA channel 15.		Table 172

Table 153. Register overview: DMA controller (base address 0x5000 8000)

Name	Access	Address offset	Description	Reset Value	Reference
XFERCFG15	R/W	0x4F8	Transfer configuration register for DMA channel 15.		Table 173
Channel16 registers					
CFG16	R/W	0x500	Configuration register for DMA channel 16.		Table 170
CTLSTAT16	RO	0x504	Control and status register for DMA channel 16.		Table 172
XFERCFG16	R/W	0x508	Transfer configuration register for DMA channel 16.		Table 173
Channel17 registers					
CFG17	R/W	0x510	Configuration register for DMA channel 17.		Table 170
CTLSTAT17	RO	0x514	Control and status register for DMA channel 17.		Table 172
XFERCFG17	R/W	0x518	Transfer configuration register for DMA channel 17.		Table 173

12.6.1 Control register

The CTRL register contains global the control bit for a enabling the DMA controller.

Table 154. Control register (CTRL, address 0x5000 8000) bit description

Bit	Symbol	Value	Description	Reset value
0	ENABLE		DMA controller master enable.	0
		0	Disabled. The DMA controller is disabled. This clears any triggers that were asserted at the point when disabled, but does not prevent re-triggering when the DMA controller is re-enabled.	
		1	Enabled. The DMA controller is enabled.	
31:1	-		Reserved. Read value is undefined, only zero should be written.	NA

12.6.2 Interrupt Status register

The Read-Only INTSTAT register provides an overview of DMA status. This allows quick determination of whether any enabled interrupts are pending. Details of which channels are involved are found in the interrupt type specific registers.

Table 155. Interrupt Status register (INTSTAT, address 0x5000 8004) bit description

Bit	Symbol	Value	Description	Reset value
0	-		Reserved. Read value is undefined, only zero should be written.	NA
1	ACTIVEINT		Summarizes whether any enabled interrupts are pending (except pending error interrupts).	0
		0	Not pending. No enabled interrupts are pending.	
		1	Pending. At least one enabled interrupt is pending.	
2	ACTIVEERRINT		Summarizes whether any error interrupts are pending.	0
		0	Not pending. No error interrupts are pending.	
		1	Pending. At least one error interrupt is pending.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	NA

12.6.3 SRAM Base address register

The SRAMBASE register must be configured with an address (preferably in on-chip SRAM) where DMA descriptors will be stored. Software must set up the descriptors for those DMA channels that will be used in the application.

Table 156. SRAM Base address register (SRAMBASE, address 0x5000 8008) bit description

Bit	Symbol	Description	Reset value
8:0	-	Reserved. Read value is undefined, only zero should be written.	NA
31:9	OFFSET	Address bits 31:9 of the beginning of the DMA descriptor table. For 18 channels, the table must begin on a 512 byte boundary.	0

Each DMA channel has an entry for the channel descriptor in the SRAM table. The values for each channel start at the address offsets found in Table 157. Only the descriptors for channels defined at extraction are used. The contents of each channel descriptor are described in Table 149.

Table 157. Channel descriptor map

Descriptor	Table offset
Channel descriptor for DMA channel 0	0x000
Channel descriptor for DMA channel 1	0x010
Channel descriptor for DMA channel 2	0x020
Channel descriptor for DMA channel 3	0x030
Channel descriptor for DMA channel 4	0x040
Channel descriptor for DMA channel 5	0x050
Channel descriptor for DMA channel 6	0x060
Channel descriptor for DMA channel 7	0x070
Channel descriptor for DMA channel 8	0x080
Channel descriptor for DMA channel 9	0x090
Channel descriptor for DMA channel 10	0x0A0
Channel descriptor for DMA channel 11	0x0B0
Channel descriptor for DMA channel 12	0x0C0
Channel descriptor for DMA channel 13	0x0D0
Channel descriptor for DMA channel 14	0x0E0
Channel descriptor for DMA channel 15	0x0F0
Channel descriptor for DMA channel 16	0x100
Channel descriptor for DMA channel 17	0x110

12.6.4 Enable read and Set registers

The ENABLESET0 register determines whether each DMA channel is enabled or disabled. Disabling a DMA channel does not reset the channel in any way. A channel can be paused and restarted by clearing, then setting the Enable bit for that channel.

Reading ENABLESET0 provides the current state of all of the DMA channels represented by that register. Writing a 1 to a bit position in ENABLESET0 that corresponds to an implemented DMA channel sets the bit, enabling the related DMA channel. Writing a 0 to any bit has no effect. Enables are cleared by writing to ENABLECLR0.

Table 158. Enable read and Set register 0 (ENABLESET0, address 0x5000 8020) bit description

Bit	Symbol	Description	Reset value
17:0	ENA	Enable for DMA channels 17:0. Bit n enables or disables DMA channel n. 0 = disabled. 1 = enabled.	0
31:18	-	Reserved.	-

12.6.5 Enable Clear register

The ENABLECLR0 register is used to clear the channel enable bits in ENABLESET0. This register is write-only.

Table 159. Enable Clear register 0 (ENABLECLR0, address 0x5000 8028) bit description

Bit	Symbol	Description	Reset value
17:0	CLR	Writing ones to this register clears the corresponding bits in ENABLESET0. Bit n clears the channel enable bit n.	NA
31:18	-	Reserved.	-

12.6.6 Active status register

The ACTIVE0 register indicates which DMA channels are active at the point when the read occurs. The register is read-only.

A DMA channel is considered active when a DMA operation has been started but not yet fully completed. The Active status will persist from a DMA operation being started, until the pipeline is empty after end of the last descriptor (when there is no reload). An active channel may be aborted by software by setting the appropriate bit in one of the Abort register (see Section 12.6.15).

Table 160. Active status register 0 (ACTIVE0, address 0x5000 8030) bit description

Bit	Symbol	Description	Reset value
17:0	ACT	Active flag for DMA channel n. Bit n corresponds to DMA channel n. 0 = not active. 1 = active.	0
31:18	-	Reserved.	-

12.6.7 Busy status register

The BUSY0 register indicates which DMA channels is busy at the point when the read occurs. This registers is read-only.

A DMA channel is considered busy when there is any operation related to that channel in the DMA controller's internal pipeline. This information can be used after a DMA channel is disabled by software (but still active), allowing confirmation that there are no remaining operations in progress for that channel.

Table 161. Busy status register 0 (BUSY0, address 0x5000 8038) bit description

Bit	Symbol	Description	Reset value
17:0	BSY	Busy flag for DMA channel n. Bit n corresponds to DMA channel n. 0 = not busy. 1 = busy.	0
31:18	-	Reserved.	-

12.6.8 Error Interrupt register

The ERRINT0 register contains flags for each DMA channel's Error Interrupt. Any pending interrupt flag in the register will be reflected on the DMA interrupt output.

Reading the registers provides the current state of all DMA channel error interrupts. Writing a 1 to a bit position in ERRINT0 that corresponds to an implemented DMA channel clears the bit, removing the interrupt for the related DMA channel. Writing a 0 to any bit has no effect.

Table 162. Error Interrupt register 0 (ERRINT0, address 0x5000 8040) bit description

Bit	Symbol	Description	Reset value
17:0	ERR	Error Interrupt flag for DMA channel n. Bit n corresponds to DMA channel n. 0 = error interrupt is not active. 1 = error interrupt is active.	0
31:18	-	Reserved.	-

12.6.9 Interrupt Enable read and Set register

The INTENSET0 register controls whether the individual Interrupts for DMA channels contribute to the DMA interrupt output.

Reading the registers provides the current state of all DMA channel interrupt enables. Writing a 1 to a bit position in INTENSET0 that corresponds to an implemented DMA channel sets the bit, enabling the interrupt for the related DMA channel. Writing a 0 to any bit has no effect. Interrupt enables are cleared by writing to INTENCLR0.

Table 163. Interrupt Enable read and Set register 0 (INTENSET0, address 0x5000 8048) bit description

Bit	Symbol	Description	Reset value
17: 0	INTEN	Interrupt Enable read and set for DMA channel n. Bit n corresponds to DMA channel n. 0 = interrupt for DMA channel is disabled. 1 = interrupt for DMA channel is enabled.	0
31:18	-	Reserved.	-

12.6.10 Interrupt Enable Clear register

The INTENCLR0 register is used to clear interrupt enable bits in INTENSET0. The register is write-only.

Table 164. Interrupt Enable Clear register 0 (INTENCLR0, address 0x5000 8050) bit description

Bit	Symbol	Description	Reset value
17:0	CLR	Writing ones to this register clears corresponding bits in the INTENSET0. Bit n corresponds to DMA channel n.	NA
31:18	-	Reserved.	-

12.6.11 Interrupt A register

The IntA0 register contains the interrupt A status for each DMA channel. The status will be set when the SETINTA bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in these registers clears the related INTA flag. Writing 0 has no effect. Any interrupt pending status in this register will be reflected on the DMA interrupt output if it is enabled in the related INTENSET register.

Remark: The error status is not included in this register. The error status is reported in the ERRINT0 status register.

Table 165. Interrupt A register 0 (INTA0, address 0x5000 8058) bit description

Bit	Symbol	Description	Reset value
17:0	IA	Interrupt A status for DMA channel n. Bit n corresponds to DMA channel n. 0 = the DMA channel interrupt A is not active. 1 = the DMA channel interrupt A is active.	0
31:18	-	Reserved.	-

12.6.12 Interrupt B register

The INTB0 register contains the interrupt B status for each DMA channel. The status will be set when the SETINTB bit is 1 in the transfer configuration for a channel, when the descriptor becomes exhausted. Writing a 1 to a bit in the register clears the related INTB flag. Writing 0 has no effect. Any interrupt pending status in this register will be reflected on the DMA interrupt output if it is enabled in the INTENSET register.

Remark: The error status is not included in this register. The error status is reported in the ERRINT0 status register.

Table 166. Interrupt B register 0 (INTB0, address 0x5000 8060) bit description

Bit	Symbol	Description	Reset value
17:0	IB	Interrupt B status for DMA channel n. Bit n corresponds to DMA channel n. 0 = the DMA channel interrupt B is not active. 1 = the DMA channel interrupt B is active.	0
31:18	-	Reserved.	-

12.6.13 Set Valid register

The SETVALID0 register allows setting the Valid bit in the CTRLSTAT register for one or more DMA channels. See Section 12.6.17 for a description of the VALID bit.

The CFGVALID and SV (set valid) bits allow more direct DMA block timing control by software. Each Channel Descriptor, in a sequence of descriptors, can be validated by either the setting of the CFGVALID bit or by setting the channel's SETVALID flag. Normally, the CFGVALID bit is set. This tells the DMA that the Channel Descriptor is active and can be executed. The DMA will continue sequencing through descriptor blocks whose CFGVALID bit are set without further software intervention. Leaving a CFGVALID bit set to 0 allows the DMA sequence to pause at the Descriptor until software triggers the continuation. If, during DMA transmission, a Channel Descriptor is found with CFGVALID

set to 0, the DMA checks for a previously buffered SETVALID0 setting for the channel. If found, the DMA will set the descriptor valid, clear the SV setting, and resume processing the descriptor. Otherwise, the DMA pauses until the channels SETVALID0 bit is set.

Table 167. Set Valid 0 register (SETVALID0, address 0x5000 8068) bit description

Bit	Symbol	Description	Reset value
17:0	SV	SETVALID control for DMA channel n. Bit n corresponds to DMA channel n. 0 = no effect. 1 = sets the VALIDPENDING control bit for DMA channel n.	NA
31:18	-	Reserved.	-

12.6.14 Set Trigger register

The SETTRIG0 register allows setting the TRIG bit in the CTRLSTAT register for one or more DMA channel. See Section 12.6.17 for a description of the TRIG bit, and Section 12.5.1 for a general description of triggering.

Table 168. Set Trigger 0 register (SETTRIG0, address 0x5000 8070) bit description

Bit	Symbol	Description	Reset value
17:0	TRIG	Set Trigger control bit for DMA channel 0. Bit n corresponds to DMA channel n. 0 = no effect. 1 = sets the TRIG bit for DMA channel n.	NA
31:18	-	Reserved.	-

12.6.15 Abort registers

The Abort0 register allows aborting operation of a DMA channel if needed. To abort a selected channel, the channel should first be disabled by clearing the corresponding Enable bit by writing a 1 to the proper bit ENABLECLR. Then wait until the channel is no longer busy by checking the corresponding bit in BUSY. Finally, write a 1 to the proper bit of ABORT. This prevents the channel from restarting an incomplete operation when it is enabled again.

Table 169. Abort 0 register (ABORT0, address 0x5000 8078) bit description

Bit	Symbol	Description	Reset value
17:0	ABORTCTRL	Abort control for DMA channel 0. Bit n corresponds to DMA channel n. 0 = no effect. 1 = aborts DMA operations on channel n.	NA
31:18	-	Reserved.	-

12.6.16 Channel configuration registers

The CFGn register contains various configuration options for DMA channel n.

See Table 171 for a summary of trigger options.

Table 170. Configuration registers for channel 0 to 17 (CFG[0:17], addresses 0x5000 8400 (CFG0) to address 0x5000 8510 (CFG17)) bit description

Bit	Symbol	Value	Description	Reset value
0	PERIPHREQEN		Peripheral request Enable. If a DMA channel is used to perform a memory-to-memory move, any peripheral DMA request associated with that channel can be disabled to prevent any interaction between the peripheral and the DMA controller.	0
		0	Disabled. Peripheral DMA requests are disabled.	
		1	Enabled. Peripheral DMA requests are enabled.	
1	HWTRIGEN		Hardware Triggering Enable for this channel.	0
		0	Disabled. Hardware triggering is not used.	
		1	Enabled. Use hardware triggering.	
3:2	-		Reserved. Read value is undefined, only zero should be written.	NA
4	TRIGPOL		Trigger Polarity. Selects the polarity of a hardware trigger for this channel.	0
		0	Active low - falling edge. Hardware trigger is active low or falling edge triggered, based on TRIGTYPE.	
		1	Active high - rising edge. Hardware trigger is active high or rising edge triggered, based on TRIGTYPE.	
5	TRIGTYPE		Trigger Type. Selects hardware trigger as edge triggered or level triggered.	0
		0	Edge. Hardware trigger is edge triggered. Transfers will be initiated and completed, as specified for a single trigger.	
		1	Level. Hardware trigger is level triggered. Note that when level triggering without burst (BURSTPOWER = 0) is selected, only hardware triggers should be used on that channel. Transfers continue as long as the trigger level is asserted. Once the trigger is de-asserted, the transfer will be paused until the trigger is, again, asserted. However, the transfer will not be paused until any remaining transfers within the current BURSTPOWER length are completed.	
6	TRIGBURST		Trigger Burst. Selects whether hardware triggers cause a single or burst transfer.	0
		0	Single transfer. Hardware trigger causes a single transfer.	
		1	Burst transfer. When the trigger for this channel is set to edge triggered, a hardware trigger causes a burst transfer, as defined by BURSTPOWER. When the trigger for this channel is set to level triggered, a hardware trigger causes transfers to continue as long as the trigger is asserted, unless the transfer is complete.	
7	-		Reserved. Read value is undefined, only zero should be written.	NA

Table 170. Configuration registers for channel 0 to 17 (CFG[0:17], addresses 0x5000 8400 (CFG0) to address 0x5000 8510 (CFG17)) bit description

Bit	Symbol	Value	Description	Reset value
11:8	BURSTPOWER		Burst Power is used in two ways. It always selects the address wrap size when SRCBURSTWRAP and/or DSTBURSTWRAP modes are selected (see descriptions elsewhere in this register). When the TRIGBURST field elsewhere in this register = 1, Burst Power selects how many transfers are performed for each DMA trigger. This can be used, for example, with peripherals that contain a FIFO that can initiate a DMA operation when the FIFO reaches a certain level. 0000: Burst size = 1 (2^0). 0001: Burst size = 2 (2^1). 0010: Burst size = 4 (2^2). ... 1010: Burst size = 1024 (2^{10}). This corresponds to the maximum supported transfer count. others: not supported. The total transfer length as defined in the XFERCOUNT bits in the XFERCFG register must be an even multiple of the burst size.	0
13:12	-		Reserved. Read value is undefined, only zero should be written.	NA
14	SRCBURSTWRAP		Source Burst Wrap. When enabled, the source data address for the DMA is “wrapped”, meaning that the source address range for each burst will be the same. As an example, this could be used to read several sequential registers from a peripheral for each DMA burst, reading the same registers again for each burst.	0
		0	Disabled. Source burst wrapping is not enabled for this DMA channel.	
		1	Enabled. Source burst wrapping is enabled for this DMA channel.	
15	DSTBURSTWRAP		Destination Burst Wrap. When enabled, the destination data address for the DMA is “wrapped”, meaning that the destination address range for each burst will be the same. As an example, this could be used to write several sequential registers to a peripheral for each DMA burst, writing the same registers again for each burst.	0
		0	Disabled. Destination burst wrapping is not enabled for this DMA channel.	
		1	Enabled. Destination burst wrapping is enabled for this DMA channel.	
18:16	CHPRIORITY		Priority of this channel when multiple DMA requests are pending. Eight priority levels are supported. 0x0 = highest priority. 0x7 = lowest priority.	0
31:19	-		Reserved. Read value is undefined, only zero should be written.	NA

Table 171. Trigger setting summary

TrigBurst	TrigType	TrigPol	Description
0	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
0	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap.
1	0	0	Hardware DMA trigger is falling edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	0	1	Hardware DMA trigger is rising edge sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	1	0	Hardware DMA trigger is low level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.
1	1	1	Hardware DMA trigger is high level sensitive. The BURSTPOWER field controls address wrapping if enabled via SrcBurstWrap and/or DstBurstWrap, and also determines how much data is transferred for each trigger.

12.6.17 Channel control and status registers

The CTLSTATn register provides status flags specific to DMA channel n.

Table 172. Control and Status registers for channel 0 to 17 (CTLSTAT[0:17], 0x5000 8404 (CTLSTAT0) to address 0x5000 8514 (CTLSTAT17)) bit description

Bit	Symbol	Value	Description	Reset value
0	VALIDPENDING		Valid pending flag for this channel. This bit is set when a 1 is written to the corresponding bit in the related SETVALID register when CFGVALID = 1 for the same channel.	0
		0	No effect on DMA operation.	
		1	Valid pending.	
1	-		Reserved. Read value is undefined, only zero should be written.	NA
2	TRIG		Trigger flag. Indicates that the trigger for this channel is currently set. This bit is cleared at the end of an entire transfer or upon reload when CLRTRIG = 1.	0
		0	Not triggered. The trigger for this DMA channel is not set. DMA operations will not be carried out.	
		1	Triggered. The trigger for this DMA channel is set. DMA operations will be carried out.	
31:3	-		Reserved. Read value is undefined, only zero should be written.	NA

12.6.18 Channel transfer configuration registers

The XFERCFGn register contains transfer related configuration information for DMA channel n. Using the Reload bit, this register can optionally be automatically reloaded when the current settings are exhausted (the full transfer count has been completed), allowing linked transfers with more than one descriptor to be performed.

See "Trigger operation" for details on trigger operation.

Table 173. Transfer Configuration registers for channel 0 to 17 (XFERCFG[0:17], addresses 0x5000 8408 (XFERCFG0) to 0x5000 8518 (XFERCFG17)) bit description

Bit	Symbol	Value	Description	Reset Value
0	CFGVALID		Configuration Valid flag. This bit indicates whether the current channel descriptor is valid and can potentially be acted upon, if all other activation criteria are fulfilled.	0
		0	Not valid. The channel descriptor is not considered valid until validated by an associated SETVALID0 setting.	
		1	Valid. The current channel descriptor is considered valid.	
1	RELOAD		Indicates whether the channel's control structure will be reloaded when the current descriptor is exhausted. Reloading allows ping-pong and linked transfers.	0
		0	Disabled. Do not reload the channels' control structure when the current descriptor is exhausted.	
		1	Enabled. Reload the channels' control structure when the current descriptor is exhausted.	
2	SWTRIG		Software Trigger.	0
		0	When written by software, the trigger for this channel is not set. A new trigger, as defined by the HWTRIGEN, TRIGPOL, and TRIGTYPE will be needed to start the channel.	
		1	When written by software, the trigger for this channel is set immediately. This feature should not be used with level triggering when TRIGBURST = 0.	
3	CLRTRIG		Clear Trigger.	0
		0	Not cleared. The trigger is not cleared when this descriptor is exhausted. If there is a reload, the next descriptor will be started.	
		1	Cleared. The trigger is cleared when this descriptor is exhausted.	
4	SETINTA		Set Interrupt flag A for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed.	0
		0	No effect.	
		1	Set. The INTA flag for this channel will be set when the current descriptor is exhausted.	
5	SETINTB		Set Interrupt flag B for this channel. There is no hardware distinction between interrupt A and B. They can be used by software to assist with more complex descriptor usage. By convention, interrupt A may be used when only one interrupt flag is needed.	0
		0	No effect.	
		1	Set. The INTB flag for this channel will be set when the current descriptor is exhausted.	
7:6	-		Reserved. Read value is undefined, only zero should be written.	NA

Table 173. Transfer Configuration registers for channel 0 to 17 (XFERCFG[0:17], addresses 0x5000 8408 (XFERCFG0) to 0x5000 8518 (XFERCFG17)) bit description

Bit	Symbol	Value	Description	Reset Value
9:8	WIDTH		Transfer width used for this DMA channel.	0
		0x0	8-bit transfers are performed (8-bit source reads and destination writes).	
		0x1	16-bit transfers are performed (16-bit source reads and destination writes).	
		0x2	32-bit transfers are performed (32-bit source reads and destination writes).	
		0x3	Reserved setting, do not use.	
11:10	-		Reserved. Read value is undefined, only zero should be written.	NA
13:12	SRCINC		Determines whether the source address is incremented for each DMA transfer.	0
		0x0	No increment. The source address is not incremented for each transfer. This is the usual case when the source is a peripheral device.	
		0x1	1 x width. The source address is incremented by the amount specified by Width for each transfer. This is the usual case when the source is memory.	
		0x2	2 x width. The source address is incremented by 2 times the amount specified by Width for each transfer.	
		0x3	4 x width. The source address is incremented by 4 times the amount specified by Width for each transfer.	
15:14	DSTINC		Determines whether the destination address is incremented for each DMA transfer.	0
		0x0	No increment. The destination address is not incremented for each transfer. This is the usual case when the destination is a peripheral device.	
		0x1	1 x width. The destination address is incremented by the amount specified by Width for each transfer. This is the usual case when the destination is memory.	
		0x2	2 x width. The destination address is incremented by 2 times the amount specified by Width for each transfer.	
		0x3	4 x width. The destination address is incremented by 4 times the amount specified by Width for each transfer.	
25:16	XFERCOUNT		Total number of transfers to be performed, minus 1 encoded. The number of bytes transferred is: (XFERCOUNT + 1) x data width (as defined by the WIDTH field). **Remark:** The DMA controller uses this bit field during transfer to count down. Hence, it cannot be used by software to read back the size of the transfer, for instance, in an interrupt handler. 0x0 = a total of 1 transfer will be performed. 0x1 = a total of 2 transfers will be performed. ... 0x3FF = a total of 1,024 transfers will be performed.	0
31:26	-		Reserved. Read value is undefined, only zero should be written.	NA

12.7 Functional description

12.7.1 Trigger operation

A trigger of some kind is always needed to start a transfer on a DMA channel. This can be a hardware or software trigger, and can be used in several ways.

If a channel is configured with the SWTRIG bit equal to 0, the channel can be later triggered either by hardware or software. Software triggering is accomplished by writing a 1 to the appropriate bit in the SETTRIG register. Hardware triggering requires setup of the HWTRIGEN, TRIGPOL, TRIGTYPE, and TRIGBURST fields in the CFG register for the related channel. When a channel is initially set up, the SWTRIG bit in the XFERCFG register can be set, causing the transfer to begin immediately.

Once triggered, transfer on a channel will be paced by DMA requests if the PERIPHREQEN bit in the related CFG register is set. Otherwise, the transfer will proceed at full speed.

The TRIG bit in the CTLSTAT register can be cleared at the end of a transfer, determined by the value CLRTRIG (bit 0) in the XFERCFG register. When a 1 is found in CLRTRIG, the trigger is cleared when the descriptor is exhausted.

13.1 How to read this chapter

Three USARTs are available on all parts depending on the switch matrix configuration.

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.
- USARTn transmit and receive functions can operated with the system DMA controller.

13.3 Basic configuration

Configure the USARTs for receiving and transmitting data:

- In the SYSAHBCLKCTRL register, set bit 3 to 5 (Table 35) to enable the clock to the register interface.
- Clear the USART peripheral resets using the PRESETCTRL register (Table 23).
- Enable or disable the USART interrupts in slots #3 to #5 in the NVIC. See Table 5.
- Configure the USART pin functions through the switch matrix.
- Configure the USART clock and baud rate. See Section 13.3.1.
- Send and receive lines are connected to DMA request lines. See Table 148.

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

All three USARTs use a common peripheral clock (U_PCLK) and, if needed, a fractional baud rate generator. The peripheral clock and the fractional divider for the baud rate calculation are set up in the SYSCON block as follows (see Figure 20):

1. Configure the UART clock by writing a value FRGCLKDIV > 0 in the common USART fractional baud rate divider register. This is the divided main clock common to all USARTs.

   Table 36 “USART clock divider register (UARTCLKDIV, address 0x4004 8094) bit description”

2. 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.

   U_PCLK = FRGCLKDIV/(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.

   Table 40 “USART fractional generator divider value register (UARTFRGDIV, address 0x4004 80F0) bit description”

3. In asynchronous mode: Configure the baud rate divider BRGVAL in the USARTn BRG register. The baud rate divider divides the common USART peripheral clock by a factor of 16 multiplied by the baud rate value to provide the baud rate = U_PCLK/16 x BRGVAL.

   Section 13.6.9 “USART Baud Rate Generator register”

4. In synchronous mode: The serial clock is Un_SCLK = U_PCLK/BRGVAL.

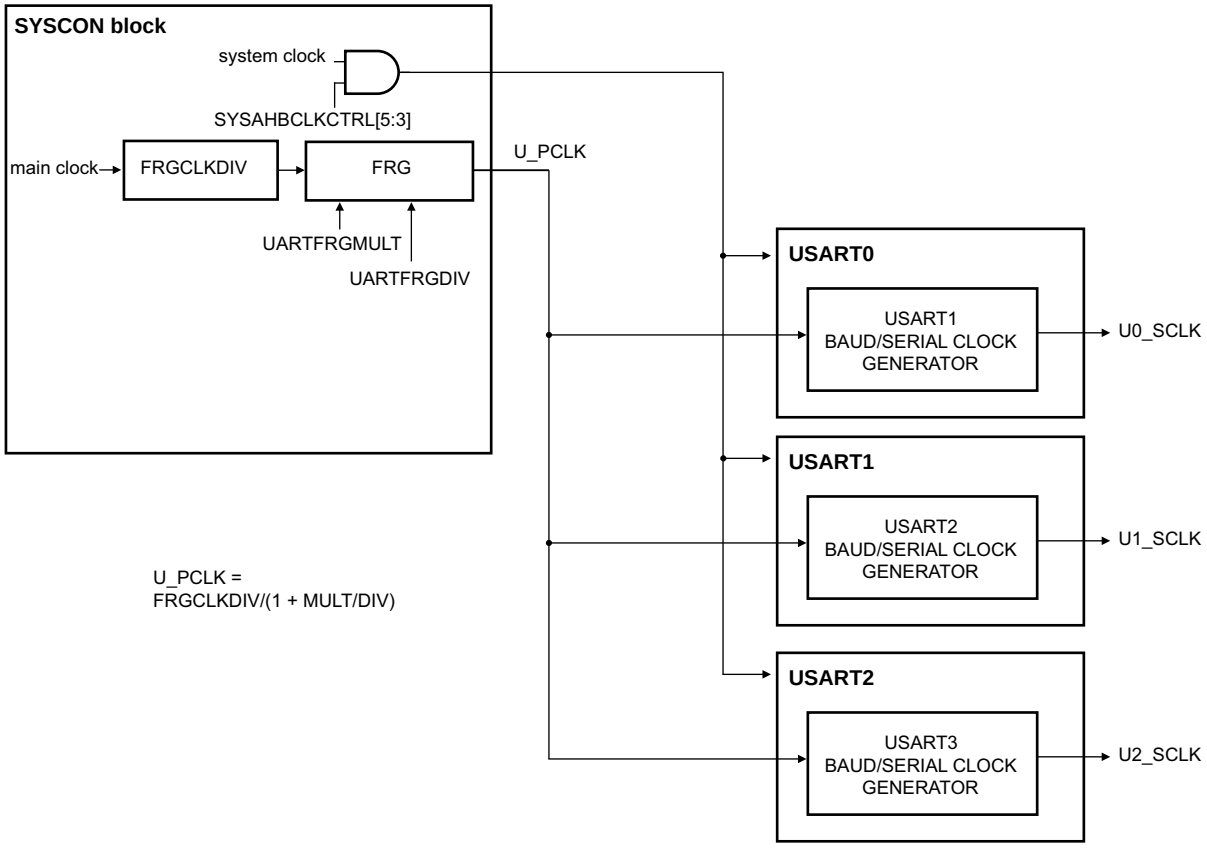


Fig 20. USART clocking

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 51 “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 67	Transmitter output for USART0. Serial transmit data.
U0_RXD	I	external to pin	any pin	PINASSIGN0	Table 67	Receiver input for USART0.
$\overline{\text{U0\_RTS}}$	O	external to pin	any pin	PINASSIGN0	Table 67	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.

Table 174. USART pin description

Function	I/O	Type	Connect to	Use register	Reference	Description
$\overline{\text{U0\_CTS}}$	I	external to pin	any pin	PINASSIGN0	Table 67	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 68	Serial clock input/output for USART0 n synchronous mode.
U1_TXD	O	external to pin	any pin	PINASSIGN1	Table 68	Transmitter output for USART1. Serial transmit data.
U1_RXD	I	external to pin	any pin	PINASSIGN1	Table 68	Receiver input for USART1.
$\overline{\text{U1\_RTS}}$	O	external to pin	any pin	PINASSIGN1	Table 68	Request To Send output for USART1.
$\overline{\text{U1\_CTS}}$	I	external to pin	any pin	PINASSIGN2	Table 69	Clear To Send input for USART1.
U1_SCLK	I/O	external to pin	any pin	PINASSIGN2	Table 69	Serial clock input/output for USART1 in synchronous mode.
U2_TXD	O	external to pin	any pin	PINASSIGN2	Table 69	Transmitter output for USART2. Serial transmit data.
U2_RXD	I	external to pin	any pin	PINASSIGN2	Table 69	Receiver input for USART2.
$\overline{\text{U2\_RTS}}$	O	external to pin	any pin	PINASSIGN3	Table 70	Request To Send output for USART2.
$\overline{\text{U2\_CTS}}$	I	external to pin	any pin	PINASSIGN3	Table 70	Clear To Send input for USART2.
U2_SCLK	I/O	external to pin	any pin	PINASSIGN3	Table 70	Serial clock input/output for USART2 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 or the DMA controller.

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).

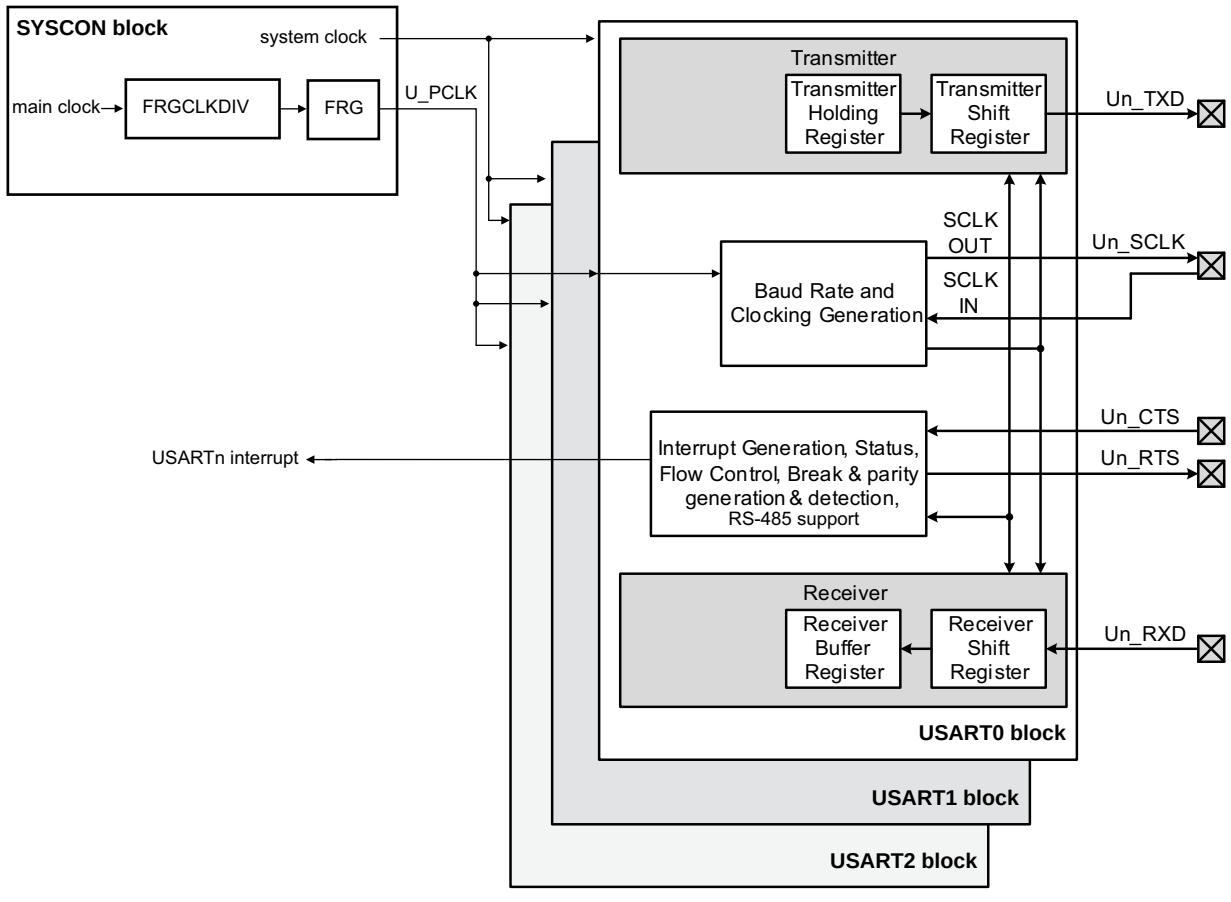
The USART transmitter block accepts data written by the CPU or DMA controllers and buffers the data in the transmit holding register. When the transmitter is available, the transmit shift register takes that data, formats it, and serializes it to the serial output, Un_TXD.

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.



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), 0x4006 8000 (USART1), 0x4006 C000 (USART2))

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 DMA or 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), 0x4006 C000 (USART2)) bit description

Bit	Symbol	Value	Description	Reset Value
0	ENABLE		USART Enable.	0
		0	Disabled. The USART is disabled and the internal state machine and counters are reset. While Enable = 0, all USART interrupts and DMA transfers are disabled. When Enable is set again, CFG and most other control bits remain unchanged. For instance, when re-enabled, the USART will immediately generate a TXRDY interrupt (if enabled in the INTENSET register) or a DMA transfer request because the transmitter has been reset and is therefore available.	
		1	Enabled. The USART is enabled for operation.	
1	-		Reserved. Read value is undefined, only zero should be written.	NA
3:2	DATALEN		Selects the data size for the USART.	00
		0x0	7 bit Data length.	
		0x1	8 bit Data length.	
		0x2	9 bit data length. The 9th bit is commonly used for addressing in multidrop mode. See the ADDRDET bit in the CTL register.	
		0x3	Reserved.	
5:4	PARITYSEL		Selects what type of parity is used by the USART.	00
		0x0	No parity.	
		0x1	Reserved.	
		0x2	Even parity. Adds a bit to each character such that the number of 1s in a transmitted character is even, and the number of 1s in a received character is expected to be even.	
		0x3	Odd parity. Adds a bit to each character such that the number of 1s in a transmitted character is odd, and the number of 1s in a received character is expected to be odd.	
6	STOPLEN		Number of stop bits appended to transmitted data. Only a single stop bit is required for received data.	0
		0	1 stop bit.	
		1	2 stop bits. This setting should only be used for asynchronous communication.	
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), 0x4006 C000 (USART2)) bit description *…continued …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.	

Table 176. USART Configuration register (CFG, address 0x4006 4000 (USART0), 0x4006 8000 (USART1), 0x4006 C000 (USART2)) bit description *…continued …continued*

Bit	Symbol	Value	Description	Reset Value
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.	
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), 0x4006 C004 (USART2)) 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. A break may be sent without danger of corrupting any currently transmitting character if the transmitter is first disabled (TXDIS in CTL is set) and then waiting for the transmitter to be disabled (TXDISINT in STAT = 1) before writing 1 to TXBRKEN.	
2	ADDRDET		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

Table 177. USART Control register (CTL, address 0x4006 4004 (USART0), 0x4006 8004 (USART1), 0x4006 C004 (USART2)) bit description

Bit	Symbol	Value	Description	Reset value
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), 0x4006 C008 (USART2)) 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

Table 178. USART Status register (STAT, address 0x4006 4008 (USART0), 0x4006 8008 (USART1), 0x4006 C008 (USART2)) bit description

Bit	Symbol	Description	Reset value	Access [1]
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
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), 0x4006C00C (USART2)) 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
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), 0x4006 C010 (USART2)) 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), 0x4006 C014 (USART2)) 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, which may be especially useful when the DMA is used with the USART receiver.

Remark: Reading this register changes the status flags.

Table 182. USART Receiver Data with Status register (RXDATSTAT, address 0x4006 4018 (USART0), 0x4006 8018 (USART1), 0x4006 C018 (USART2)) 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), 0x4006 C01C (USART2)) 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), 0x4006 8020 (USART2)) bit description

Bit	Symbol	Description	Reset Value
15:0	BRGVAL	This value is used to divide the USART input clock to determine the baud rate, based on the input clock from the FRG. 0 = The FRG clock is used directly by the USART function. 1 = The FRG clock is divided by 2 before use by the USART function. 2 = The FRG clock is divided by 3 before use by the USART function. ... 0xFFFF = The FRG clock is divided by 65,536 before use by the USART function.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

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), 0x4006 8024 (USART2)) 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), 0x4006 8028 (USART2)) bit description

Bit	Symbol	Description	Reset value
3:0	OSRVAL	Oversample Selection Value. 0 to 3 = not supported 0x4 = 5 peripheral clocks are used to transmit and receive each data bit. 0x5 = 6 peripheral clocks are used to transmit and receive each data bit. ... 0xF= 16 peripheral clocks are used to transmit and receive each data bit.	0xF
31:4	-	Reserved, the value read from a reserved bit is not defined.	NA

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), 0x4006 C02C (USART2)) 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 setting up the fractional divider use the following registers:

Table 36 “USART clock divider register (UARTCLKDIV, address 0x4004 8094) bit description”, Table 40 “USART fractional generator divider value register (UARTFRGDIV, address 0x4004 80F0) bit description”, and Table 41 “USART fractional generator multiplier value register (UARTFRGMULT, address 0x4004 80F4) bit description”.

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 DMA

A DMA request is provided for each USART direction, and can be used in lieu of interrupts for transferring data by configuring the DMA controller appropriately. The DMA controller provides an acknowledgement signal that clears the related request when it completes handling a that request. The transmitter DMA request is asserted when the transmitter can accept more data. The receiver DMA request is asserted when received data is available to be read.

When DMA is used to perform USART data transfers, other mechanisms can be used to generate interrupts when needed. For instance, completion of the configured DMA transfer can generate an interrupt from the DMA controller. Also, interrupts for special conditions, such as a received break, can still generate useful interrupts.

13.7.3 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.

13.7.4 Flow control

The USART supports both hardware and software flow control.

13.7.4.1 Hardware flow control

The USART supports hardware flow control using RTS and/or CTS signalling. If RTS is configured to appear on a device pin so that it can be sent to an external device, it indicates to an external device the ability of the receiver to receive more data.

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.

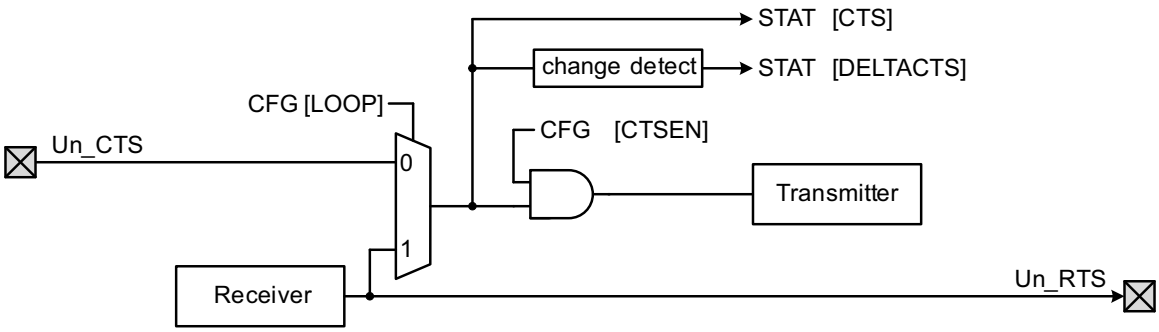


Fig 22. Hardware flow control using RTS and CTS

13.7.4.2 Software flow control

Software flow control could include XON / XOFF flow control, or other mechanisms. these are supported by the ability to check the current state of the CTS input, and/or have an interrupt when CTS changes state (via the CTS and DELTACTS bits, respectively, in the STAT register), and by the ability of software to gracefully turn off the transmitter (via the TXDIS bit in the CTL register).

13.7.5 Autobaud function

The autobaud functions attempts to measure the start bit time of the next received character. For this to work, the measured character must have a 1 in the least significant bit position, so that the start bit is bounded by a falling and rising edge. The measurement is made using the current clocking settings, including the oversampling configuration. The

result is that a value is stored in the BRG register that is as close as possible to the correct setting for the sampled character and the current clocking settings. The sampled character is provided in the RXDAT and RXDATSTAT registers, allowing software to double-check for the expected character.

Autobaud includes a time-out that is flagged by ABERR if no character is received at the expected time. It is recommended that autobaud only be enabled when the USART receiver is idle. Once enabled, either RXRDY or ABERR will be asserted at some point. 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.

13.7.6 RS-485 support

RS-485 support requires some form of address recognition and data direction control.

This USART has provisions for hardware address recognition (see the AUTOADDR bit in the CFG register in Section 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.

13.7.7 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 interfaces are available on all parts depending on the switch matrix configuration.

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 four Slave Select input/outputs with selectable polarity and flexible usage.
- Supports DMA transfers: SPIn transmit and receive functions can operated with the system DMA controller.

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

14.3 Basic configuration

Configure SPI0/1 using the following registers:

- In the SYSAHBCLKCTRL register, set bit 11 and 12 (Table 35) to enable the clock to the register interface.
- Clear the SPI0/1 peripheral resets using the PRESETCTRL register (Table 23).
- Enable/disable the SPI0/1 interrupts in interrupt slots #0 and 1 in the NVIC.
- Configure the SPI0/1 pin functions through the switch matrix. See Section 14.4.
- The peripheral clock for both SPIs is the system clock (see Figure 5 “Clock generation”).

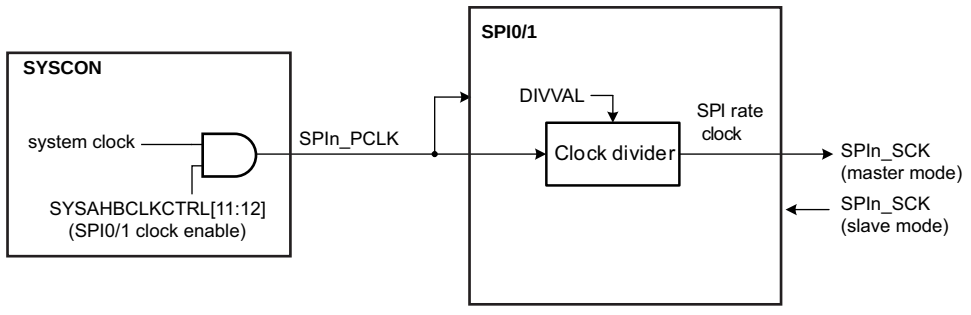


Fig 23. SPI clocking

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 51 "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

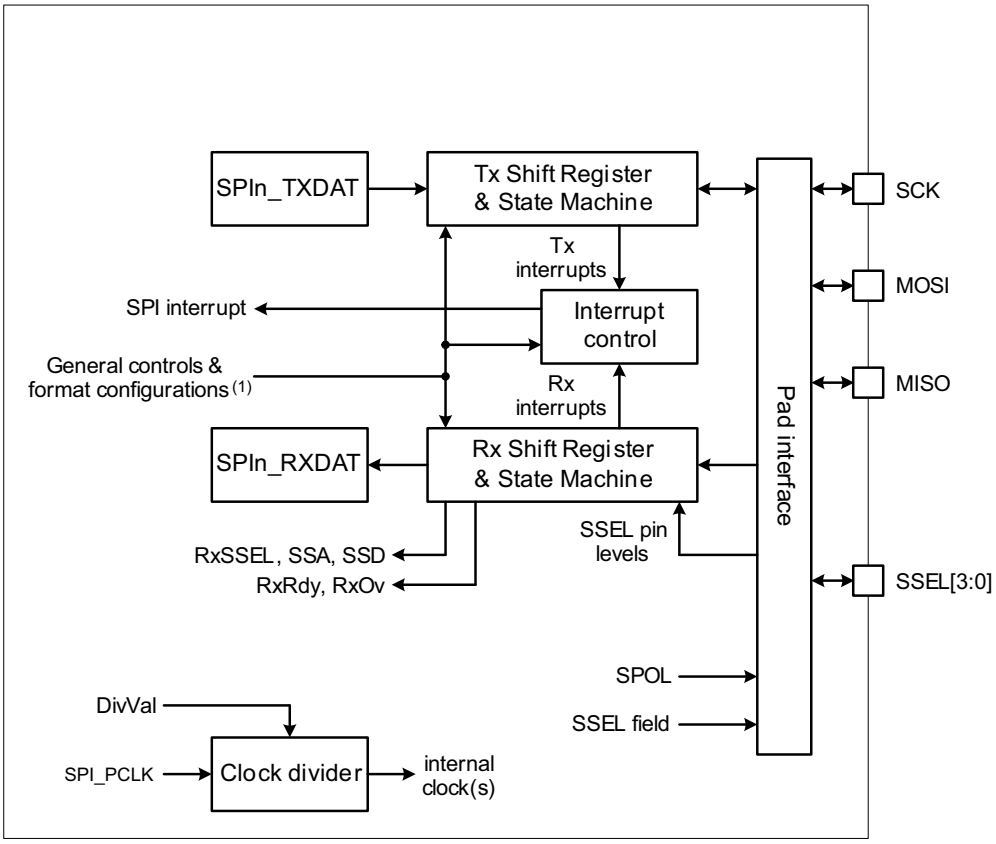
The SPI signals are movable functions and are assigned to external pins through the switch matrix.

See Section 7.3.1 "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	PINASSIGN3	Table 70	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	PINASSIGN4	Table 71	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	PINASSIGN4	Table 71	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	PINASSIGN4	Table 71	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	PINASSIGN4	Table 71	Slave Select 1.
SPI0_SSEL2	I/O	external to pin	any pin	PINASSIGN5	Table 72	Slave Select 2.
SPI0_SSEL3	I/O	external to pin	any pin	PINASSIGN5	Table 72	Slave Select 3.
SPI1_SCK	I/O	external to pin	any pin	PINASSIGN5	Table 72	Serial Clock.
SPI1_MOSI	I/O	external to pin	any pin	PINASSIGN5	Table 72	Master Out Slave In.
SPI1_MISO	I/O	external to pin	any pin	PINASSIGN6	Table 73	Master In Slave Out.
SPI1_SSEL0	I/O	external to pin	any pin	PINASSIGN6	Table 73	Slave Select 0.
SPI1_SSEL1	I/O	external to pin	any pin	PINASSIGN6	Table 73	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 (SPI0) and 0x4005 C000 (SPI1))

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

Table 189. Register overview: SPI (base address 0x4005 8000 (SPI0) and 0x4005 C000 (SPI1))
…continued

Name	Access	Offset	Description	Reset value	Reference
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
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 (SPI0), 0x4005 C000 (SPI1)) 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).	

Table 190. SPI Configuration register (CFG, addresses 0x4005 8000 (SPI0), 0x4005 C000 (SPI1)) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
4	CPHA		Clock Phase select.	0
		0	Change. The SPI captures serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is changed on the following edge.	
		1	Capture. The SPI changes serial data on the first clock transition of the transfer (when the clock changes away from the rest state). Data is captured on the following edge.	
5	CPOL		Clock Polarity select.	0
		0	Low. The rest state of the clock (between transfers) is low.	
		1	High. The rest state of the clock (between transfers) is high.	
6	-		Reserved. Read value is undefined, only zero should be written.	NA
7	LOOP		Loopback mode enable. Loopback mode applies only to Master mode, and connects transmit and receive data connected together to allow simple software testing.	0
		0	Disabled.	
		1	Enabled.	
8	SPOL0		SSEL0 Polarity select.	0
		0	Low. The SSEL0 pin is active low. 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.	
10	SPOL2		SSEL2 Polarity select.	0
		0	Low. The SSEL2 pin is active low. The value in the SSEL2 fields of the RXDAT, TXDATCTL, and TXCTL registers related to SSEL2 is not inverted relative to the pins.	
		1	High. The SSEL2 pin is active high. The value in the SSEL2 fields of the RXDAT, TXDATCTL, and TXCTL registers related to SSEL2 is inverted relative to the pins.	
11	SPOL3		SSEL3 Polarity select.	0
		0	Low. The SSEL3 pin is active low. The value in the SSEL3 fields of the RXDAT, TXDATCTL, and TXCTL registers related to SSEL3 is not inverted relative to the pins.	
		1	High. The SSEL3 pin is active high. The value in the SSEL3 fields of the RXDAT, TXDATCTL, and TXCTL registers related to SSEL3 is inverted relative to the pins.	
31:12	-		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 (SPI0), 0x4005 C004 (SPI1)) bit description

Bit	Symbol	Description	Reset value
3:0	PRE_DELAY	Controls the amount of time between SSEL assertion and the beginning of a data transfer. There is always one SPI clock time between SSEL assertion and the first clock edge. This is not considered part of the pre-delay. 0x0 = No additional time is inserted. 0x1 = 1 SPI clock time is inserted. 0x2 = 2 SPI clock times are inserted. ... 0xF = 15 SPI clock times are inserted.	0
7:4	POST_DELAY	Controls the amount of time between the end of a data transfer and SSEL deassertion. 0x0 = No additional time is inserted. 0x1 = 1 SPI clock time is inserted. 0x2 = 2 SPI clock times are inserted. ... 0xF = 15 SPI clock times are inserted.	0
11:8	FRAME_DELAY	If the EOF flag is set, controls the minimum amount of time between the current frame and the next frame (or SSEL deassertion if EOT). 0x0 = No additional time is inserted. 0x1 = 1 SPI clock time is inserted. 0x2 = 2 SPI clock times are inserted. ... 0xF = 15 SPI clock times are inserted.	0
15:12	TRANSFER_DELAY	Controls the minimum amount of time that the SSEL is deasserted between transfers. 0x0 = The minimum time that SSEL is deasserted is 1 SPI clock time. (Zero added time.) 0x1 = The minimum time that SSEL is deasserted is 2 SPI clock times. 0x2 = The minimum time that SSEL is deasserted is 3 SPI clock times. ... 0xF = The minimum time that SSEL is deasserted is 16 SPI clock times.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

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 (SPI0), 0x4005 C008 (SPI1)) 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 (SPI0), 0x4005 C008 (SPI1)) 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 (SPI0), 0x4005 C00C (SPI1)) 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. 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 (SPI0), 0x4005 C00C (SPI1)) 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.	
31:6	-		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 (SPI0), 0x4005 C010 (SPI1)) 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
31:6	-	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 (SPI0), 0x4005 C014 (SPI1)) 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, which can be especially useful when the SPI is used with DMA.

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 14.7.6 “Data lengths greater than 16 bits”.

Table 196. SPI Transmitter Data and Control register (TXDATCTL, addresses 0x4005 8018 (SPI0), 0x4005 C018 (SPI1)) 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. **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. **Remark:** The active state of the SSEL1 pin is configured by bits in the CFG register.	0
		0	SSEL1 asserted.	
		1	SSEL1 not asserted.	
18	TXSSEL2_N		Transmit Slave Select. This field asserts SSEL2 in master mode. The output on the pin is active LOW by default. **Remark:** The active state of the SSEL2 pin is configured by bits in the CFG register.	0
		0	SSEL2 asserted.	
		1	SSEL2 not asserted.	
19	TXSSEL3_N		Transmit Slave Select. This field asserts SSEL3 in master mode. The output on the pin is active LOW by default. **Remark:** The active state of the SSEL3 pin is configured by bits in the CFG register.	0
		0	SSEL3 asserted.	
		1	SSEL3 not asserted.	

Table 196. SPI Transmitter Data and Control register (TXDATCTL, addresses 0x4005 8018 (SPI0), 0x4005 C018 (SPI1)) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
20	EOT		End of Transfer. The asserted SSEL will be deasserted at the end of a transfer, and remain so for at least the time specified by the Transfer_delay value in the DLY register.	0
		0	SSEL not deasserted. This piece of data is not treated as the end of a transfer. SSEL will not be deasserted at the end of this data.	
		1	SSEL deasserted. This piece of data is treated as the end of a transfer. SSEL will be deasserted at the end of this piece of data.	
21	EOF		End of Frame. Between frames, a delay may be inserted, as defined by the FRAME_DELAY value in the DLY register. The end of a frame may not be particularly meaningful if the FRAME_DELAY value = 0. This control can be used as part of the support for frame lengths greater than 16 bits.	0
		0	Data not EOF. This piece of data transmitted is not treated as the end of a frame.	
		1	Data EOF. This piece of data is treated as the end of a frame, causing the FRAME_DELAY time to be inserted before subsequent data is transmitted.	
22	RXIGNORE		Receive Ignore. This allows data to be transmitted using the SPI without the need to read unneeded data from the receiver.Setting this bit simplifies the transmit process and can be used with the DMA.	0
		0	Read received data. Received data must be read in order to allow transmission to progress. In slave mode, an overrun error will occur if received data is not read before new data is received.	
		1	Ignore received data. Received data is ignored, allowing transmission without reading unneeded received data. No receiver flags are generated.	
23	-		Reserved. Read value is undefined, only zero should be written.	NA
27:24	LEN		Data Length. Specifies the data length from 1 to 16 bits. Note that transfer lengths greater than 16 bits are supported by implementing multiple sequential transmits. 0x0 = Data transfer is 1 bit in length. 0x1 = Data transfer is 2 bits in length. 0x2 = Data transfer is 3 bits in length. ... 0xF = Data transfer is 16 bits in length.	0x0
31:28	-		Reserved. Read value is undefined, only zero should be written.	NA

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 (SPI0), 0x4005 C01C (SPI1)) 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 (SPI0), 0x4005 C020 (SPI1)) bit description

Bit	Symbol	Description	Reset value
15:0	-	Reserved. Read value is undefined, only zero should be written.	NA
16	TXSSEL0_N	Transmit Slave Select 0.	0x0
17	TXSSEL1_N	Transmit Slave Select 1.	0x0
18	TXSSEL2_N	Transmit Slave Select 2.	0x0
19	TXSSEL3_n	Transmit Slave Select 3.	0x0
20	EOT	End of Transfer.	0
21	EOF	End of Frame.	0
22	RXIGNORE	Receive Ignore.	0
23	-	Reserved. Read value is undefined, only zero should be written.	NA
27:24	LEN	Data transfer Length.	0x0
31:28	-	Reserved. Read value is undefined, only zero should be written.	NA

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 (SPI0), 0x4005 C024 (SPI1)) bit description

Bit	Symbol	Description	Reset Value
15:0	DIVVAL	Rate divider value. Specifies how the PCLK for the SPI is divided to produce the SPI clock rate in master mode. DIVVAL is -1 encoded such that the value 0 results in PCLK/1, the value 1 results in PCLK/2, up to the maximum possible divide value of 0xFFFF, which results in PCLK/65536.	0
31:16	-	Reserved. Read value is undefined, only zero should be written.	NA

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 (SPI0), 0x4005 C028 (SPI1)) 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
31:6	-	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

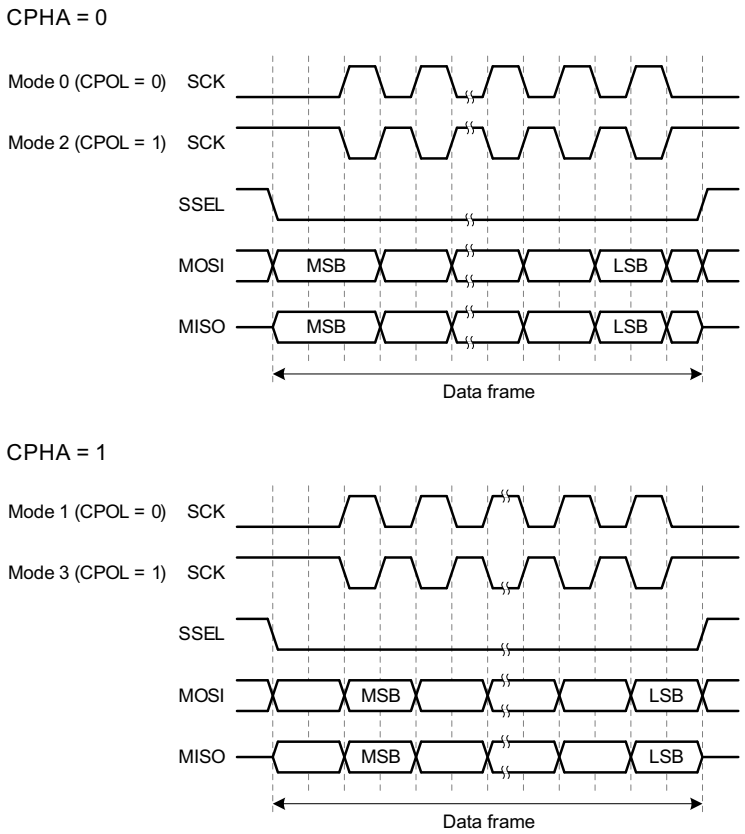


Fig 25. Basic SPI operating modes

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.

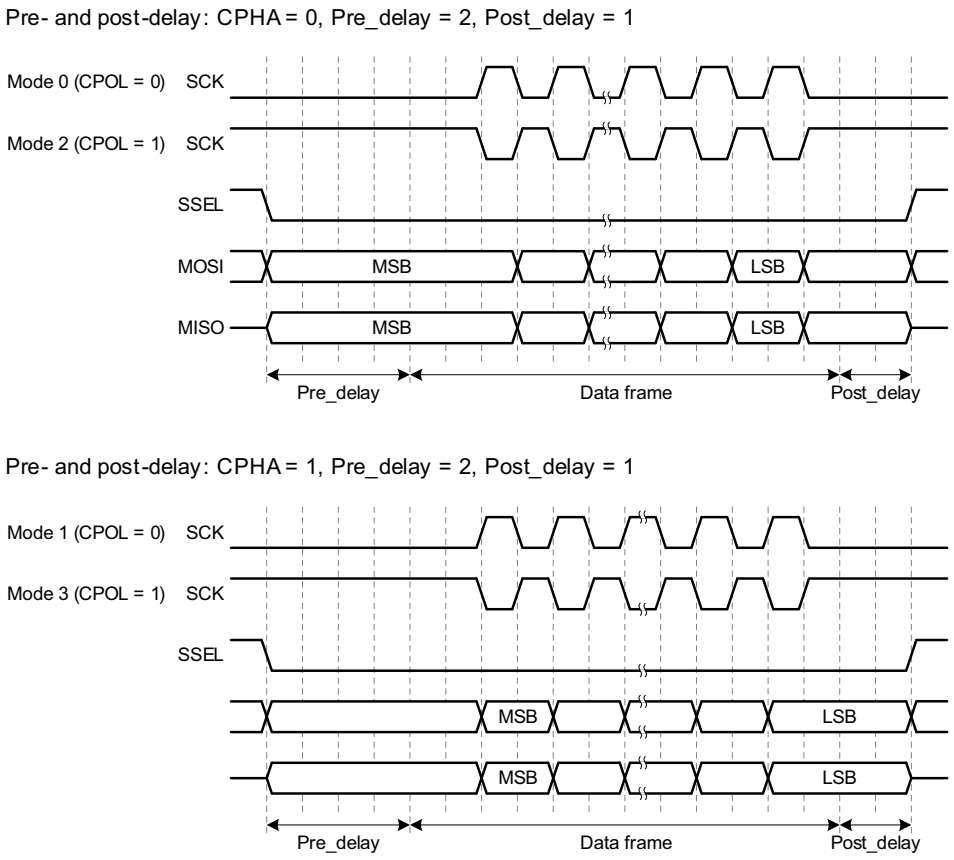


Fig 26. Pre_delay and Post_delay

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 14.7.6 for more information.

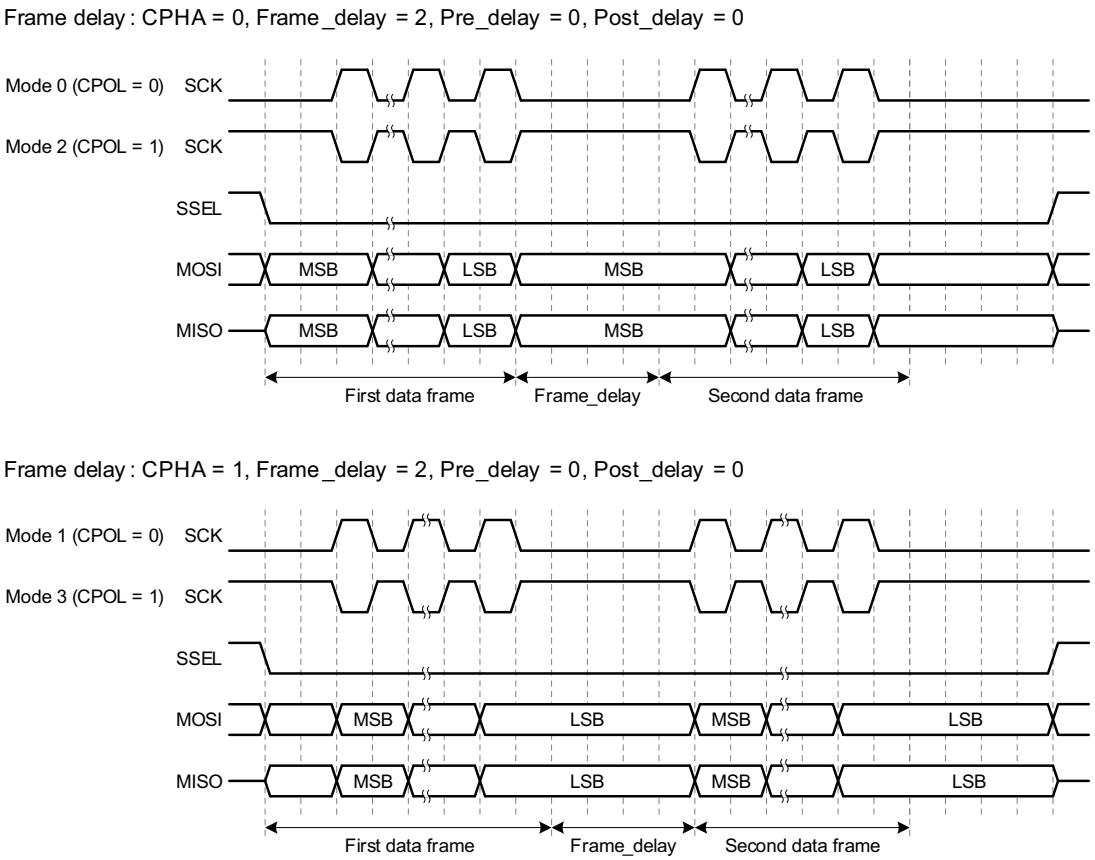


Fig 27. Frame_delay

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.

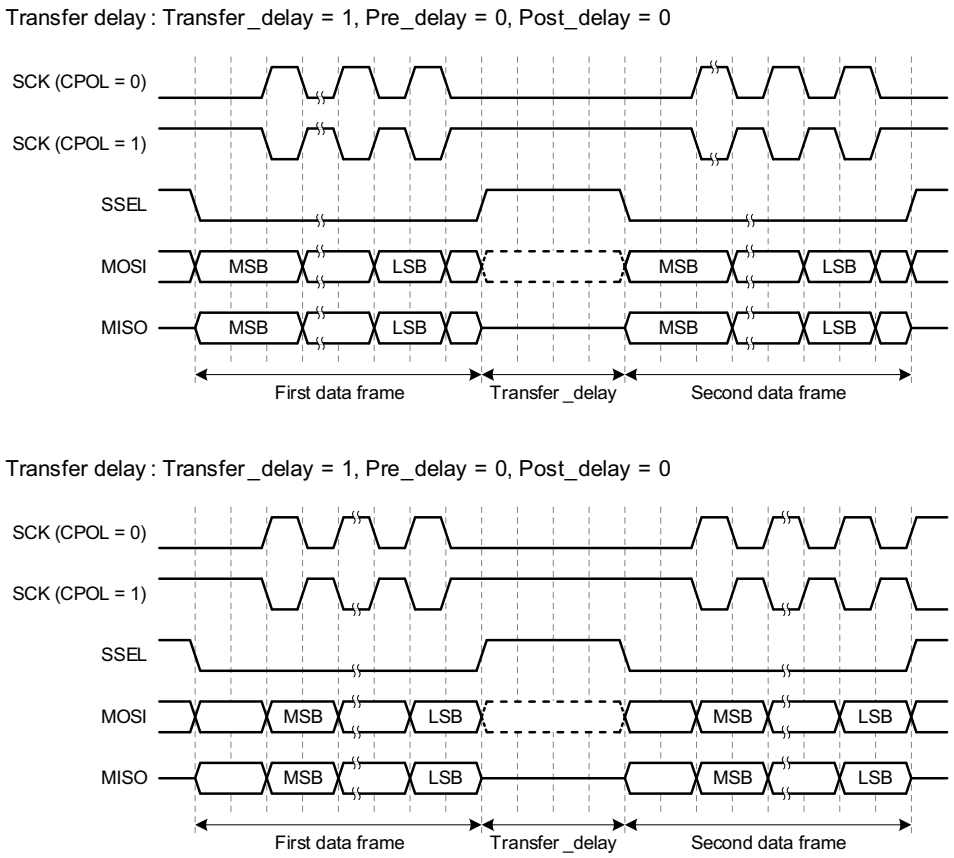


Fig 28. Transfer_delay

14.7.3 Clocking and data rates

In order to use the SPI, clocking details must be defined. This includes configuring the system clock 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 PCLK, or at lower integer divide rates. The SPI rate will be = PCLK_SPIn / DIVVAL.

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 four Slave Select inputs in slave mode or outputs in master mode. Each SSEL can be set for normal polarity (active low), or can be inverted (active high). Representation of the 4 SSELs in a register is always active low. If an SSEL is inverted, this is done as the signal leaves/enters the SPI block.

In slave mode, **any** asserted SSEL that is connected to a pin will activate the SPI. In master mode, all SSELs that are connected to a pin will be output as defined in the SPI registers. In the latter case, the SSELs could potentially be decoded externally in order to address more than four slave devices. Note that at least one SSEL is asserted when data is transferred in master mode.

In master mode, Slave Selects come from the SSELN field, which appears in both the CTL and DATCTL registers. In slave mode, the state of all four SSELs is saved along with received data in the RXSSEL_N field of the RXDAT register.

14.7.5 DMA operation

A DMA request is provided for each SPI direction, and can be used in lieu of interrupts for transferring data by configuring the DMA controller appropriately, and enabling the Rx and/or Tx DMA via the CFG register. The DMA controller provides an acknowledgement signal that clears the related request when it completes handling that request.

The transmitter DMA request is asserted when Tx DMA is enabled and the transmitter can accept more data.

The receiver DMA request is asserted when Rx DMA is enabled and received data is available to be read.

14.7.6 Data lengths greater than 16 bits

The SPI interface handles data frame sizes from 1 to 16 bits directly. Larger sizes can be handled by splitting data up into groups of 16 bits or less. For example, 24 bits can be supported as 2 groups of 16 bits and 8 bits or 2 groups of 12 bits, among others. Frames of any size, including greater than 32 bits, can supported in the same way.

Details of how to handle larger data widths depend somewhat on other SPI configuration options. For instance, if it is intended for Slave Selects to be 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.

14.7.7 Data stalls

A stall for Master transmit data can happen in modes 0 and 2 when SCK cannot be returned to the rest state until the MSB of the next data frame can be driven on MOSI. In this case, the stall happens just before the final clock edge of data if the next piece of data is not yet available.

A stall for Master receive can happen when a receiver overrun would otherwise occur if the transmitter was not stalled. In modes 0 and 2, this occurs if the previously received data is not read before the end of the next piece of is received. This stall happens one clock edge earlier than the transmitter stall.

In modes 1 and 3, the same kind of receiver stall can occur, but just before the final clock edge of the received data. Also, a transmitter stall will not happen in modes 1 and 3 because the transmitted data is complete at the point where a stall would otherwise occur, so it is not needed.

Stalls are reflected in the STAT register by the Stalled status flag, which indicates the current SPI status.

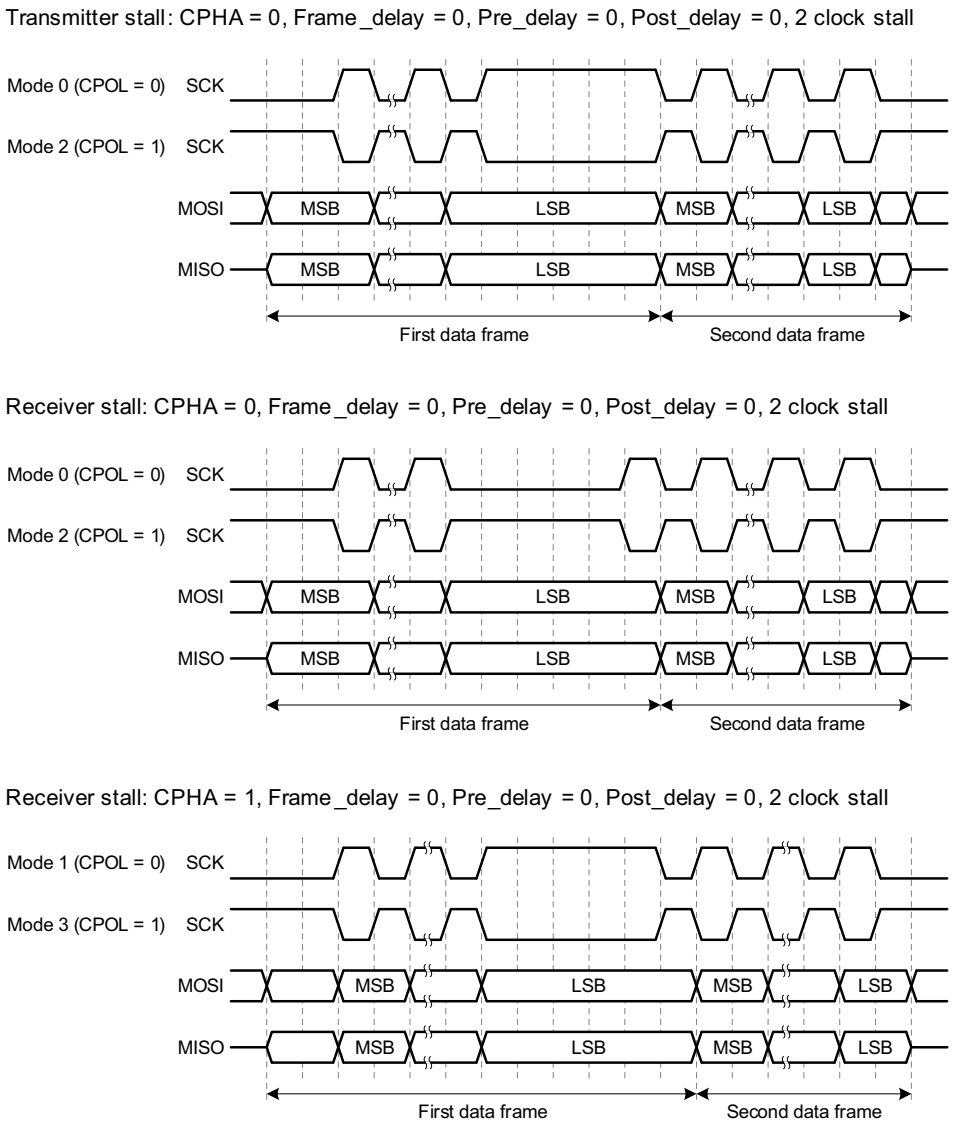


Fig 29. Examples of data stalls

15.1 How to read this chapter

Four I2C interfaces are available on all parts depending on the switch matrix configuration.

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.

The LPC82x provides an on-chip ROM-based I2C API to configure and operate the I2C.

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 SMBus.
- Supports the I^2C-bus specification up to Fast-mode Plus (up to 1 MHz).

15.3 Basic configuration

Configure the I2C interfaces using the following registers:

- In the SYSAHBCLKCTRL register, set the corresponding bits to enable the clocks to the register interfaces. See Table 35.
- Clear the I2C peripheral resets using the PRESETCTRL register (Table 23).
- Enable/disable the I2C interrupt in interrupt slots #7, 8, 21, 22 in the NVIC. See Table 5.
- Configure the I2C pin functions through the switch matrix. See Table 202.
- The peripheral clock for the I2C is the system clock (see Figure 30).

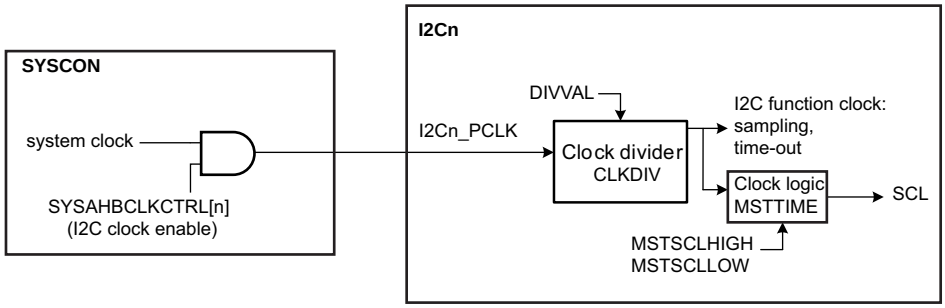


Fig 30. I2C clocking

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 30 MHz and the bit rate is approximately 400 kHz. You must enable the I2C0_SCL and I2C0_SDA functions on pins PIO0_11 and PIO0_10 or assign the SCL and SDA functions for any of the other I2C blocks to pins through the switch matrix. See Table 202.

For a 400 kHz bit rate, the I2C0 pins can be configured in standard mode in the IOCON block. See Table 90 “PIO0_11 register (PIO0_11, address 0x4004 401C) bit description” and Table 91 “PIO0_10 register (PIO0_10, address 0x4004 4020) bit description”.

The transmission of the address and data bits is controlled by the state of the MSTPENDING status bit. Whenever the status is Master pending, the master can read or write to the MSTDAT register and go to the next step of the transmission protocol by writing to the MSTCTL register.

Configure the I2C bit rate:

- Divide the system clock (I2C_PCLK) by a factor of 2. See Table 211 “I2C Clock Divider register (CLKDIV, address 0x4005 0014 (I2C0), 0x4005 4014 (I2C1), 0x4007 0014 (I2C2), 0x4007 4014 (I2C3)) 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), 0x4007 0024 (I2C2), 0x4007 4024 (I2C3)) 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 30 MHz and the bit rate is approximately 400 kHz. You must enable the I2C0_SCL and I2C0_SDA functions on pins PIO0_11 and PIO0_10 or assign the SCL and SDA functions for any of the other I2C blocks to pins through the switch matrix. See Table 202.

For a 400 kHz bit rate, the pins can be configured in standard mode in the IOCON block. See Table 90 “PIO0_11 register (PIO0_11, address 0x4004 401C) bit description” and Table 91 “PIO0_10 register (PIO0_10, address 0x4004 4020) bit description”.

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 I2C 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 I2C_PCLK remains active in sleep mode, the I2C can wake up the part independently of whether the I2C block is configured in master or slave mode.

In Deep-sleep or Power-down mode, the I2C clock is turned off as are all peripheral clocks. However, if the I2C is configured in slave mode and an external master on the I2C-bus provides the clock signal, the I2C block can create an interrupt asynchronously. This interrupt, if enabled in the NVIC and in the I2C block's INTENCLR register, can then wake up the core.

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 51 "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

The I2C0 pins are fixed-pin functions and enabled through the switch matrix.

If the I2C0-bus interface is used in Fast-mode Plus mode, configure the I2C-pins for this mode in the IOCON block: Table 90 “PIO0_11 register (PIO0_11, address 0x4004 401C) bit description” and Table 91 “PIO0_10 register (PIO0_10, address 0x4004 4020) bit description”.

Pins for the I2C1/2/3 interfaces are movable functions and can be assigned to any pin. However, except for PIO0_10 and PIO0_11, 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	PIO0_11	PINENABLE0	Table 79	I2C0 serial data.
I2C0_SCL	I/O	external to pin	PIO0_10	PINENABLE0	Table 79	I2C0 serial clock.
I2C1_SDA	I/O	external to pin	any pin	PINASSIGN9	Table 76	I2C1 serial data.
I2C1_SCL	I/O	external to pin	any pin	PINASSIGN9	Table 76	I2C1 serial clock.
I2C2_SDA	I/O	external to pin	any pin	PINASSIGN9	Table 76	I2C2 serial data.
I2C2_SCL	I/O	external to pin	any pin	PINASSIGN10	Table 77	I2C2 serial clock.
I2C3_SDA	I/O	external to pin	any pin	PINASSIGN10	Table 77	I2C3 serial data.
I2C3_SCL	I/O	external to pin	any pin	PINASSIGN10	Table 77	I2C3 serial clock.

15.5 General description

The architecture of the I2C-bus interface is shown in Figure 31.

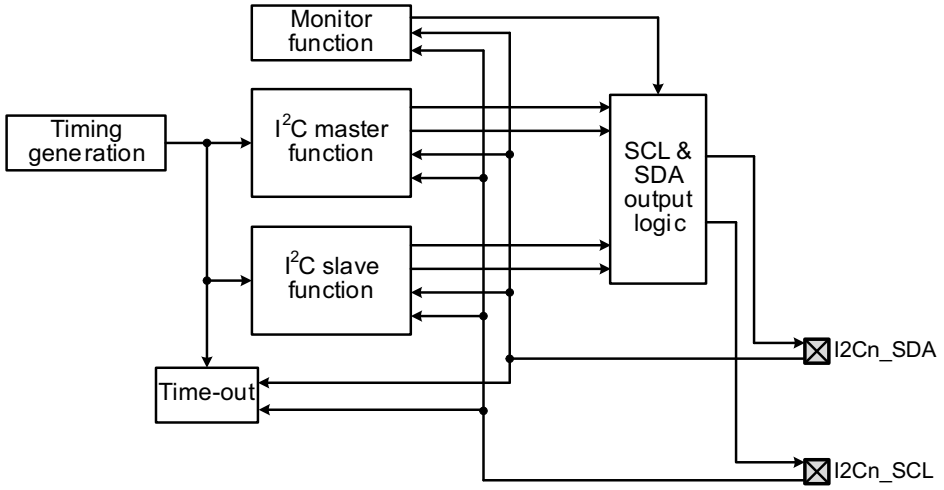


Fig 31. I2C block diagram

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), 0x4007 0000 (I2C2), 0x4007 4000 (I2C3)) bit description”
  - Table 205 “I2C Status register (STAT, address 0x4005 0004 (I2C0), 0x4005 4004 (I2C1), 0x4007 0004 (I2C2), 0x4007 4004 (I2C3)) bit description”
  - Table 212 “I2C Interrupt Status register (INTSTAT, address 0x4005 0018 (I2C0), 0x4005 4018 (I2C1), 0x4007 0018 (I2C2), 0x4007 4018 (I2C3)) bit description”
  - Table 208 “Interrupt Enable Set and read register (INTENSET, address 0x4005 0008 (I2C0), 0x4005 4008 (I2C1), 0x4007 0008 (I2C2), 0x4007 4008 (I2C3)) bit description”
  - Table 209 “Interrupt Enable Clear register (INTENCLR, address 0x4005 000C (I2C0), 0x4005 400C (I2C1), 0x4007 000C (I2C2), 0x4007 400C (I2C3)) bit description”
  - Table 210 “Time-out value register (TIMEOUT, address 0x4005 0010 (I2C0), 0x4005 4010 (I2C1), 0x4007 0010 (I2C2), 0x4007 4010 (I2C3)) bit description”
  - Table 211 “I2C Clock Divider register (CLKDIV, address 0x4005 0014 (I2C0), 0x4005 4014 (I2C1), 0x4007 0014 (I2C2), 0x4007 4014 (I2C3)) bit description”
- Master function registers:
  - Table 213 “Master Control register (MSTCTL, address 0x4005 0020 (I2C0), 0x4005 4020 (I2C1), 0x4007 0020 (I2C2), 0x4007 4020 (I2C3)) bit description”
  - Table 214 “Master Time register (MSTTIME, address 0x4005 0024 (I2C0), 0x4005 4024 (I2C1), 0x4007 0024 (I2C2), 0x4007 4024 (I2C3)) bit description”
  - Table 215 “Master Data register (MSTDAT, address 0x4005 0028 (I2C0), 0x4005 4028 (I2C1), 0x4007 0028 (I2C2), 0x4007 4028 (I2C3)) bit description”
- Slave function registers:

- Table 216 “Slave Control register (SLVCTL, address 0x4005 0040 (I2C0), 0x4005 4040 (I2C1), 0x4007 0040 (I2C2), 0x4007 4040 (I2C3)) bit description”
- Table 216 “Slave Control register (SLVCTL, address 0x4005 0040 (I2C0), 0x4005 4040 (I2C1), 0x4007 0040 (I2C2), 0x4007 4040 (I2C3)) 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), 0x4007 0048 (SLVADR0) to 0x4007 0054 (SLVADR3) (I2C2), 0x4007 4048 (SLVADR0) to 0x4007 4054 (SLVADR3) (I2C3)) bit description”
- Table 219 “Slave address Qualifier 0 register (SLVQUAL0, address 0x4005 0058 (I2C0), 0x4005 4058 (I2C1), 0x4007 0058 (I2C2), 0x4007 4058 (I2C3)) bit description”

- Monitor function register: Table 220 “Monitor data register (MONRXDAT, address 0x4005 0080 (I2C0), 0x4005 4080 (I2C1), 0x4007 0080 (I2C2), 0x4007 4080 (I2C3)) bit description”

Table 203. Register overview: I2C (base address 0x4005 0000 (I2C0), 0x4005 4000 (I2C1), 0x4007 0000 (I2C2), 0x4007 4000 (I2C3))

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 I^2C block. This determines what time increments are used for the MSTTIME and SLVTIME registers.	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), 0x4007 0000 (I2C2), 0x4007 4000 (I2C3)) bit description

Bit	Symbol	Value	Description	Reset Value
0	MSTEN		Master Enable. When disabled, configurations settings for the Master function are not changed, but the Master function is internally reset.	0
		0	Disabled. The I^2C Master function is disabled.	
		1	Enabled. The I^2C Master function is enabled.	
1	SLVEN		Slave Enable. When disabled, configurations settings for the Slave function are not changed, but the Slave function is internally reset.	0
		0	Disabled. The I^2C slave function is disabled.	
		1	Enabled. The I^2C slave function is enabled.	

Table 204. I2C Configuration register (CFG, address 0x4005 0000 (I2C0), 0x4005 4000 (I2C1), 0x4007 0000 (I2C2), 0x4007 4000 (I2C3)) bit description

Bit	Symbol	Value	Description	Reset Value
2	MONEN		Monitor Enable. When disabled, configurations settings for the Monitor function are not changed, but the Monitor function is internally reset.	0
		0	Disabled. The I^2C monitor function is disabled.	
		1	Enabled. The I^2C monitor function is enabled.	
3	TIMEOUTEN		I^2C bus Time-out Enable. When disabled, the time-out function is internally reset.	0
		0	Disabled. Time-out function is disabled.	
		1	Enabled. Time-out function is enabled. Both types of time-out flags will be generated and will cause interrupts if they are enabled. Typically, only one time-out will be used in a system.	
4	MONCLKSTR		Monitor function Clock Stretching.	0
		0	Disabled. The monitor function will not perform clock stretching. Software or DMA may not always be able to read data provided by the monitor function before it is overwritten. This mode may be used when non-invasive monitoring is critical.	
		1	Enabled. The monitor function will perform clock stretching in order to ensure that software or DMA can read all incoming data supplied by the monitor function.	
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 I^2C 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. I^2C Status register (STAT, address 0x4005 0004 (I2C0), 0x4005 4004 (I2C1), 0x4007 0004 (I2C2), 0x4007 4004 (I2C3)) 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. The MSTPENDING flag is not set when the DMA is handling an event (if the MSTDMA bit in the MSTCTL register is set). If the master is in the idle state, and no communication is needed, mask this interrupt.	1	RO
		0	In progress. Communication is in progress and the Master function is busy and cannot currently accept a command.		
		1	Pending. The Master function needs software service or is in the idle state. If the master is not in the idle state, it is waiting to receive or transmit data or the NACK bit.		
3:1	MSTSTATE		Master State code. The master state code reflects the master state when the MSTPENDING bit is set, that is the master is pending or in the idle state. Each value of this field indicates a specific required service for the Master function. All other values are reserved.	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. 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), 0x4007 0004 (I2C2), 0x4007 4004 (I2C3)) 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. 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 not set when the DMA is handling an event (if the SLVDMA bit in the SLVCTL register is set). The SLVPENDING flag is read-only and is automatically cleared when a 1 is written to the SLVCONTINUE bit in the 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.		

Table 205. I²C Status register (STAT, address 0x4005 0004 (I2C0), 0x4005 4004 (I2C1), 0x4007 0004 (I2C2), 0x4007 4004 (I2C3)) bit description *…continued*

Bit	Symbol	Value	Description	Reset value	Access
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.		
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.		

Table 205. I²C Status register (STAT, address 0x4005 0004 (I2C0), 0x4005 4004 (I2C1), 0x4007 0004 (I2C2), 0x4007 4004 (I2C3)) bit description *…continued*

Bit	Symbol	Value	Description	Reset value	Access
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
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	DMA access allowed
0x0	**Idle.** The Master function is available to be used for a new transaction.	Send a Start or disable MSTPENDING interrupt if the Master function is not needed currently.	No
0x1	**Received data is available (Master Receiver mode).** Address plus Read was previously sent and Acknowledged by slave.	Read data and either continue, send a Stop, or send a Repeated Start.	Yes
0x2	**Data can be transmitted (Master Transmitter mode).** Address plus Write was previously sent and Acknowledged by slave.	Send data and continue, or send a Stop or Repeated Start.	Yes
0x3	**Slave NACKed address.**	Send a Stop or Repeated Start.	No
0x4	**Slave NACKed transmitted data.**	Send a Stop or Repeated Start.	No

Table 207. Slave function state codes (SLVSTATE)

SLVSTATE		Description	Actions	DMA access allowed
0	SLVST_ADDR	**Address plus R/W received.** At least one of the 4 slave addresses has been matched by hardware.	Software can further check the address if needed, for instance if a subset of addresses qualified by SLVQUAL0 is to be used. Software can ACK or NACK the address by writing 1 to either SLVCONTINUE or SLVNACK. Also see Section 15.7.3 regarding 10-bit addressing.	No
1	SLVST_RX	**Received data is available (Slave Receiver mode).**	Read data reply with an ACK or a NACK.	Yes
2	SLVST_TX	**Data can be transmitted (Slave Transmitter mode).**	Send data.	Yes
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 4008 (I2C1), 0x4007 0008 (I2C2), 0x4007 4008 (I2C3)) 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

Table 208. Interrupt Enable Set and read register (INTENSET, address 0x4005 0008 (I2C0), 0x4005 4008 (I2C1), 0x4007 0008 (I2C2), 0x4007 4008 (I2C3)) bit description

Bit	Symbol	Value	Description	Reset value
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
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), 0x4007 000C (I2C2), 0x4007 400C (I2C3)) 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 I^2C 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), 0x4007 0010 (I2C2), 0x4007 4010 (I2C3)) bit description

Bit	Symbol	Description	Reset value
3:0	TOMIN	Time-out time value, bottom four bits. These are hard-wired to 0xF. This gives a minimum time-out of 16 I^2C function clocks and also a time-out resolution of 16 I^2C function clocks.	0xF
15:4	TO	Time-out time value. Specifies the time-out interval value in increments of 16 I^2C function clocks, as defined by the CLKDIV register. To change this value while I^2C is in operation, disable all time-outs, write a new value to TIMEOUT, then re-enable time-outs. 0x000 = A time-out will occur after 16 counts of the I^2C function clock. 0x001 = A time-out will occur after 32 counts of the I^2C function clock. ... 0xFFF = A time-out will occur after 65,536 counts of the I^2C 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 Peripheral Clock (PCLK) to produce the I^2C function clock that is used to time various aspects of the I^2C interface. The I^2C function clock is used for some internal operations in the I^2C block and to generate the timing required by the I^2C bus specification, some of which are user configured in the MSTTIME register for Master operation and the SLVTIME register for Slave operation.

See Section 15.7.1.1 “Rate calculations” for details on bus rate setup.

Table 211. I^2C Clock Divider register (CLKDIV, address 0x4005 0014 (I2C0), 0x4005 4014 (I2C1), 0x4007 0014 (I2C2), 0x4007 4014 (I2C3)) bit description

Bit	Symbol	Description	Reset value
15:0	DIVVAL	This field controls how the clock (PCLK) is used by the I^2C functions that need an internal clock in order to operate. 0x0000 = PCLK is used directly by the I^2C function. 0x0001 = PCLK is divided by 2 before use by the I^2C function. 0x0002 = PCLK is divided by 3 before use by the I^2C function. ... 0xFFFF = PCLK is divided by 65,536 before use by the I^2C 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), 0x4007 0018 (I2C2), 0x4007 4018 (I2C3)) 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. An exception is when DMA is being used and a transfer completes. In this case there is no

MSTPENDING flag, and the MSTDMA control bit would be cleared by software potentially at the same time as setting either the MSTSTOP or MSTSTART control bit.

Remark: When in the idle or slave NACKed states (see Table 206), set the MSTDMA bit either with or after the MSTCONTINUE bit. MSTDMA can be cleared at any time.

Table 213. Master Control register (MSTCTL, address 0x4005 0020 (I2C0), 0x4005 4020 (I2C1), 0x4007 0020 (I2C2), 0x4007 4020 (I2C3)) 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).	
3	MSTDMA		Master DMA enable. Data operations of the I^2C can be performed with DMA. Protocol type operations such as Start, address, Stop, and address match must always be done with software, typically via an interrupt. When a DMA data transfer is complete, MSTDMA must be cleared prior to beginning the next operation, typically a Start or Stop.This bit is read/write.	0
		0	Disable. No DMA requests are generated for master operation.	
		1	Enable. A DMA request is generated for I^2C master data operations. When this I^2C master is generating Acknowledge bits in Master Receiver mode, the acknowledge is generated automatically.	
31:4	-		Reserved. Read value is undefined, only zero should be written.	NA

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 (I2C0), 0x4005 4024 (I2C1), 0x4007 0024 (I2C2), 0x4007 4024 (I2C3)) 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.	0
		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.	
3	-		Reserved.	0

Table 214. Master Time register (MSTTIME, address 0x4005 0024 (I2C0), 0x4005 4024 (I2C1), 0x4007 0024 (I2C2), 0x4007 4024 (I2C3)) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
6:4	MSTSCLHIGH		Master SCL High time. Specifies the minimum high time that will be asserted by this master on SCL. Other masters in a multi-master system could shorten this time. This corresponds to the parameter t_{HIGH} in the I^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.	0
		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 (I2C0), 0x4005 4028 (I2C1), 0x4007 0028 (I2C2), 0x4007 4028 (I2C3)) bit description

Bit	Symbol	Description	Reset value
7:0	DATA	Master function data register. Read: read the most recently received data for the Master function. Write: transmit data using the Master function.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

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).

Remark: When in the slave address state (slave state 0, see Table 207), set the SLVDMA bit either with or after the SLVCONTINUE bit. SLVDMA can be cleared at any time.

Table 216. Slave Control register (SLVCTL, address 0x4005 0040 (I2C0), 0x4005 4040 (I2C1), 0x4007 0040 (I2C2), 0x4007 4040 (I2C3)) 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.	
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
3	SLVDMA		Slave DMA enable.	0
		0	Disabled. No DMA requests are issued for Slave mode operation.	
		1	Enabled. DMA requests are issued for I^2C slave data transmission and reception.	
31:4	-		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 (I2C0), 0x4005 4044 (I2C1), 0x4007 0044 (I2C2), 0x4007 4044 (I2C3)) bit description

Bit	Symbol	Description	Reset Value
7:0	DATA	Slave function data register. Read: read the most recently received data for the Slave function. Write: transmit data using the Slave function.	0
31:8	-	Reserved. Read value is undefined, only zero should be written.	NA

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) (I2C0), 0x4005 4048 (SLVADR0) to 0x4005 4054 (SLVADR3) (I2C1), 0x4007 0048 (SLVADR0) to 0x4007 0054 (SLVADR3) (I2C2), 0x4007 4048 (SLVADR0) to 0x4007 4054 (SLVADR3) (I2C3)) 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 (I2C0), 0x4005 4058 (I2C1), 0x4007 0058 (I2C2), 0x4007 4058 (I2C3)) 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. If QUALMODE0 = 0, any bit in this field which is set to 1 will cause an automatic match of the corresponding bit of the received address when it is compared to the SLVADR0 register. If QUALMODE0 = 1, an address range is matched for address 0. This range extends from the value defined by SLVADR0 to the address defined by SLVQUAL0 (address matches when SLVADR0[7:1] <= 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 or DMA response time). In order to improve the chance of collecting all Monitor information if clock stretching is not enabled, Monitor data is buffered such that it is available until the end of the next piece of information from the I^2C bus.

Table 220. Monitor data register (MONRXDAT, address 0x4005 0080 (I2C0), 0x4005 4080 (I2C1), 0x4007 0080 (I2C2), 0x4007 4080 (I2C3)) 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

Table 220. Monitor data register (MONRXDAT, address 0x4005 0080 (I2C0), 0x4005 4080 (I2C1), 0x4007 0080 (I2C2), 0x4007 4080 (I2C3)) bit description

Bit	Symbol	Value	Description	Reset value
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.	
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 (i.e. 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)

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 I2C is simply instructed to perform the 2-byte addressing as a normal write operation, followed either by more write data, or by a Repeated Start with a repeat of the first part of the 10-bit slave address and then reading in the normal fashion.

For the Slave function, the first part of the address is automatically matched in the same fashion as 7-bit addressing. The Slave address qualifier feature (see Section 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.

15.7.6 DMA

Generally, data transfers can be handled by DMA for Master mode after an address is sent and acknowledged by a slave, and for Slave mode after software has acknowledged an address. In either mode, software is always involved in the address portion of a message. In master and slave modes, data receive and transmit data can be transferred by the DMA. The DMA supports three DMA requests: data transfer in master mode, slave mode, and monitor mode.

16.1 How to read this chapter

The SCTimer/PWM is available on all parts.

Remark: For a detailed description of SCTimer/PWM applications and code examples, see Ref. 5 “AN11538”.

16.2 Features

- The SCTimer/PWM supports:
  - Eight match/capture registers.
  - Eight events.
  - Eight states.
  - Four inputs with multiple connection options through the input MUX.
  - Six outputs.
- Counter/timer features:
  - Each SCTimer is configurable as two 16-bit counters or one 32-bit counter.
  - Counters clocked by system clock or selected input.
  - Configurable as up counters or up-down counters.
  - Configurable number of match and capture registers. Up to five match and capture registers total.
  - Upon match and/or an input or output transition create the following events: interrupt; stop, limit, halt the timer or change counting direction; toggle outputs; change the state.
  - Counter value can be loaded into capture register triggered by a match or input/output toggle.
- PWM features:
  - Counters can be used in conjunction with match registers to toggle outputs and create time-proportioned PWM signals.
  - Up to 6 single-edge or dual-edge PWM outputs with independent duty cycle and common PWM cycle length.
- Event creation features:
  - The following conditions define an event: a counter match condition, an input (or output) condition such as an rising or falling edge or level, a combination of match and/or input/output condition.
  - Selected events can limit, halt, start, or stop a counter or change its direction.
  - Events trigger state changes, output toggles, interrupts, and DMA transactions.
  - Match register 0 can be used as an automatic limit.
  - In bi-directional mode, events can be enabled based on the count direction.
  - Match events can be held until another qualifying event occurs.

- State control features:
  - A state is defined by events that can happen in the state while the counter is running.
  - A state changes into another state as a result of an event.
  - Each event can be assigned to one or more states.
  - State variable allows sequencing across multiple counter cycles.

16.3 Basic configuration

Configure the SCT as follows:

- Enable the clock to the SCTimer/PWM (SCT) in the SYSAHBCLKCTRL register (Table 35) to enable the register interface and the peripheral clock.
- Clear the SCT peripheral reset using the PRESETCTRL register (Table 23).
- The SCT uses interrupt in slot #9 in the NVIC (Table 5).
- Use the INPUT MUX to connect the SCT inputs to external pins. See Table 221.
- Use the switch matrix registers to connect the SCT outputs to external pins. See Table 221.
- The SCT DMA request lines are connected to the DMA trigger inputs via the DMA_ITRIG_PINMUX registers. See Table 145 “DMA input trigger Input mux registers 0 to 17 (DMA_ITRIG_INMUX[0:17], address 0x4002 80E0 (DMA_ITRIG_INMUX0) to 0x4002 8124 (DMA_ITRIG_INMUX17)) bit description”.

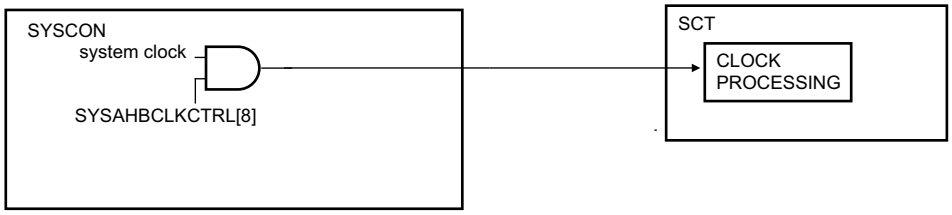
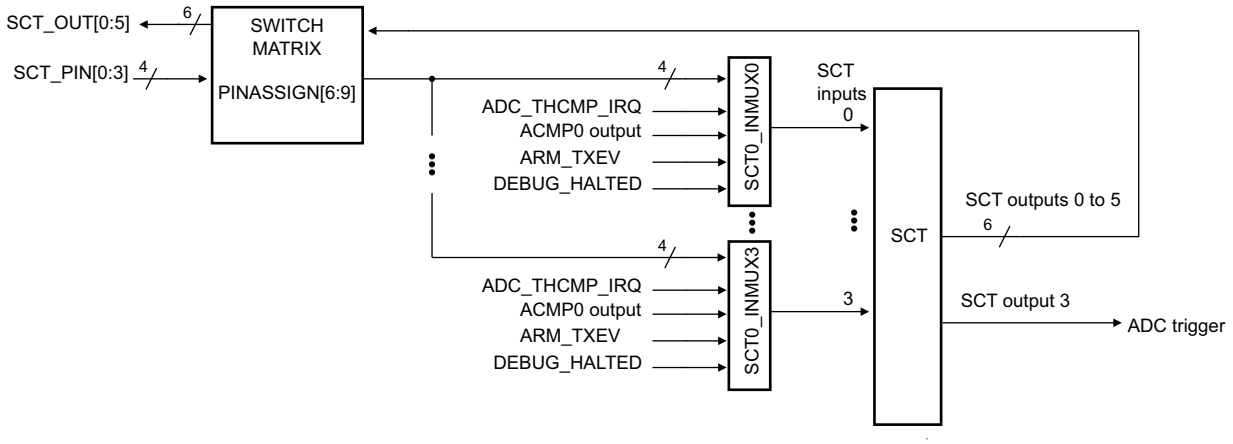


Fig 32. SCT clocking



See Table 147 “SCT input mux registers 0 to 3 (SCT0_INMUX[0:3], address 0x4002 C020 (SCT0_INMUX0) to 0x4002 C02C (SCT0_INMUX3)) bit description” to configure the SCTimer/PWM input multiplexers.

Fig 33. SCT connections

16.4 Pin description

Table 221. SCT pin description

Function	Type	Connect to	Use register	Reference	Description
SCT input 0, SCT input 1, SCT input 2, SCT input 3,	external to pins or internal	one of the following: SCT_PIN[0:3] through the switch matrix, or internally to the analog comparator output, ADC threshold compare interrupt, ARM TXEV	SCT0_INMUX[0:3]	Table 147	SCT capture and event inputs.
SCT_OUT0	external to pins	pins through the switch matrix	PINASSIGN7	Table 74	SCT match and PWM output.
SCT_OUT1, SCT_OUT2, SCT_OUT3, SCT_OUT4	external to pins and internal	pins through the switch matrix; to ADC trigger	PINASSIGN8	Table 75	SCT match and PWM outputs. SCT_OUT3 can be selected as ADC input trigger.
SCT_OUT5	external to pin	pin through the switch matrix	PINASSIGN9	Table 75	SCT match and PWM output.

16.5 General description

The SCTimer/PWM, or in short SCT, is a powerful, flexible timer module capable of creating complex PWM waveforms and performing other advanced timing and control operations with minimal or no CPU intervention.

The SCT can operate as a single 32-bit counter or as two independent, 16-bit counters in uni-directional or bi-directional mode. As with most timers, the SCT supports a selection of match registers against which the count value can be compared, and capture registers where the current count value can be recorded when some pre-defined condition is detected.

An additional feature contributing to the versatility of the SCT is the concept of “events”. The SCT module supports multiple separate events that can be defined by the user based on some combination of parameters including a match on one of the match registers, and/or a transition on one of the SCT inputs or outputs, the direction of count, and other factors.

Every action that the SCT block can perform occurs in direct response to one if these user-defined events without any software overhead. Any event can be enabled to:

- Start, stop, or halt the counter.
- Limit the counter which means to clear the counter in uni-directional mode or change its direction in bi-directional mode.
- Set, clear, or toggle any SCT output.
- Force a capture of the count value into any capture registers.
- Generate an interrupt of DMA request.

Regarding states, maybe include something like this:

The SCT allows the user to group and filter events, thereby selecting some events to be enabled together while others are disabled. A group of enabled and disabled events can be described as a state, and several states with different sets of enabled and disabled events are allowed. Changing from one state to another is event driven as well and can therefore happen without software intervention. By defining these states, the SCTimer/PWM provides the means to run entire state machines in hardware with any desired level of complexity to accomplish complex waveform and timing tasks.

In a simple system such as a classical timer/counter with capture and match capabilities. all events that could cause the timer to capture the timer value or toggle a match output are enabled and remain enabled at all times while the counter is running. In this case, no events are filtered and the system is described by one state that does not change. This is the default configuration of the SCT.

In a more complex system, two states could be set up that allow some events in one state and not in the other. An event itself, enabled in both states, can then be used, to move from one state to the other and back while filtering out events in either state. In such a two-state system different waveforms at the SCT output can be created depending on the event history. Changing between states is event-driven and happens without any intervention by the CPU.

Formally, the SCTimer/PWM can be programmed as state machine generator. The ability to perform switching between groups of events provides the SCT the unique capability to be utilized as a highly complex State Machine engine. Events identify the occurrence of conditions that warrant state changes and determine the next state to move to. This provides an extremely powerful control tool - particularly when the SCT inputs and outputs are connected to other on-chip resources (comparators, ADC triggers, other timers etc.) in addition to general-purpose I/O.

In addition to events and states, the SCTimer/PWM provides other enhanced features:

- Four alternative clocking modes including a fully asynchronous mode.
- Selection of any SCT input as a clock source or a clock gate.
- Capability of selecting a “greater-than-or-equal-to” match condition for the purpose of

event generation.

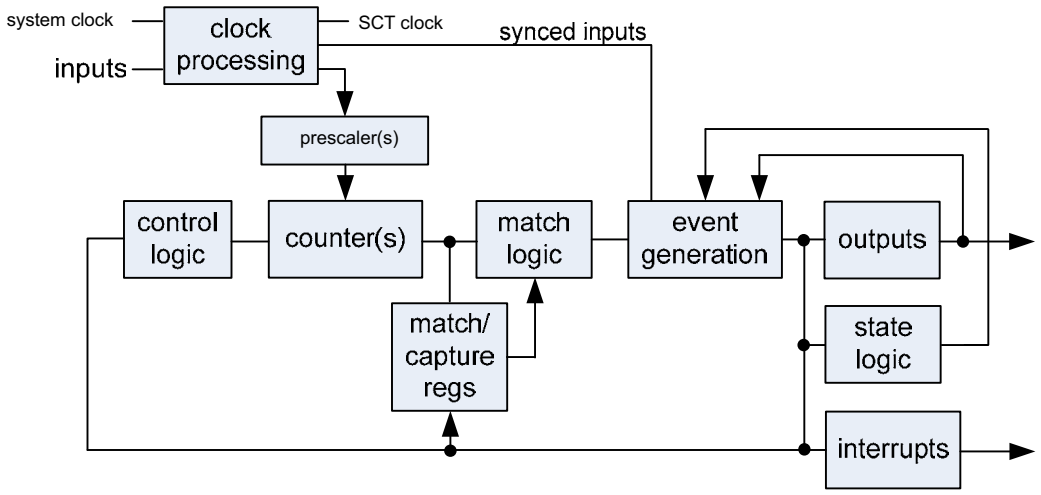


Fig 34. SCTimer/PWM block diagram

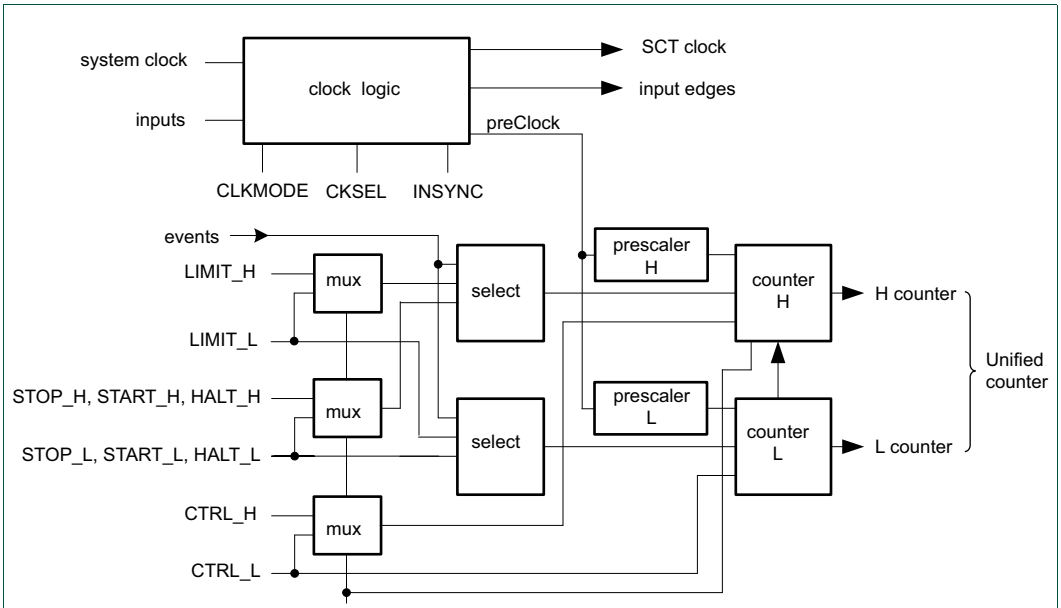


Fig 35. SCTimer/PWM counter and select logic

Remark: In this chapter, the term bus error indicates an SCT response that makes the processor take an exception.

16.6 Register description

The register addresses of the State Configurable Timer are shown in Table 222. For most of the SCT registers, the register function depends on the setting of certain other register bits:

1. The UNIFY bit in the CONFIG register determines whether the SCT is used as one 32-bit register (for operation as one 32-bit counter/timer) or as two 16-bit counter/timers named L and H. The setting of the UNIFY bit is reflected in the register map:
   - UNIFY = 1: Only one register is used (for operation as one 32-bit counter/timer).
   - UNIFY = 0: Access the L and H registers by a 32-bit read or write operation or can be read or written to individually (for operation as two 16-bit counter/timers).

   Typically, the UNIFY bit is configured by writing to the CONFIG register before any other registers are accessed.
2. The REGMODEn bits in the REGMODE register determine whether each set of Match/Capture registers uses the match or capture functionality:
   - REGMODEn = 1: Registers operate as match and reload registers.
   - REGMODEn = 0: Registers operate as capture and capture control registers.

Table 222. Register overview: State Configurable Timer SCT/PWM (base address 0x5000 4000)

Name	Access	Address offset	Description	Reset value	Reference
CONFIG	R/W	0x000	SCT configuration register	0x0000 7E00	Table 223
CTRL	R/W	0x004	SCT control register	0x0004 0004	Table 224
CTRL_L	R/W	0x004	SCT control register low counter 16-bit	0x0004 0004	Table 224
CTRL_H	R/W	0x006	SCT control register high counter 16-bit	0x0004 0004	Table 224
LIMIT	R/W	0x008	SCT limit event select register	0x0000 0000	Table 225
LIMIT_L	R/W	0x008	SCT limit event select register low counter 16-bit	0x0000 0000	Table 225
LIMIT_H	R/W	0x00A	SCT limit event select register high counter 16-bit	0x0000 0000	Table 225
HALT	R/W	0x00C	SCT halt events elect register	0x0000 0000	Table 226
HALT_L	R/W	0x00C	SCT halt event select register low counter 16-bit	0x0000 0000	Table 226
HALT_H	R/W	0x00E	SCT halt event select register high counter 16-bit	0x0000 0000	Table 226
STOP	R/W	0x010	SCT stop event select register	0x0000 0000	Table 227
STOP_L	R/W	0x010	SCT stop event select register low counter 16-bit	0x0000 0000	Table 227
STOP_H	R/W	0x012	SCT stop event select register high counter 16-bit	0x0000 0000	Table 227
START	R/W	0x014	SCT start event select register	0x0000 0000	Table 228
START_L	R/W	0x014	SCT start event select register low counter 16-bit	0x0000 0000	Table 228
START_H	R/W	0x016	SCT start event select register high counter 16-bit	0x0000 0000	Table 228
-	-	0x018 - 0x03C	Reserved		-
COUNT	R/W	0x040	SCT counter register	0x0000 0000	Table 229
COUNT_L	R/W	0x040	SCT counter register low counter 16-bit	0x0000 0000	Table 229
COUNT_H	R/W	0x042	SCT counter register high counter 16-bit	0x0000 0000	Table 229
STATE	R/W	0x044	SCT state register	0x0000 0000	Table 230
STATE_L	R/W	0x044	SCT state register low counter 16-bit	0x0000 0000	Table 230
STATE_H	R/W	0x046	SCT state register high counter 16-bit	0x0000 0000	Table 230
INPUT	RO	0x048	SCT input register	0x0000 0000	Table 231
REGMODE	R/W	0x04C	SCT match/capture mode register	0x0000 0000	Table 232
REGMODE_L	R/W	0x04C	SCT match/capture mode register low counter 16-bit	0x0000 0000	Table 232

Table 222. Register overview: State Configurable Timer SCT/PWM (base address 0x5000 4000) *…continued*

Name	Access	Address offset	Description	Reset value	Reference
REGMODE_H	R/W	0x04E	SCT match/capture registers mode register high counter 16-bit	0x0000 0000	Table 232
OUTPUT	R/W	0x050	SCT output register	0x0000 0000	Table 233
OUTPUTDIRCTRL	R/W	0x054	SCT output counter direction control register	0x0000 0000	Table 234
RES	R/W	0x058	SCT conflict resolution register	0x0000 0000	Table 235
DMAREQ0	R/W	0x05C	SCT DMA request 0 register	0x0000 0000	Table 236
DMAREQ1	R/W	0x060	SCT DMA request 1 register	0x0000 0000	Table 237
-	-	0x064 - 0x0EC	Reserved	-	-
EVEN	R/W	0x0F0	SCT event interrupt enable register	0x0000 0000	Table 238
EVFLAG	R/W	0x0F4	SCT event flag register	0x0000 0000	Table 239
CONEN	R/W	0x0F8	SCT conflict interrupt enable register	0x0000 0000	Table 240
CONFLAG	R/W	0x0FC	SCT conflict flag register	0x0000 0000	Table 241
MATCH0 to MATCH7	R/W	0x100 to 0x11C	SCT match value register of match channels 0 to 7; REGMOD0 to REGMODE7 = 0	0x0000 0000	Table 241
MATCH0_L to MATCH7_L	R/W	0x100 to 0x11C	SCT match value register of match channels 0 to 7; low counter 16-bit; REGMOD0_L to REGMODE7_L = 0	0x0000 0000	Table 241
MATCH0_H to MATCH7_H	R/W	0x102 to 0x11E	SCT match value register of match channels 0 to 7; high counter 16-bit; REGMOD0_H to REGMODE7_H = 0	0x0000 0000	Table 241
CAP0 to CAP7	R/W	0x100 to 0x11C	SCT capture register of capture channel 0 to 7; REGMOD0 to REGMODE7 = 1	0x0000 0000	Table 243
CAP0_L to CAP7_L	R/W	0x100 to 0x11C	SCT capture register of capture channel 0 to 7; low counter 16-bit; REGMOD0_L to REGMODE7_L = 1	0x0000 0000	Table 243
CAP0_H to CAP7_H	R/W	0x102 to 0x11E	SCT capture register of capture channel 0 to 7; high counter 16-bit; REGMOD0_H to REGMODE7_H = 1	0x0000 0000	Table 243
MATCHREL0 to MATCHREL7	R/W	0x200 to 0x21C	SCT match reload value register 0 to 7; REGMOD0 = 0 to REGMODE7 = 0	0x0000 0000	Table 244
MATCHREL0_L to MATCHREL7_L	R/W	0x200 to 0x21C	SCT match reload value register 0 to 7; low counter 16-bit; REGMOD0_L = 0 to REGMODE7_L = 0	0x0000 0000	Table 244
MATCHREL0_H to MATCHREL7_H	R/W	0x202 to 0x21E	SCT match reload value register 0 to 7; high counter 16-bit; REGMOD0_H = 0 to REGMODE7_H = 0	0x0000 0000	Table 244
CAPCTRL0 to CAPCTRL7	R/W	0x200 to 0x21C	SCT capture control register 0 to 7; REGMOD0 = 1 to REGMODE7 = 1	0x0000 0000	Table 245
CAPCTRL0_L to CAPCTRL7_L	R/W	0x200 to 0x21C	SCT capture control register 0 to 7; low counter 16-bit; REGMOD0_L = 1 to REGMODE7_L = 1	0x0000 0000	Table 245
CAPCTRL0_H to CAPCTRL7_H	R/W	0x202 to 0x21E	SCT capture control register 0 to 7; high counter 16-bit; REGMOD0 = 1 to REGMODE7 = 1	0x0000 0000	Table 245
EV0_STATE	R/W	0x300	SCT event state register 0	0x0000 0000	Table 246
EV0_CTRL	R/W	0x304	SCT event control register 0	0x0000 0000	Table 247

Table 222. Register overview: State Configurable Timer SCT/PWM (base address 0x5000 4000) *…continued*

Name	Access	Address offset	Description	Reset value	Reference
EV1_STATE	R/W	0x308	SCT event state register 1	0x0000 0000	Table 246
EV1_CTRL	R/W	0x30C	SCT event control register 1	0x0000 0000	Table 247
EV2_STATE	R/W	0x310	SCT event state register 2	0x0000 0000	Table 246
EV2_CTRL	R/W	0x314	SCT event control register 2	0x0000 0000	Table 247
EV3_STATE	R/W	0x318	SCT event state register 3	0x0000 0000	Table 246
EV3_CTRL	R/W	0x31C	SCT event control register 3	0x0000 0000	Table 247
EV4_STATE	R/W	0x320	SCT event state register 4	0x0000 0000	Table 246
EV4_CTRL	R/W	0x324	SCT event control register4	0x0000 0000	Table 247
EV5_STATE	R/W	0x328	SCT event state register 5	0x0000 0000	Table 246
EV5_CTRL	R/W	0x32C	SCT event control register 5	0x0000 0000	Table 247
EV6_STATE	R/W	0x330	SCT event state register 6	0x0000 0000	Table 246
EV6_CTRL	R/W	0x334	SCT event control register 6	0x0000 0000	Table 247
EV7_STATE	R/W	0x338	SCT event state register 7	0x0000 0000	Table 246
EV7_CTRL	R/W	0x33C	SCT event control register 7	0x0000 0000	Table 247
OUT0_SET	R/W	0x500	SCT output 0 set register	0x0000 0000	Table 248
OUT0_CLR	R/W	0x504	SCT output 0 clear register	0x0000 0000	Table 249
OUT1_SET	R/W	0x508	SCT output 1 set register	0x0000 0000	Table 248
OUT1_CLR	R/W	0x50C	SCT output 1 clear register	0x0000 0000	Table 249
OUT2_SET	R/W	0x510	SCT output 2 set register	0x0000 0000	Table 248
OUT2_CLR	R/W	0x514	SCT output 2 clear register	0x0000 0000	Table 249
OUT3_SET	R/W	0x518	SCT output 3 set register	0x0000 0000	Table 248
OUT3_CLR	R/W	0x51C	SCT output 3 clear register	0x0000 0000	Table 249
OUT4_SET	R/W	0x520	SCT output 4 set register	0x0000 0000	Table 248
OUT4_CLR	R/W	0x524	SCT output 4 clear register	0x0000 0000	Table 249
OUT5_SET	R/W	0x528	SCT output 5 set register	0x0000 0000	Table 248
OUT5_CLR	R/W	0x52C	SCT output 5 clear register	0x0000 0000	Table 249

16.6.1 Register functional grouping

Most SCT registers either configure an event or select an event for a specific action of the counter (or counters) and outputs. Figure 36 shows the registers and register bits that can be configured for each event.

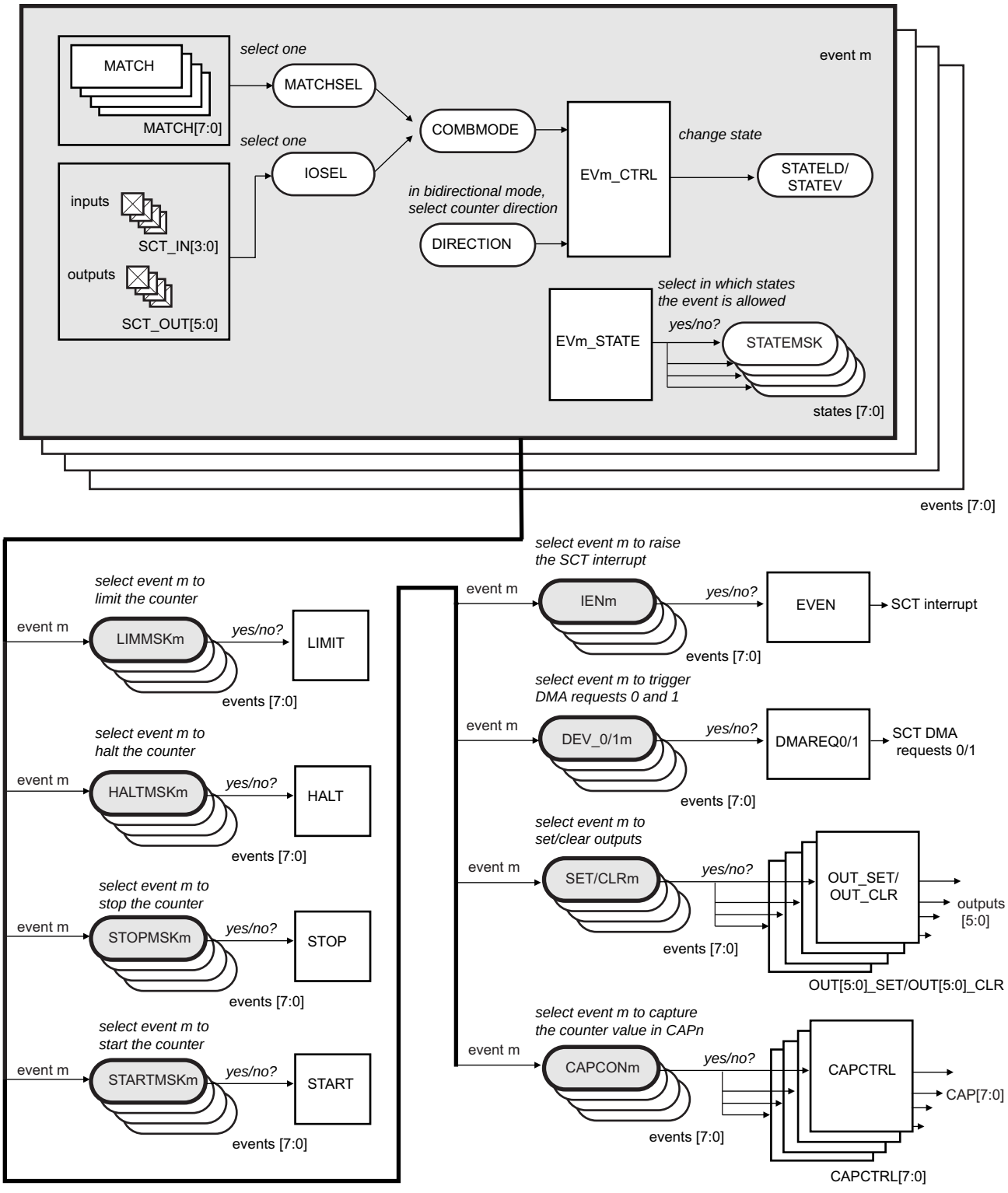


Fig 36. SCT event configuration and selection registers

16.6.1.1 Counter configuration and control registers

The SCT contains two registers for configuring the SCT and monitor and control its operation by software.

- The configuration register (CONFIG) configures the SCT in single, 32-bit counter mode or in dual, 16-bit counter mode, configures the clocking and clock synchronization, and configures automatic limits and the use of reload registers.
- The control register (CTRL) allows to monitor and set the counter direction, and to clear, start, stop, or halt the 32-bit counter or each individual 16-bit counter if in dual-counter mode.

16.6.1.2 Event configuration registers

Each event is associated with two registers:

- One EVn_CTRL register per event to define what triggers the event.
- One EVn_STATE register per event to enable the event.

16.6.1.3 Match and capture registers

The SCT includes a set of registers to store the SCT's match or capture values. Each match register is associated with a match reload register which automatically reloads the match register at the beginning of each counter cycle. This register group includes the following registers:

- One REGMODE register per match/capture register to configure each match/capture register for either storing a match value or a capture value.
- A set of match/capture registers with each register, depending on the setting of REGMODE, either storing a match value or a counter value.
- One reload register for each match register.

16.6.1.4 Event select registers for the counter operations

This group contains the registers that select the events which affect the counter. Counter actions are limit, halt, and start or stop and apply to the unified counter or to the two 16-bit counters. Also included is the counter register with the counter value, or values in the dual-counter set-up. This register group includes the following registers:

- LIMIT selects the events that limit the counter.
- START and STOP select events that start or stop the counter.
- HALT selects events that halt the counter: HALT
- COUNT contains the counter value.

The LIMIT, START, STOP, and HALT registers each contain one bit per event that selects for each event whether the event limits, stops, starts, or halts the counter, or counters in dual-counter mode.

In the dual-counter mode, the events can be selected independently for each counter.

16.6.1.5 Event select registers for setting or clearing the outputs

This group contains the registers that select the events which affect the level of each SCT output. Also included are registers to manage conflicts that occur when events try to set or clear the same output. This register group includes the following registers:

- One OUTn_SET register for each output to select the events which set the output.
- One OUTn_CLR register for each output to select the events which clear the output.
- The conflict resolution register which defines an action when more than one event try to control an output at the same time.
- The conflict flag and conflict interrupt enable registers that monitor interrupts arising from output set and clear conflicts.
- The output direction control register that interchanges the set and clear output operation caused by an event in bi-directional mode.

The OUTn_SET and OUTn_CLR registers each contain one bit per event that selects whether the event changes the state a given output n.

In the dual-counter mode, the events can be selected independently for each output.

16.6.1.6 Event select registers for capturing a counter value

This group contains registers that select events which capture the counter value and store it in one of the CAP registers. Each capture register m has one associated CAPCTRLm register which in turn selects the events to capture the counter value.

16.6.1.7 Event select register for initiating DMA transfers

One register is provided for each of the two DMA requests to select the events that can trigger a DMA request.

The DMAREQn register contain one bit for each event that selects whether this event triggers a DMA request. An additional bit enables the DMA trigger when the match registers are reloaded.

16.6.1.8 Interrupt handling registers

The following registers provide flags that are set by events and select the events that when they occur request an interrupt.

- The event flag register provides one flag for each event that is set when the event occurs.
- The event flag interrupt enable register provides one bit for each event to be enabled for the SCT interrupt.

16.6.1.9 Registers for controlling SCT inputs and outputs by software

Two registers are provided that allow software (as opposed to events) to set input and outputs of the SCT:

- The SCT input register to read the state of any of the SCT inputs.
- The SCT output register to set or clear any of the SCT outputs or to read the state of the outputs.

16.6.2 SCT configuration register

This register configures the overall operation of the SCT. Write to this register before any other registers. Only word-writes are permitted to this register. Attempting to write a half-word value results in a bus error.

Table 223. SCT configuration register (CONFIG, address 0x5000 4000) bit description

Bit	Symbol	Value	Description	Reset value
0	UNIFY		SCT operation	0
		0	The SCT operates as two 16-bit counters named L and H.	
		1	The SCT operates as a unified 32-bit counter.	
2:1	CLKMODE		SCT clock mode	00
		0x0	System Clock Mode. The system clock clocks the entire SCT module including the counter(s) and counter prescalers.	
		0x1	Sampled System Clock Mode. The system clock clocks the SCT module, but the counter and prescalers are only enabled to count when the designated edge is detected on the input selected by the CKSEL field. The minimum pulse width on the selected clock-gate input is one system clock period. This mode is the high-performance, sampled-clock mode.	
		0x2	SCT Input Clock Mode. The input/edge selected by the CKSEL field clocks the SCT module, including the counters and prescalers, after first being synchronized to the system clock. The minimum pulse width on the clock input is 1 bus clock one system clock period. This mode is the low-power, sampled-clock mode.	
		0x3	Asynchronous Mode. The entire SCT module is clocked directly by the input/edge selected by the CKSEL field. In this mode, the SCT outputs are switched synchronously to the SCT input clock - not the system clock. The input clock rate must be at least half the system clock rate and can be the same or faster than the system clock.	
6:3	CKSEL		SCT clock select. The specific functionality of the designated input/edge is dependent on the CLKMODE bit selection in this register.	0000
		0x0	Rising edges on input 0.	
		0x1	Falling edges on input 0.	
		0x2	Rising edges on input 1.	
		0x3	Falling edges on input 1.	
		0x4	Rising edges on input 2.	
		0x5	Falling edges on input 2.	
		0x6	Rising edges on input 3.	
		0x7	Falling edges on input 3.	
7	NORELAOD_L	-	A 1 in this bit prevents the lower match registers from being reloaded from their respective reload registers. Setting this bit eliminates the need to write to the reload registers MATCHREL if the match values are fixed. Software can write to set or clear this bit at any time. This bit applies to both the higher and lower registers when the UNIFY bit is set.	0
8	NORELOAD_H	-	A 1 in this bit prevents the higher match registers from being reloaded from their respective reload registers. Setting this bit eliminates the need to write to the reload registers MATCHREL if the match values are fixed. Software can write to set or clear this bit at any time. This bit is not used when the UNIFY bit is set.	0

Table 223. SCT configuration register (CONFIG, address 0x5000 4000) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
12:9	INSYNC	-	Synchronization for input N (bit 9 = input 0, bit 10 = input 1,..., bit 12 = input 3); all other bits are reserved. A 1 in one of these bits subjects the corresponding input to synchronization to the SCT clock, before it is used to create an event. If an input is known to already be synchronous to the SCT clock, this bit may be set to 0 for faster input response. (Note: The SCT clock is the system clock for CLKMODEs 0-2. It is the selected, asynchronous SCT input clock for CLKMODE3). Note that the INSYNC field only affects inputs used for event generation. It does not apply to the clock input specified in the CKSEL field.	1
16:13	-	-	Reserved.	
17	AUTOLIMIT_L	-	A one in this bit causes a match on match register 0 to be treated as a de-facto LIMIT condition without the need to define an associated event. As with any LIMIT event, this automatic limit causes the counter to be cleared to zero in uni-directional mode or to change the direction of count in bi-directional mode. Software can write to set or clear this bit at any time. This bit applies to both the higher and lower registers when the UNIFY bit is set.	
18	AUTOLIMIT_H	-	A one in this bit will cause a match on match register 0 to be treated as a de-facto LIMIT condition without the need to define an associated event. As with any LIMIT event, this automatic limit causes the counter to be cleared to zero in uni-directional mode or to change the direction of count in bi-directional mode. Software can write to set or clear this bit at any time. This bit is not used when the UNIFY bit is set.	
31:19	-		Reserved	-

16.6.3 SCT control register

If bit UNIFY = 1 in the CONFIG register, only the _L bits are used.

If bit UNIFY = 0 in the CONFIG register, this register can be written to as two registers CTRL_L and CTRL_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

All bits in this register can be written to when the counter is stopped or halted. When the counter is running, the only bits that can be written are STOP or HALT. (Other bits can be written in a subsequent write after HALT is set to 1.)

Remark: If CLKMODE = 0x3 is selected, wait at least 12 system clock cycles between a write access to the H, L or unified version of this register and the next write access. This restriction does not apply when writing to the HALT bit or bits and then writing to the CTRL register again to restart the counters - for example because software must update the MATCH register, which is only allowed when the counters are halted.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Table 224. SCT control register (CTRL, address 0x5000 4004) bit description

Bit	Symbol	Value	Description	Reset value
0	DOWN_L	-	This bit is 1 when the L or unified counter is counting down. Hardware sets this bit when the counter is counting up, counter limit occurs, and BIDIR = 1.Hardware clears this bit when the counter is counting down and a limit condition occurs or when the counter reaches 0.	0
1	STOP_L	-	When this bit is 1 and HALT is 0, the L or unified counter does not run, but I/O events related to the counter can occur. If a designated start event occurs, this bit is cleared and counting resumes.	0
2	HALT_L	-	When this bit is 1, the L or unified counter does not run and no events can occur. A reset sets this bit. When the HALT_L bit is one, the STOP_L bit is cleared. It is possible to remove the halt condition while keeping the SCT in the stop condition (not running) with a single write to this register to simultaneously clear the HALT bit and set the STOP bit. **Remark:** Once set, only software can clear this bit to restore counter operation. This bit is set on reset.	1
3	CLRCTR_L	-	Writing a 1 to this bit clears the L or unified counter. This bit always reads as 0.	0
4	BIDIR_L		L or unified counter direction select	0
		0	Up. The counter counts up to a limit condition, then is cleared to zero.	
		1	Up-down. The counter counts up to a limit, then counts down to a limit condition or to 0.	
12:5	PRE_L	-	Specifies the factor by which the SCT clock is prescaled to produce the L or unified counter clock. The counter clock is clocked at the rate of the SCT clock divided by PRE_L+1. **Remark:** Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.	0
15:13	-		Reserved	
16	DOWN_H	-	This bit is 1 when the H counter is counting down. Hardware sets this bit when the counter is counting, a counter limit condition occurs, and BIDIR is 1. Hardware clears this bit when the counter is counting down and a limit condition occurs or when the counter reaches 0.	0
17	STOP_H	-	When this bit is 1 and HALT is 0, the H counter does not, run but I/O events related to the counter can occur. If such an event matches the mask in the Start register, this bit is cleared and counting resumes.	0
18	HALT_H	-	When this bit is 1, the H counter does not run and no events can occur. A reset sets this bit. When the HALT_H bit is one, the STOP_H bit is cleared. It is possible to remove the halt condition while keeping the SCT in the stop condition (not running) with a single write to this register to simultaneously clear the HALT bit and set the STOP bit. **Remark:** Once set, this bit can only be cleared by software to restore counter operation. This bit is set on reset.	1
19	CLRCTR_H	-	Writing a 1 to this bit clears the H counter. This bit always reads as 0.	0

Table 224. SCT control register (CTRL, address 0x5000 4004) bit description

Bit	Symbol	Value	Description	Reset value
20	BIDIR_H		Direction select	0
		0	The H counter counts up to its limit condition, then is cleared to zero.	
		1	The H counter counts up to its limit, then counts down to a limit condition or to 0.	
28:21	PRE_H	-	Specifies the factor by which the SCT clock is prescaled to produce the H counter clock. The counter clock is clocked at the rate of the SCT clock divided by PRELH+1. **Remark:** Clear the counter (by writing a 1 to the CLRCTR bit) whenever changing the PRE value.	0
31:29	-		Reserved	

16.6.4 SCT limit event select register

The running counter can be limited by an event. When any of the events selected in this register occur, the counter is cleared to zero from its current value or changes counting direction if in bi-directional mode.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit causes its associated event to serve as a LIMIT event. When any limit event occurs, the counter is reset to zero in uni-directional mode or changes its direction of count in bi-directional mode.To define the actual limiting event (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: Counting up to all ones or counting down to zero is always equivalent to a limit event occurring.

Note that in addition to using this register to specify events that serve as limits, it is also possible to automatically cause a limit condition whenever a match register 0 match occurs. This eliminates the need to define an event for the sole purpose of creating a limit. The AUTOLIMITL and AUTOLIMITH bits in the configuration register enable/disable this feature (see Table 223).

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers LIMIT_L and LIMIT_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 225. SCT limit event select register (LIMIT, address 0x5000 4008) bit description

Bit	Symbol	Description	Reset value
7:0	LIMMSK_L	If bit n is one, event n is used as a counter limit for the L or unified counter (event 0 = bit 0, event 1 = bit 1, event 7 = bit 7).	0
15:8	-	Reserved.	-
23:16	LIMMSK_H	If bit n is one, event n is used as a counter limit for the H counter (event 0 = bit 16, event 1 = bit 17, event 7 = bit 23).	0
31:24	-	Reserved.	-

16.6.5 SCT halt event select register

The running counter can be disabled (halted) by an event. When any of the events selected in this register occur, the counter stops running and all further events are disabled.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a HALT event. To define the actual events that cause the counter to halt (a match, an I/O pin toggle, etc.), see the EVn_CTRL registers.

Remark: A HALT condition can only be removed when software clears the HALT bit in the CTRL register (Table 224).

If UNIFY = 1 in the CONFIG register, only the L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers HALT_L and HALT_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 226. SCT halt event select register (HALT, address 0x5000 400C) bit description

Bit	Symbol	Description	Reset value
7:0	HALTMSK_L	If bit n is one, event n sets the HALT_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, event 7 = bit 7).	0
15:8	-	Reserved.	-
23:16	HALTMSK_H	If bit n is one, event n sets the HALT_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, event 7 = bit 23).	0
31:24	-	Reserved.	-

16.6.6 SCT stop event select register

The running counter can be stopped by an event. When any of the events selected in this register occur, counting is suspended, that is the counter stops running and remains at its current value. Event generation remains enabled, and any event selected in the START register such as an I/O event or an event generated by the other counter can restart the counter.

This register specifies which events stop the counter. Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a STOP event. To define the actual event that causes the counter to stop (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: Software can stop and restart the counter by writing to the CTRL register.

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers STOPT_L and STOP_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 227. SCT stop event select register (STOP, address 0x5000 4010) bit description

Bit	Symbol	Description	Reset value
7:0	STOPMSK_L	If bit n is one, event n sets the STOP_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, event 7 = bit 7).	0
15:8	-	Reserved.	-
23:16	STOPMSK_H	If bit n is one, event n sets the STOP_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, event 7= bit 23).	0
31:24	-	Reserved.	-

16.6.7 SCT start event select register

The stopped counter can be re-started by an event. When any of the events selected in this register occur, counting is restarted from the current counter value.

Each bit of the register is associated with a different event (bit 0 with event 0, etc.). Setting a bit will cause its associated event to serve as a START event. When any START event occurs, hardware will clear the STOP bit in the Control Register CTRL. Note that a START event has no effect on the HALT bit. Only software can remove a HALT condition. To define the actual event that starts the counter (an I/O pin toggle or an event generated by the other running counter in dual-counter mode), see the EVn_CTRL register.

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers START_L and START_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 228. SCT start event select register (START, address 0x5000 4014) bit description

Bit	Symbol	Description	Reset value
7:0	STARTMSK_L	If bit n is one, event n clears the STOP_L bit in the CTRL register (event 0 = bit 0, event 1 = bit 1, event 7 = bit 7).	0
15:8	-	Reserved.	-
23:16	STARTMSK_H	If bit n is one, event n clears the STOP_H bit in the CTRL register (event 0 = bit 16, event 1 = bit 17, event 7 = bit 23).	0
31:24	-	Reserved.	-

16.6.8 SCT counter register

If UNIFY = 1 in the CONFIG register, the counter is a unified 32-bit register and both the _L and _H bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers COUNT_L and COUNT_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation. In this case, the L and H registers count independently under the control of the other registers.

Writing to the COUNT_L, COUNT_H, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register). Attempting to write to the counter when it is not halted causes a bus error. Software can read the counter registers at any time.

Table 229. SCT counter register (COUNT, address 0x5000 4040) bit description

Bit	Symbol	Description	Reset value
15:0	CTR_L	When UNIFY = 0, read or write the 16-bit L counter value. When UNIFY = 1, read or write the lower 16 bits of the 32-bit unified counter.	0
31:16	CTR_H	When UNIFY = 0, read or write the 16-bit H counter value. When UNIFY = 1, read or write the upper 16 bits of the 32-bit unified counter.	0

16.6.9 SCT state register

Each group of enabled and disabled events is assigned a number called the state variable. For example, a state variable with a value of 0 could have events 0, 2, and 3 enabled and all other events disabled. A state variable with the value of 1 could have events 1, 4, and 5 enabled and all others disabled.

Remark: The EVm_STATE registers define which event is enabled in each group.

Software can read the state associated with a counter at any time. Writing to the STATE_L, STATE_H, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register).

The state variable is the main feature that distinguishes the SCTimer/PWM from other counter/timer/ PWM blocks. Events can be made to occur only in certain states. Events, in turn, can perform the following actions:

- set and clear outputs
- limit, stop, and start the counter
- cause interrupts and DMA requests
- modify the state variable

The value of a state variable is completely under the control of the application. If an application does not use states, the value of the state variable remains zero, which is the default value.

A state variable can be used to track and control multiple cycles of the associated counter in any desired operational sequence. The state variable is logically associated with a state machine diagram which represents the SCT configuration. See Section 16.6.24 and 16.6.25 for more about the relationship between states and events.

The STATELD/STADEV fields in the event control registers of all defined events set all possible values for the state variable. The change of the state variable during multiple counter cycles reflects how the associated state machine moves from one state to the next.

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers STATE_L and STATE_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Table 230. SCT state register (STATE, address 0x5000 4044) bit description

Bit	Symbol	Description	Reset value
4:0	STATE_L	State variable.	0
15:5	-	Reserved.	-
20:16	STATE_H	State variable.	0
31:21	-	Reserved.	

16.6.10 SCT input register

Software can read the state of the SCT inputs in this read-only register in slightly different forms.

1. The AIN bit displays the state of the input captured on each rising edge of the SCT clock This corresponds to a nearly direct read-out of the input but can cause spurious fluctuations in case of an asynchronous input signal.
2. The SIN bit displays the form of the input as it is used for event detection. This may include additional stages of synchronization, depending on what is specified for that input in the INSYNC field in the CONFIG register:
   - If the INSYNC bit is set for the input, the input is triple-synchronized to the SCT clock resulting in a stable signal that is delayed by three SCT clock cycles.
   - If the INSYNC bit is not set, the SIN bit value is identical to the AIN bit value.

Table 231. SCT input register (INPUT, address 0x5000 4048) bit description

Bit	Symbol	Description	Reset value
0	AIN0	Input 0 state.Input 0 state on the last SCT clock edge.	-
1	AIN1	Input 1 state. Input 1 state on the last SCT clock edge.	-
2	AIN2	Input 2 state. Input 2 state on the last SCT clock edge.	-
3	AIN3	Input 3 state. Input 3 state on the last SCT clock edge.	-
15:4	-	Reserved.	-
16	SIN0	Input 0 state. Input 0 state following the synchronization specified by INSYNC0.	-
17	SIN1	Input 1 state. Input 1 state following the synchronization specified by INSYNC0.	-
18	SIN2	Input 2 state. Input 2 state following the synchronization specified by INSYNC0.	-
19	SIN3	Input 3 state. Input 3 state following the synchronization specified by INSYNC0.	-
31:20	-	Reserved	-

16.6.11 SCT match/capture mode register

If UNIFY = 1 in the CONFIG register, only the _L bits of this register are used. In this case, REGMODE_H is not used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers REGMODE_L and REGMODE_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation. The _L bits/registers control the L match/capture registers, and the _H bits/registers control the H match/capture registers.

The SCT contains multiple Match/Capture registers. The Register Mode register selects whether each register acts as a Match register (see Section 16.6.20) or as a Capture register (see Section 16.6.21). Each Match/Capture register has an accompanying register which functions as a Reload register when the primary register is used as a Match register (Section 16.6.22) or as a Capture-Control register when the register is used as a capture register (Section 16.6.23). REGMODE_H is used only when the UNIFY bit is 0.

Table 232. SCT match/capture mode register (REGMODE, address 0x5000 404C) bit description

Bit	Symbol	Description	Reset value
7:0	REGMOD_L	Each bit controls one match/capture register (register 0 = bit 0, register 1 = bit 1,..., register = bit 7). 0 = register operates as match register. 1 = register operates as capture register.	0
15:8	-	Reserved.	-
23:16	REGMOD_H	Each bit controls one match/capture register (register 0 = bit 16, register 1 = bit 17,..., register 7 = bit 23). 0 = register operates as match registers. 1 = register operates as capture registers.	0
31:24	-	Reserved.	-

16.6.12 SCT output register

Each SCT output has a corresponding bit in this register to allow software to control the output state directly or read its current state.

While the counter is running, outputs are set, cleared, or toggled only by events. However, using this register, software can write to any of the output registers when both counters are halted to control the outputs directly. Writing to the OUT register is only allowed when all counters (L-counter, H-counter, or unified counter) are halted (HALT bits are set to 1 in the CTRL register).

Software can read this register at any time to sense the state of the outputs.

Table 233. SCT output register (OUTPUT, address 0x5000 4050) bit description

Bit	Symbol	Description	Reset value
5:0	OUT	Writing a 1 to bit n forces the corresponding output HIGH. Writing a 0 forces the corresponding output LOW (output 0 = bit 0, output 1 = bit 1,..., output 5 = bit 5).	0
31:6	-	Reserved	

16.6.13 SCT bi-directional output control register

For bi-directional mode, this register specifies (for each output) the impact of the counting direction on the meaning of set and clear operations on the output (see Section 16.6.26 and Section 16.6.27). The purpose of this register is to facilitate the creation of center-aligned output waveforms without the need to define additional events.

Table 234. SCT bidirectional output control register (OUTPUTDIRCTRL, address 0x5000 4054) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SETCLR0		Set/clear operation on output 0. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
3:2	SETCLR1		Set/clear operation on output 1. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
5:4	SETCLR2		Set/clear operation on output 2. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
7:6	SETCLR3		Set/clear operation on output 3. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
9:8	SETCLR4		Set/clear operation on output 4. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
11:10	SETCLR5		Set/clear operation on output 5. Value 0x3 is reserved. Do not program this value.	0
		0x0	Set and clear do not depend on the direction of any counter.	
		0x1	Set and clear are reversed when counter L or the unified counter is counting down.	
		0x2	Set and clear are reversed when counter H is counting down. Do not use if UNIFY = 1.	
31:12	-		Reserved	-

16.6.14 SCT conflict resolution register

The output conflict resolution register specifies what action should be taken if multiple events (or even the same event) dictate that a given output should be both set and cleared at the same time.

To enable an event to toggle an output each time the event occurs, set the bits for that event in both the OUTn_SET and OUTn_CLR registers and set the On_RES value to 0x3 in this register.

.

Table 235. SCT conflict resolution register (RES, address 0x5000 4058) bit description

Bit	Symbol	Value	Description	Reset value
1:0	O0RES		Effect of simultaneous set and clear on output 0.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR0 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR0 field).	
		0x3	Toggle output.	
3:2	O1RES		Effect of simultaneous set and clear on output 1.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR1 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR1 field).	
		0x3	Toggle output.	
5:4	O2RES		Effect of simultaneous set and clear on output 2.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR2 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output n (or set based on the SETCLR2 field).	
		0x3	Toggle output.	
7:6	O3RES		Effect of simultaneous set and clear on output 3.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR3 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR3 field).	
		0x3	Toggle output.	
9:8	O4RES		Effect of simultaneous set and clear on output 4.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR4 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR4 field).	
		0x3	Toggle output.	
11:10	O5RES		Effect of simultaneous set and clear on output 5.	0
		0x0	No change.	
		0x1	Set output (or clear based on the SETCLR5 field in the OUTPUTDIRCTRL register).	
		0x2	Clear output (or set based on the SETCLR5 field).	
		0x3	Toggle output.	
31:12	-	-	Reserved	-

16.6.15 SCT DMA request 0 and 1 registers

The SCT includes two DMA request outputs. These registers enable the DMA requests to be triggered when a particular event occurs or when counter Match registers are loaded from its Reload registers. The DMA request registers are word-write only. Attempting to write a half-word value to these registers result in a bus error.

Event-triggered DMA requests are particularly useful for launching DMA activity to or from other peripherals under the control of the SCT.

Table 236. SCT DMA 0 request register (DMAREQ0, address 0x5000 405C) bit description

Bit	Symbol	Description	Reset value
5:0	DEV_0	If bit n is one, event n triggers DMA request 0 (event 0 = bit 0, event 1 = bit 1,..., event 7 = bit 7).	0
29:6	-	Reserved	-
30	DRL0	A 1 in this bit triggers DMA request 0 when it loads the Match_L/Unified registers from the Reload_L/Unified registers.	
31	DRQ0	This read-only bit indicates the state of DMA Request 0	

Table 237. SCT DMA 1 request register (DMAREQ1, address 0x5000 C060) bit description

Bit	Symbol	Description	Reset value
5:0	DEV_1	If bit n is one, event n triggers DMA request 1 (event 0 = bit 0, event 1 = bit 1,..., event 7 = bit 7).	0
29:6	-	Reserved	-
30	DRL1	A 1 in this bit triggers DMA request 1 when it loads the Match L/Unified registers from the Reload L/Unified registers.	
31	DRQ1	This read-only bit indicates the state of DMA Request 1.	

16.6.16 SCT event interrupt enable register

This register enables flags to request an interrupt if the FLAGn bit in the SCT event flag register (Section 16.6.17) is also set.

Table 238. SCT event interrupt enable register (EVEN, address 0x5000 40F0) bit description

Bit	Symbol	Description	Reset value
7:0	IEN	The SCT requests an interrupt when bit n of this register and the event flag register are both one (event 0 = bit 0, event 1 = bit 1,..., event 7 = bit 7).	0
31:8	-	Reserved	

16.6.17 SCT event flag register

This register records events. Writing ones to this register clears the corresponding flags and negates the SCT interrupt request if all enabled flag register bits are zero.

Table 239. SCT event flag register (EVFLAG, address 0x5000 40F4) bit description

Bit	Symbol	Description	Reset value
7:0	FLAG	Bit n is one if event n has occurred since reset or a 1 was last written to this bit (event 0 = bit 0, event 1 = bit 1,..., event 7 = bit 7).	0
31:8	-	Reserved	-

16.6.18 SCT conflict interrupt enable register

This register enables the no-change conflict events specified in the SCT conflict resolution register to generate an interrupt request.

Table 240. SCT conflict interrupt enable register (CONEN, address 0x5000 40F8) bit description

Bit	Symbol	Description	Reset value
5:0	NCEN	The SCT requests an interrupt when bit n of this register and the SCT conflict flag register are both one (output 0 = bit 0, output 1 = bit 1,..., output 5 = bit 5).	0
31:6	-	Reserved	

16.6.19 SCT conflict flag register

This register records a no-change conflict occurrence and provides details of a bus error. Writing ones to the NCFLAG bits clears the corresponding read bits and negates the SCT interrupt request if all enabled Flag bits are zero.

Table 241. SCT conflict flag register (CONFLAG, address 0x5000 40FC) bit description

Bit	Symbol	Description	Reset value
5:0	NCFLAG	Bit n is one if a no-change conflict event occurred on output n since reset or a 1 was last written to this bit (output 0 = bit 0, output 1 = bit 1,..., output 5 = bit 5).	0
29:6	-	Reserved.	-
30	BUSERRL	The most recent bus error from this SCT involved writing CTR L/Unified, STATE L/Unified, MATCH L/Unified, or the Output register when the L/U counter was not halted. A word write to certain L and H registers can be half successful and half unsuccessful.	0
31	BUSERRH	The most recent bus error from this SCT involved writing CTR H, STATE H, MATCH H, or the Output register when the H counter was not halted.	0

16.6.20 SCT match registers 0 to 7 (REGMODEn bit = 0)

Match registers are compared to the counters to help create events. When the UNIFY bit is 0, the L and H registers are independently compared to the L and H counters. When UNIFY is 1, the combined L and H registers hold a 32-bit value that is compared to the unified counter. A Match can only occur in a clock in which the counter is running (STOP and HALT are both 0).

Match registers can be read at any time. Writing to the MATCH_L, MATCH_H, or unified register is only allowed when the corresponding counter is halted (HALT bits are set to 1 in the CTRL register). Match events occur in the SCT clock in which the counter is (or would be) incremented to the next value. When a Match event limits its counter as described in Section 16.6.4, the value in the Match register is the last value of the counter before it is cleared to zero (or decremented if BIDIR is 1).

There is no “write-through” from Reload registers to Match registers. Before starting a counter, software can write one value to the Match register used in the first cycle of the counter and a different value to the corresponding Match Reload register used in the second cycle.

Table 242. SCT match registers 0 to 7 (MATCH[0:7], address 0x5000 4100 (MATCH0) to 0x5000 411C (MATCH7)) bit description (REGMODEn bit = 0)

Bit	Symbol	Description	Reset value
15:0	MATCHn_L	When UNIFY = 0, read or write the 16-bit value to be compared to the L counter. When UNIFY = 1, read or write the lower 16 bits of the 32-bit value to be compared to the unified counter.	0
31:16	MATCHn_H	When UNIFY = 0, read or write the 16-bit value to be compared to the H counter. When UNIFY = 1, read or write the upper 16 bits of the 32-bit value to be compared to the unified counter.	0

16.6.21 SCT capture registers 0 to 7 (REGMODEn bit = 1)

These registers allow software to record the counter values upon occurrence of the events selected by the corresponding Capture Control registers occurred.

Table 243. SCT capture registers 0 to 7 (CAP[0:7], address 0x5000 4100 (CAP0) to 0x5000 411C (CAP7)) bit description (REGMODEn bit = 1)

Bit	Symbol	Description	Reset value
15:0	CAPn_L	When UNIFY = 0, read the 16-bit counter value at which this register was last captured. When UNIFY = 1, read the lower 16 bits of the 32-bit value at which this register was last captured.	0
31:16	CAPn_H	When UNIFY = 0, read the 16-bit counter value at which this register was last captured. When UNIFY = 1, read the upper 16 bits of the 32-bit value at which this register was last captured.	0

16.6.22 SCT match reload registers 0 to 7 (REGMODEn bit = 0)

A Match register (L, H, or unified 32-bit) is loaded from its corresponding Reload register at the start of each new counter cycle, that is

- when BIDIR = 0 and the counter is cleared to zero upon reaching it limit condition.

- when BIDIR = 1 and the counter counts down to 0, unless the appropriate NORELOAD bit is set in the CFG register.

Table 244. SCT match reload registers 0 to 7 (MATCHREL[0:7], address 0x5000 4200 (MATCHREL0) to 0x5000 421C (MATCHREL7)) bit description (REGMODEn bit = 0)

Bit	Symbol	Description	Reset value
15:0	RELOADn_L	When UNIFY = 0, specifies the 16-bit value to be loaded into the SCTMATCHn_L register. When UNIFY = 1, specifies the lower 16 bits of the 32-bit value to be loaded into the MATCHn register.	0
31:16	RELOADn_H	When UNIFY = 0, specifies the 16-bit to be loaded into the MATCHn_H register. When UNIFY = 1, specifies the upper 16 bits of the 32-bit value to be loaded into the MATCHn register.	0

16.6.23 SCT capture control registers 0 to 7 (REGMODEn bit = 1)

If UNIFY = 1 in the CONFIG register, only the _L bits are used.

If UNIFY = 0 in the CONFIG register, this register can be written to as two registers CAPCTRLn_L and CAPCTRLn_H. Both the L and H registers can be read or written individually or in a single 32-bit read or write operation.

Based on a selected event, the capture registers can be loaded with the current counter value when the event occurs.

Each Capture Control register (L, H, or unified 32-bit) controls which events cause loading of the corresponding Capture register from the counter.

Table 245. SCT capture control registers 0 to 7 (CAPCTRL[0:7], address 0x5000 4200 (CAPCTRL0) to 0x5000 421C (CAPCTRL7)) bit description (REGMODEn bit = 1)

Bit	Symbol	Description	Reset value
7:0	CAPCONn_L	If bit m is one, event m causes the CAPn_L (UNIFY = 0) or the CAPn (UNIFY = 1) register to be loaded (event 0 = bit 0, event 1 = bit 1,..., event 7 = bit 7).	0
15:8	-	Reserved.	-
23:16	CAPCONn_H	If bit m is one, event m causes the CAPn_H (UNIFY = 0) register to be loaded (event 0 = bit 16, event 1 = bit 17,..., event 7 = bit 23).	0
31:24	-	Reserved.	-

16.6.24 SCT event enable registers 0 to 7

Each event can be enabled in selected states and disabled in others. Each event defined in the EV_CTRL register has one associated event enable register that can enable or disable the event for each available state.

Each event has one associated SCT event state mask register that allow this event to happen in one or more states of the counter selected by the HEVENT bit in the corresponding EVn_CTRL register.

An event n is disabled when its EVn_STATE register contains all zeros, since it is masked regardless of the current state.

In simple applications that do not use states, write 0x01 to this register to enable each event in exactly one state. Since the state doesn't change (that is, the state variable always remains at its reset value of 0), writing 0x01 permanently enables this event.

Table 246. SCT event state mask registers 0 to 7 (EV[0:7]_STATE, addresses 0x5000 4300 (EV0_STATE) to 0x5000 4338 (EV7_STATE)) bit description

Bit	Symbol	Description	Reset value
7:0	STATEMSKn	If bit m is one, event n (n= 0 to 7) happens in state m of the counter selected by the HEVENT bit (m = state number; state 0 = bit 0, state 1= bit 1,..., state 7 = bit 7).	0
31:8	-	Reserved.	-

16.6.25 SCT event control registers 0 to 7

This register defines the conditions for event n (n = 0 to 7) to occur, aside from the event enable (EVn_STATE) registers. Most events are associated with a particular counter (H, L, or unified), in which case the event can depend on a match to that register. The other possible ingredient of an event is a selected input or output signal.

This is a rewrite of the section above:

This register defines the conditions for an event to occur based on the counter values or input and output states.Once the event is configured, it can trigger any of the actions for which it has been selected (for example stop the counter and toggle an output) unless the event is blocked in the current state of the SCT or the counter is halted. To block a particular event from occurring in any given context, use the EV_STATE register. To block all events for a given counter, set the HALT bit in the CTRL register or select an event to halt the counter.

An event can be programmed to occur based on a selected input or output edge or level and/or based on its counter value matching a selected match register. In bi-directional mode, events can also be enabled based on the direction of count.

When the UNIFY bit is 0, each event is associated with a particular counter by the HEVENT bit in its event control register. An event is permanently disabled when its event state mask register contains all 0s.

Each event can modify its counter STATE value. If more than one event associated with the same counter occurs in a given clock cycle, only the state change specified for the highest-numbered event among them takes place. Other actions dictated by any simultaneously occurring events all take place.

Table 247. SCT event control register 0 to 7 (EV[0:7]_CTRL, address 0x5000 4304 (EV0_CTRL) to 0x5000 433C (EV7_CTRL)) bit description

Bit	Symbol	Value	Description	Reset value
3:0	MATCHSEL	-	Selects the Match register associated with this event (if any). A match can occur only when the counter selected by the HEVENT bit is running.	0
4	HEVENT		Select L/H counter. Do not set this bit if UNIFY = 1.	0
		0	Selects the L state and the L match register selected by MATCHSEL.	
		1	Selects the H state and the H match register selected by MATCHSEL.	
5	OUTSEL		Input/output select	0
		0	Selects the inputs elected by IOSEL.	
		1	Selects the outputs selected by IOSEL.	
9:6	IOSEL	-	Selects the input or output signal number (0 to 3 for inputs or 0 to 5 for outputs) associated with this event (if any). Do not select an input in this register, if CLKMODE is 1x. In this case the clock input is an implicit ingredient of every event.	0
11:10	IOCOND		Selects the I/O condition for event n. (The detection of edges on outputs lag the conditions that switch the outputs by one SCT clock). In order to guarantee proper edge/state detection, an input must have a minimum pulse width of at least one SCT clock period .	0
		0x0	LOW	
		0x1	Rise	
		0x2	Fall	
		0x3	HIGH	
13:12	COMBMODE		Selects how the specified match and I/O condition are used and combined.	
		0x0	OR. The event occurs when either the specified match or I/O condition occurs.	
		0x1	MATCH. Uses the specified match only.	
		0x2	IO. Uses the specified I/O condition only.	
		0x3	AND. The event occurs when the specified match and I/O condition occur simultaneously.	
14	STATELD		This bit controls how the STATEV value modifies the state selected by HEVENT when this event is the highest-numbered event occurring for that state.	
		0	STATEV value is added into STATE (the carry-out is ignored).	
		1	STATEV value is loaded into STATE.	
19:15	STATEV		This value is loaded into or added to the state selected by HEVENT, depending on STATELD, when this event is the highest-numbered event occurring for that state. If STATELD and STATEV are both zero, there is no change to the STATE value.	
20	MATCHMEM		If this bit is one and the COMBMODE field specifies a match component to the triggering of this event, then a match is considered to be active whenever the counter value is GREATER THAN OR EQUAL TO the value specified in the match register when counting up, LESS THEN OR EQUAL TO the match value when counting down. If this bit is zero, a match is only be active during the cycle when the counter is equal to the match value.	

Table 247. SCT event control register 0 to 7 (EV[0:7]_CTRL, address 0x5000 4304 (EV0_CTRL) to 0x5000 433C (EV7_CTRL)) bit description

Bit	Symbol	Value	Description	Reset value
22:21	DIRECTION		Direction qualifier for event generation. This field only applies when the counters are operating in BIDIR mode. If BIDIR = 0, the SCT ignores this field. Value 0x3 is reserved.	
		0x0	Direction independent. This event is triggered regardless of the count direction.	
		0x1	Counting up. This event is triggered only during up-counting when BIDIR = 1.	
		0x2	Counting down. This event is triggered only during down-counting when BIDIR = 1.	
31:23	-		Reserved	

16.6.26 SCT output set registers 0 to 5

Based on a selected event, each SCT output can be set.

There is one output set register for each SCT output which selects which events can set that output. Each bit of an output set register is associated with a different event (bit 0 with event 0, etc.). A selected event can set or clear the output depending on the setting of the SETCLRn field in the OUTPUTDIRCTRL register. To define the actual event that sets the output (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Table 248. SCT output set register (OUT[0:5]_SET, address 0x5000 4500 (OUT0_SET) to 0x5000 4528 (OUT5_SET) bit description

Bit	Symbol	Description	Reset value
7:0	SET	A 1 in bit m selects event m to set output n (or clear it if SETCLRn = 0x1 or 0x2) event 0 = bit 0, event 1 = bit 1,..., event 7 = bit 7. When the counter is used in bi-directional mode, it is possible to reverse the action specified by the output set and clear registers when counting down, See the OUTPUTCTRL register.	0
31:8	-	Reserved	

16.6.27 SCT output clear registers 0 to 5

Based on a selected event, each SCT output can be cleared.

There is one register for each SCT output which selects which events can clear that output. Each bit of an output clear register is associated with a different event (bit 0 with event 0, etc.). A selected event can clear or set the output depending on the setting of the SETCLRn field in the OUTPUTDIRCTRL register. To define the actual event that clears the output (a match, an I/O pin toggle, etc.), see the EVn_CTRL register.

Remark: If the SCTimer/PWM is operating as two 16-bit counters, events can only modify the state of the outputs when neither counter is halted. This is true regardless of what triggered the event.

Table 249. SCT output clear register (OUT[0:5]_CLR, address 0x5000 4504 (OUT0_CLR) to 0x5000 452C (OUT5_CLR)) bit description

Bit	Symbol	Description	Reset value
7:0	CLR	A 1 in bit m selects event m to clear output n (or set it if SETCLRn = 0x1 or 0x2) event 0 = bit 0, event 1 = bit 1,..., event 7 = bit 7. When the counter is used in bi-directional mode, it is possible to reverse the action specified by the output set and clear registers when counting down, See the OUTPUTCTRL register.	0
31:8	-	Reserved	

16.7 Functional description

16.7.1 Match logic

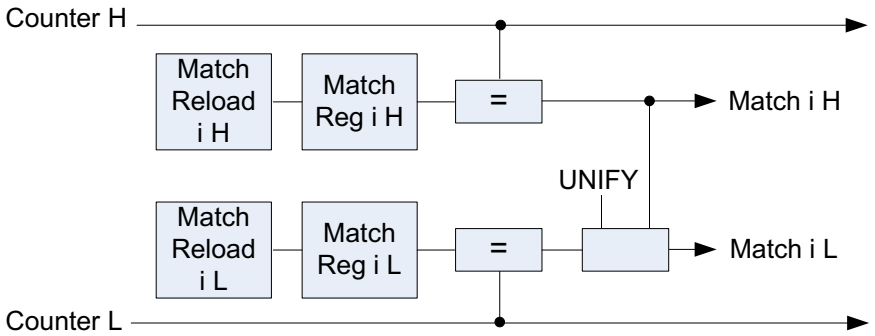


Fig 37. Match logic

16.7.2 Capture logic

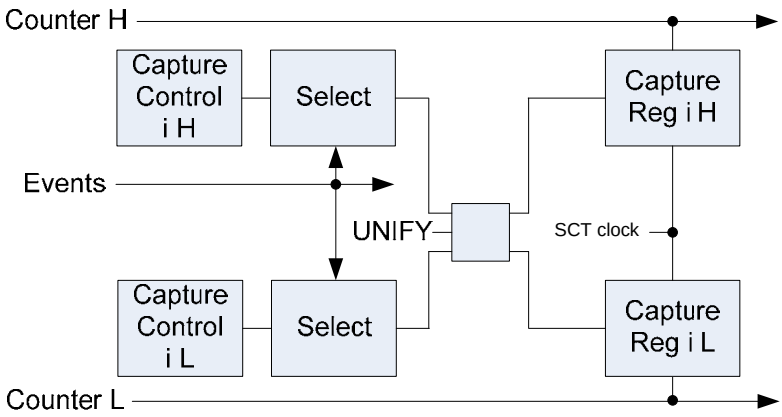


Fig 38. Capture logic

16.7.3 Event selection

State variables allow control of the SCT across more than one cycle of the counter. Counter matches, input/output edges, and state values are combined into a set of general-purpose events that can switch outputs, request interrupts, and change state values.

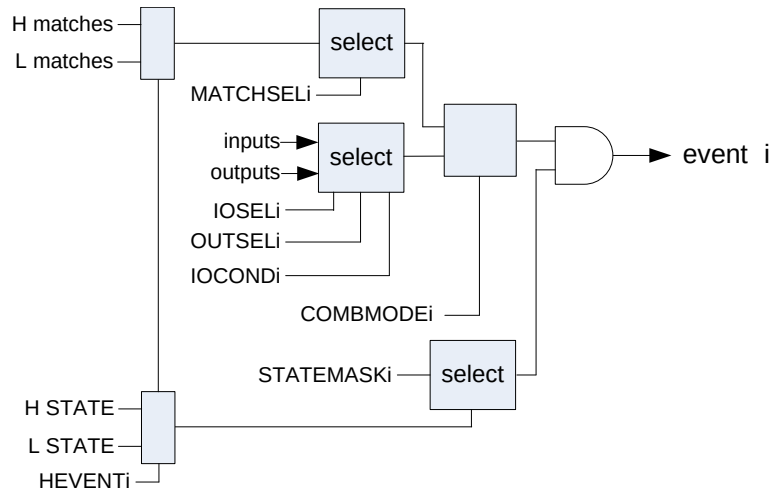


Fig 39. Event selection

16.7.4 Output generation

Figure 40 shows one output slice of the SCT.

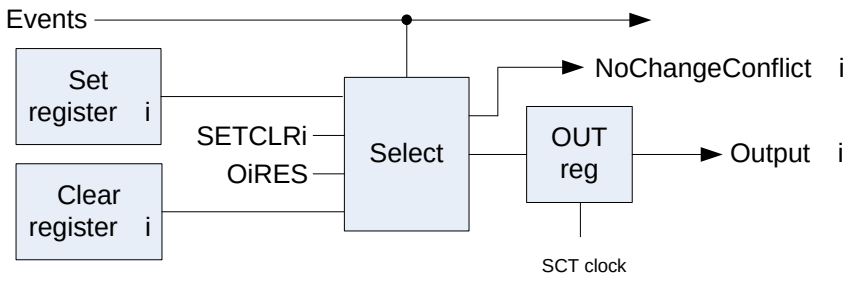


Fig 40. Output slice i

16.7.5 State logic

The SCT can be configured as a timer/counter with multiple programmable states. The states are user-defined through the events that can be captured in each particular state. In a multi-state SCT. the SCT can change from one state to another state when a user-defined event triggers a state change. The state change is triggered through each event's EV_CTRL register in one of the following ways:

- The event can increment the current state number by a new value.
- The event can write a new state value.

If an event increments the state number beyond the number of available states, the SCT enters a locked state in which all further events are ignored while the counter is still running. Software must interfere to change out of this state.

Software can capture the counter value (and potentially create an interrupt and write to all outputs) when the event moving the SCT into a locked state occurs.Later, while the SCT is in the locked state, software can read the counter again to record the time passed since the locking event and can also read the state variable to obtain the current state number

If the SCT registers an event that forces an abort, putting the SCT in a locked state can be a safe way to record the time that has passed since the abort event while no new events are allowed to occur. Since multiple states (any state number between the maximum implemented state and 31) are locked states, multiple abort or error events can be defined each incrementing the state number by a different value.

16.7.6 Interrupt generation

The SCT generates one interrupt to the NVIC.

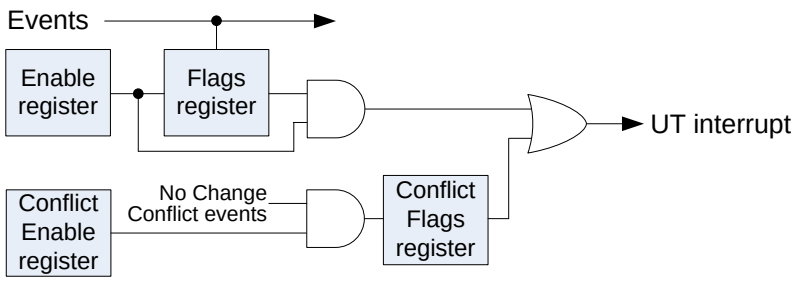


Fig 41. SCT interrupt generation

16.7.7 Clearing the prescaler

When enabled by a non-zero PRE field in the Control register, the prescaler acts as a clock divider for the counter, like a fractional part of the counter value. The prescaler is cleared whenever the counter is cleared or loaded for any of the following reasons:

- Hardware reset
- Software writing to the counter register
- Software writing a 1 to the CLRCTR bit in the control register
- an event selected by a 1 in the counter limit register when BIDIR = 0

When BIDIR is 0, a limit event caused by an I/O signal can clear a non-zero prescaler. However, a limit event caused by a Match only clears a non-zero prescaler in one special case as described Section 16.7.8.

A limit event when BIDIR is 1 does not clear the prescaler. Rather it clears the DOWN bit in the Control register, and decrements the counter on the same clock if the counter is enabled in that clock.

16.7.8 Match vs. I/O events

Counter operation is complicated by the prescaler and by clock mode 01 in which the SCT clock is the bus clock. However, the prescaler and counter are enabled to count only when a selected edge is detected on a clock input.

- The prescaler is enabled when the clock mode is not 01, or when the input edge selected by the CLKSEL field is detected.
- The counter is enabled when the prescaler is enabled, and (PRELIM=0 or the prescaler is equal to the value in PRELIM).

An I/O component of an event can occur in any SCT clock when its counter HALT bit is 0. In general, a Match component of an event can only occur in a UT clock when its counter HALT and STOP bits are both 0 and the counter is enabled.

Table 250 shows when the various kinds of events can occur.

Table 250. Event conditions

COMBMODE	IOMODE	Event can occur on clock:
IO	Any	Event can occur whenever HALT = 0 (type A).
MATCH	Any	Event can occur when HALT = 0 and STOP = 0 and the counter is enabled (type C).
OR	Any	From the IO component: Event can occur whenever HALT = 0 (A). From the match component: Event can occur when HALT = 0 and STOP = 0 and the counter is enabled (C).
AND	LOW or HIGH	Event can occur when HALT = 0 and STOP = 0 and the counter is enabled (C).
AND	RISE or FALL	Event can occur whenever HALT = 0 (A).

16.7.9 SCT operation

In its simplest, single-state configuration, the SCT operates as an event controlled one- or bidirectional counter. Events can be configured to be counter match events, an input or output level, transitions on an input or output pin, or a combination of match and input/output behavior. In response to an event, the SCT output or outputs can transition, or the SCT can perform other actions such as creating an interrupt or starting, stopping, or resetting the counter. Multiple simultaneous actions are allowed for each event. Furthermore, any number of events can trigger one specific action of the SCT.

An action or multiple actions of the SCT uniquely define an event. A state is defined by which events are enabled to trigger an SCT action or actions in any stage of the counter. Events not selected for this state are ignored.

In a multi-state configuration, states change in response to events. A state change is an additional action that the SCT can perform when the event occurs. When an event is configured to change the state, the new state defines a new set of events resulting in different actions of the SCT. Through multiple cycles of the counter, events can change the state multiple times and thus create a large variety of event controlled transitions on the SCT outputs and/or interrupts.

Once configured, the SCT can run continuously without software intervention and can generate multiple output patterns entirely under the control of events.

- To configure the SCT, see Section 16.7.10.
- To start, run, and stop the SCT, see Section 16.7.11.
- To configure the SCT as simple event controlled counter/timer, see Section 16.7.12.

16.7.10 Configure the SCT

To set up the SCT for multiple events and states, perform the following configuration steps:

16.7.10.1 Configure the counter

1. Configure the L and H counters in the CONFIG register by selecting two independent 16-bit counters (L counter and H counter) or one combined 32-bit counter in the UNIFY field.
2. Select the SCT clock source in the CONFIG register (fields CLKMODE and CLKSEL) from any of the inputs or an internal clock.

16.7.10.2 Configure the match and capture registers

1. Select how many match and capture registers the application uses (total of up to 5):
   - In the REGMODE register, select for each of the 5 match/capture register pairs whether the register is used as a match register or capture register.
2. Define match conditions for each match register selected:
   - Each match register MATCH sets one match value, if a 32-bit counter is used, or two match values, if the L and H 16-bit counters are used.
   - Each match reload register MATCHRELOAD sets a reload value that is loaded into the match register when the counter reaches a limit condition or the value 0.

16.7.10.3 Configure events and event responses

1. Define when each event can occur in the following way in the EVn_CTRL registers (up to 6, one register per event):
   - Select whether the event occurs on an input or output changing, on an input or output level, a match condition of the counter, or a combination of match and input/output conditions in field COMBMODE.
   - For a match condition:

     Select the match register that contains the match condition for the event to occur. Enter the number of the selected match register in field MATCHSEL.

     If using L and H counters, define whether the event occurs on matching the L or the H counter in field HEVENT.
   - For an SCT input or output level or transition:

     Select the input number or the output number that is associated with this event in fields IOSEL and OUTSEL.

     Define how the selected input or output triggers the event (edge or level sensitive) in field IOCOND.
2. Define what the effect of each event is on the SCT outputs in the OUTn_SET or OUTn_CLR registers (up to 4 outputs, one register per output):
   - For each SCT output, select which events set or clear this output. More than one event can change the output, and each event can change multiple outputs.
3. Define how each event affects the counter:
   - Set the corresponding event bit in the LIMIT register for the event to set an upper limit for the counter.

     When a limit event occurs in unidirectional mode, the counter is cleared to zero and begins counting up on the next clock edge.

     When a limit event occurs in bidirectional mode, the counter begins to count down from the current value on the next clock edge.
   - Set the corresponding event bit in the HALT register for the event to halt the counter. If the counter is halted, it stops counting and no new events can occur. The counter operation can only be restored by clearing the HALT_L and/or the HALT_H bits in the CTRL register.
   - Set the corresponding event bit in the STOP register for the event to stop the counter. If the counter is stopped, it stops counting. However, an event that is configured as a transition on an input/output can restart the counter.
   - Set the corresponding event bit in the START register for the event to restart the counting. Only events that are defined by an input changing can be used to restart the counter.
4. Define which events contribute to the SCT interrupt:
   - Set the corresponding event bit in the EVEN and the EVFLAG registers to enable the event to contribute to the SCT interrupt.

16.7.10.4 Configure multiple states

1. In the EVn_STATE register for each event (up to 6 events, one register per event), select the state or states (up to 2) in which this event is allowed to occur. Each state can be selected for more than one event.
2. Determine how the event affects the system state:

   In the EVn_CTRL registers (up to 6 events, one register per event), set the new state value in the STATEV field for this event. If the event is the highest numbered in the current state, this value is either added to the existing state value or replaces the existing state value, depending on the field STATELD.

   Remark: If there are higher numbered events in the current state, this event cannot change the state.

   If the STATEV and STATELD values are set to zero, the state does not change.

16.7.10.5 Miscellaneous options

- There are a certain (selectable) number of capture registers. Each capture register can be programmed to capture the counter contents when one or more events occur.
- If the counter is in bidirectional mode, the effect of set and clear of an output can be made to depend on whether the counter is counting up or down by writing to the OUTPUTDIRCTRL register.

16.7.11 Run the SCT

1. Configure the SCT (see Section 16.7.10 “Configure the SCT”).
2. Write to the STATE register to define the initial state. By default the initial state is state 0.
3. To start the SCT, write to the CTRL register:
   - Clear the counters.
   - Clear or set the STOP_L and/or STOP_H bits.

     Remark: The counter starts counting once the STOP bit is cleared as well. If the STOP bit is set, the SCT waits instead for an event to occur that is configured to start the counter.
   - For each counter, select unidirectional or bidirectional counting mode (field BIDIR_L and/or BIDIR_H).
   - Select the prescale factor for the counter clock (CTRL register).
   - Clear the HALT_L and/or HALT_H bit. By default, the counters are halted and no events can occur.
4. To stop the counters by software at any time, stop or halt the counter (write to STOP_L and/or STOP_H bits or HALT_L and/or HALT_H bits in the CTRL register).
   - When the counters are stopped, both an event configured to clear the STOP bit or software writing a zero to the STOP bit can start the counter again.
   - When the counter are halted, only a software write to clear the HALT bit can start the counter again. No events can occur.
   - When the counters are halted, software can set any SCT output HIGH or LOW directly by writing to the OUT register.

The current state can be read at any time by reading the STATE register.

To change the current state by software (that is independently of any event occurring), set the HALT bit and write to the STATE register to change the state value. Writing to the STATE register is only allowed when the counter is halted (the HALT_L and/or HALT_H bits are set) and no events can occur.

16.7.12 Configure the SCT without using states

The SCT can be used as standard counter/timer with external capture inputs and match outputs without using the state logic. To operate the SCT without states, configure the SCT as follows:

- Write zero to the STATE register (zero is the default).
- Write zero to the STATELD and STATEV fields in the EVCTRL registers for each event.
- Write 0x1 to the EVn_STATE register of each event. Writing 0x1 enables the event.

  In effect, the event is allowed to occur in a single state which never changes while the counter is running.

16.7.13 SCT PWM Example

Figure 42 shows a simple application of the SCT using two sets of match events (EV0/1 and EV3/4) to set/clear SCT output 0. The timer is automatically reset whenever it reaches the MAT0 match value.

In the initial state 0, match event EV0 sets output 0 to HIGH and match event EV1 clears output 0. The SCT input 0 is monitored: If input0 is found LOW by the next time the timer is reset(EV2), the state is changed to state 1, and EV3/4 are enabled, which create the same output but triggered by different match values. If input 0 is found HIGH by the next time the timer is reset, the associated event (EV5) causes the state to change back to state 0where the events EV0 and EV1 are enabled.

The example uses the following SCT configuration:

- 1 input
- 1 output
- 5 match registers
- 6 events and match 0 used with autolimit function
- 2 states

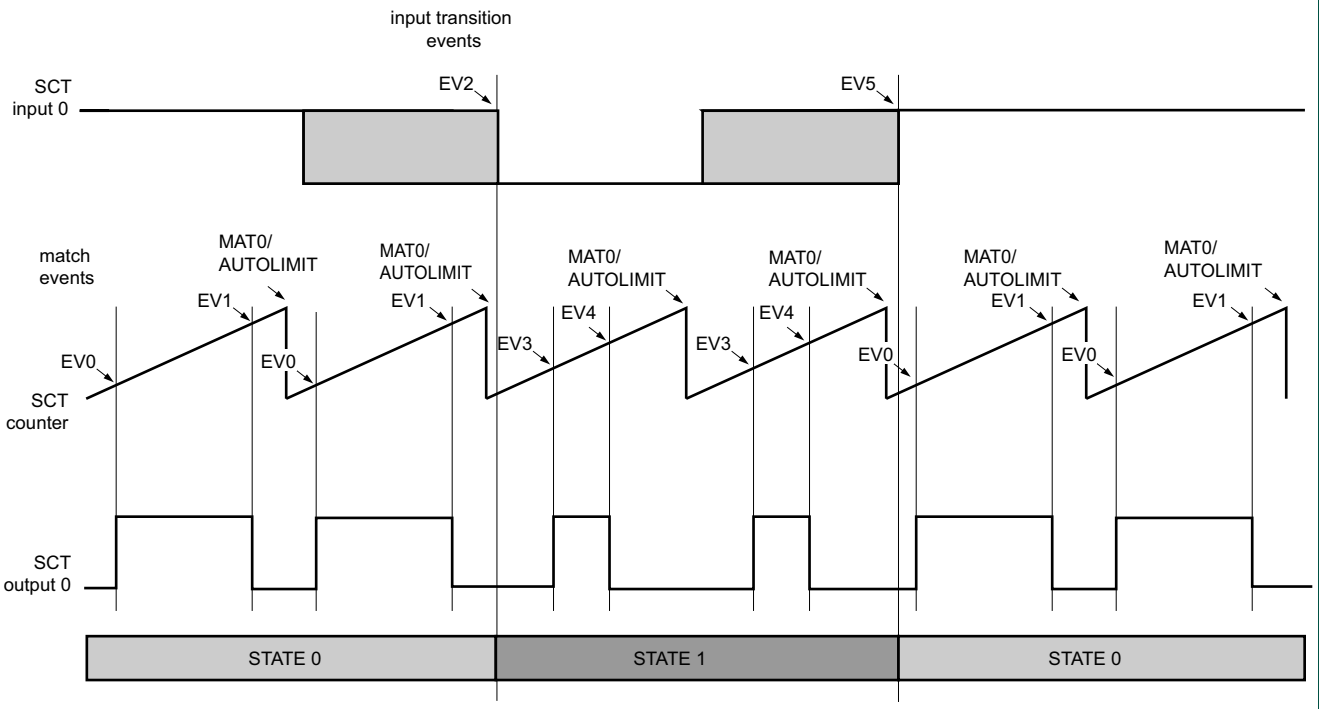


Fig 42. SCT configuration example

This application of the SCT uses the following configuration (all register values not listed in Table 251 are set to their default values):

Table 251. SCT configuration example

Configuration	Registers	Setting
Counter	CONFIG	Uses one counter (UNIFY = 1).
	CONFIG	Enable the autolimit for MAT0. (AUTOLIMIT = 1.)
	CTRL	Uses unidirectional counter (BIDIR_L = 0).
Clock base	CONFIG	Uses default values for clock configuration.
Match/Capture registers	REGMODE	Configure one match register for each match event by setting REGMODE_L bits 0,1, 2, 3, 4 to 0. This is the default.
Define match values	MATCH0/1/2/3/4	Set a match value MATCH0/1/2/4/5_L in each register. The match 0 register serves as an automatic limit event that resets the counter. without using an event. To enable the automatic limit, set the AUTOLIMIT bit in the CONFIG register.
Define match reload values	MATCHREL0/1/2/3/4	Set a match reload value RELOAD0/1/2/3/4_L in each register (same as the match value in this example).
Define when event 0 occurs	EV0_CTRL	• Set COMBMODE = 0x1. Event 0 uses match condition only. • Set MATCHSEL = 1. Select match value of match register 1. The match value of MAT1 is associated with event 0.
Define when event 1 occurs	EV1_CTRL	• Set COMBMODE = 0x1. Event 1 uses match condition only. • Set MATCHSEL = 2 Select match value of match register 2. The match value of MAT2 is associated with event 1.

Table 251. SCT configuration example

Configuration	Registers	Setting
Define when event 2 occurs	EV2_CTRL	• Set COMBMODE = 0x3. Event 2 uses match condition and I/O condition. • Set IOSEL = 0. Select input 0. • Set IOCOND = 0x0. Input 0 is LOW. • Set MATCHSEL = 0. Chooses match register 0 to qualify the event.
Define how event 2 changes the state	EV2_CTRL	Set STATEV bits to 1 and the STATED bit to 1. Event 2 changes the state to state 1.
Define when event 3 occurs	EV3_CTRL	• Set COMBMODE = 0x1. Event 3 uses match condition only. • Set MATCHSEL = 0x3. Select match value of match register 3. The match value of MAT3 is associated with event 3..
Define when event 4 occurs	EV4_CTRL	• Set COMBMODE = 0x1. Event 4 uses match condition only. • Set MATCHSEL = 0x4. Select match value of match register 4.The match value of MAT4 is associated with event 4.
Define when event 5 occurs	EV5_CTRL	• Set COMBMODE = 0x3. Event 5 uses match condition and I/O condition. • Set IOSEL = 0. Select input 0. • Set IOCOND = 0x3. Input 0 is HIGH. • Set MATCHSEL = 0. Chooses match register 0 to qualify the event.
Define how event 5 changes the state	EV5_CTRL	Set STATEV bits to 0 and the STATED bit to 1. Event 5 changes the state to state 0.
Define by which events output 0 is set	OUT0_SET	Set SET0 bits 0 (for event 0) and 3 (for event 3) to one to set the output when these events 0 and 3 occur.
Define by which events output 0 is cleared	OUT0_CLR	Set CLR0 bits 1 (for events 1) and 4 (for event 4) to one to clear the output when events 1 and 4 occur.
Configure states in which event 0 is enabled	EV0_STATE	Set STATEMSK0 bit 0 to 1. Set all other bits to 0. Event 0 is enabled in state 0.
Configure states in which event 1 is enabled	EV1_STATE	Set STATEMSK1 bit 0 to 1. Set all other bits to 0. Event 1 is enabled in state 0.
Configure states in which event 2 is enabled	EV2_STATE	Set STATEMSK2 bit 0 to 1. Set all other bits to 0. Event 2 is enabled in state 0.
Configure states in which event 3 is enabled	EV3_STATE	Set STATEMSK3 bit 1 to 1. Set all other bits to 0. Event 3 is enabled in state 1.
Configure states in which event 4 is enabled	EV4_STATE	Set STATEMSK4 bit 1 to 1. Set all other bits to 0. Event 4 is enabled in state 1.
Configure states in which event 5 is enabled	EV5_STATE	Set STATEMSK5 bit 1 to 1. Set all other bits to 0. Event 5 is enabled in state 1.

17.1 How to read this chapter

The watchdog timer is identical on all LPC82x 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 value can optionally be protected such that it can only be changed after the “warning interrupt” time is reached.
- Flag to indicate Watchdog reset.
- The Watchdog clock (WDCLK) source is the 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 35.
- Enable the WWDT clock source (the watchdog oscillator) in the PDRUNCFG register (Table 54). 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 51).

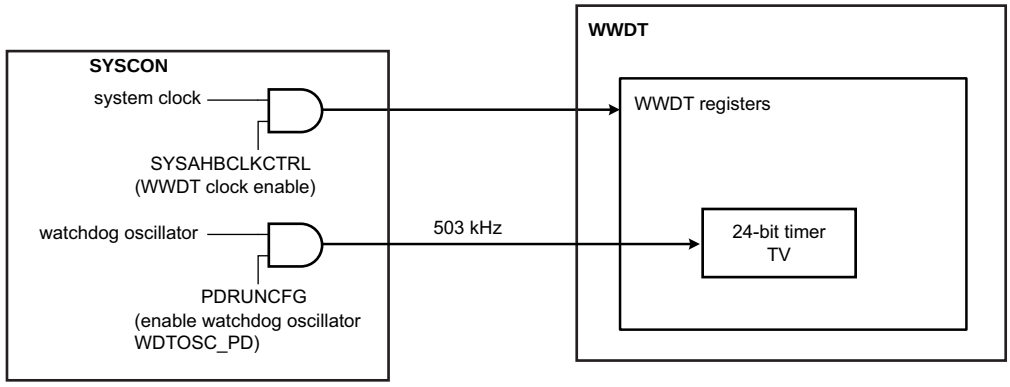


Fig 43. WWDT clocking

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.
- 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 44. The synchronization logic (PCLK - WDCLK) is not shown in the block diagram.

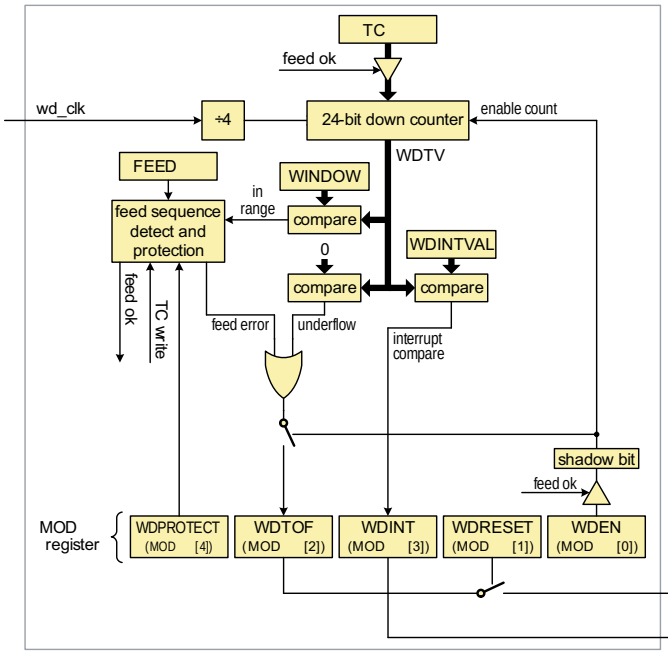


Fig 44. Windowed Watchdog timer block diagram

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 5). 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
- Changing the WWDT reload value

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 61).

17.5.3.2 Changing the WWDT reload value

If bit 4 is set in the WWDT MOD register, the watchdog time-out value (TC) can be changed only after the counter is below the value of WDWARNINT and WDWINDOW.

The reload overwrite lock mechanism can only be disabled by a reset of any type.

17.6 Register description

The Watchdog Timer contains the registers shown in Table 252.

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

Table 252. 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 253
TC	R/W	0x004	Watchdog timer constant register. This 24-bit register determines the time-out value.	0xFF	Table 255
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 256
TV	RO	0x00C	Watchdog timer value register. This 24-bit register reads out the current value of the Watchdog timer.	0xFF	Table 257
-	-	0x010	Reserved	-	-
WARNINT	R/W	0x014	Watchdog Warning Interrupt compare value.	0	Table 258
WINDOW	R/W	0x018	Watchdog Window compare value.	0xFF FFFF	Table 259

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 253. 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 software. Causes a chip reset if WDRESET = 1.	0 (only after external reset)

Table 253. 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 software.	0
4	WDPROTECT		Watchdog update mode. This bit can be set once by software and is only cleared by a reset.	0
		0	The watchdog time-out value (TC) can be changed at any time.	
		1	The watchdog time-out value (TC) can be changed only after the counter is below the value of WDWARNINT 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 51 “Start logic 1 interrupt wake-up enable register (STARTERP1, address 0x4004 8214) bit description”

Table 254. Watchdog operating modes selection

WDEN	WDRESET	Mode of Operation
0	X (0 or 1)	Debug/Operate without the Watchdog running.
1	0	Watchdog interrupt mode: the watchdog warning interrupt will be generated but watchdog reset will not. When this mode is selected, the watchdog counter reaching the value specified by WDWARNINT will set the WDINT flag and the Watchdog interrupt request will be generated.
1	1	Watchdog reset mode: both the watchdog interrupt and watchdog reset are enabled. When this mode is selected, the watchdog counter reaching the value specified by WDWARNINT will set the WDINT flag and the Watchdog interrupt request will be generated, and the watchdog counter reaching zero will reset the microcontroller. A watchdog feed prior to reaching the value of WDWINDOW will also cause a watchdog reset.

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 change the value of TC before the watchdog counter is below the values of WDWARNINT and WDWINDOW will cause a watchdog reset and set the WDTOF flag.

Table 255. 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 256. 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 257. 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 matches the value defined by WARNINT, an interrupt will be generated after the subsequent WDCLK.

A match of the watchdog timer counter to WARNINT occurs when the bottom 10 bits of the counter have the same value as the 10 bits of WARNINT, and the remaining upper bits of the counter are all 0. This gives a maximum time of 1,023 watchdog timer counts (4,096 watchdog clocks) for the interrupt to occur prior to a watchdog event. If WARNINT is 0, the interrupt will occur at the same time as the watchdog event.

Table 258. 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 259. 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.

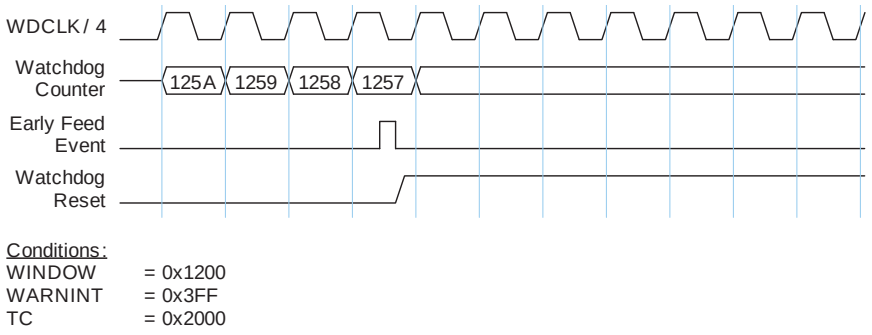


Fig 45. Early watchdog feed with windowed mode enabled

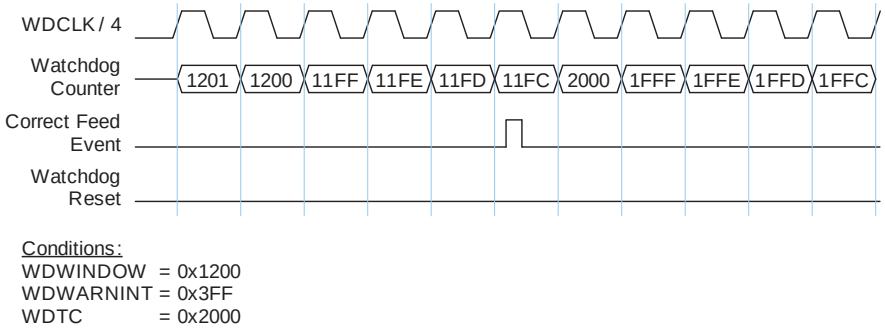


Fig 46. Correct watchdog feed with windowed mode enabled

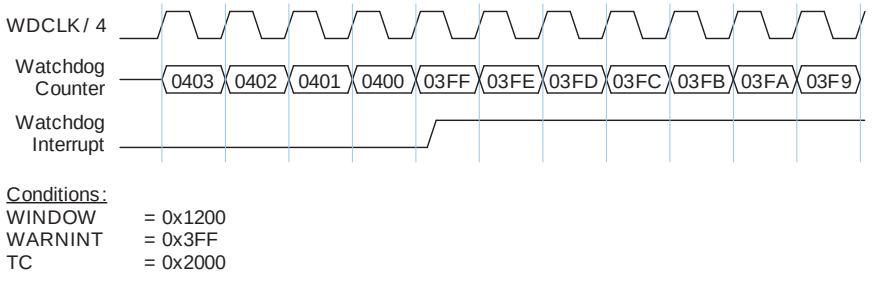


Fig 47. Watchdog warning interrupt

18.1 How to read this chapter

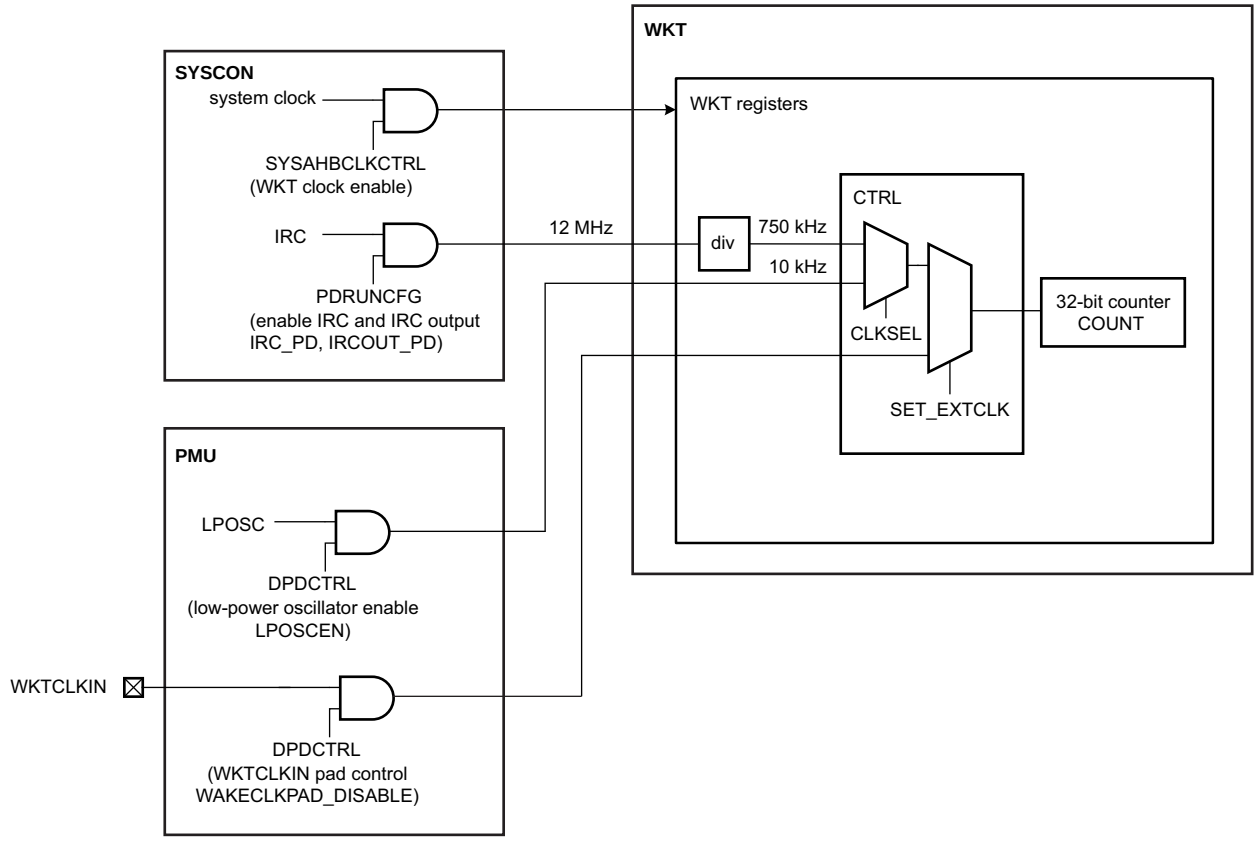
The self-wake-up timer is available on all LPC82x parts.

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 resides in a separate, always-on power domain.
- The WKT supports three clock sources: The IRC, the internal low-power oscillator, or the WKTCLKIN pin. The low-power oscillator and the external clock are valid clock sources in all power modes including deep power-down. The IRC 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 any low power mode or for general-purpose timing.

18.3 Basic configuration

- In the SYSAHBCLKCTRL register, set bit 9 (Table 35) to enable the clock to the register interface.
- Clear the WKT reset using the PRESETCTRL register (Table 23).
- The WKT interrupt is connected to interrupt #15 in the NVIC. See Table 5.
- Enable the low power oscillator in the PMU (Table 63).
- Enable the IRC and IRC output in the PDRUNCFG register if used as the clock source for the timer (Table 54).
- To use an external clock source for the self-wake-up timer, enable the clock input for pin PIO0_28 in the DPDCRTL register (Table 63) and enable the external clock option in the self-wake-up timer CTRL register (see Table 261). The external clock source can be used in all power modes including deep power-down mode.
- Disable the external clock input in the DPDCTRL register to minimize power consumption if not using the external clock source option. See Table 63.
- Disable the WAKEUP function in the DPDCTRL register to minimize power consumption if the part does not need to wake up from deep power-down mode via a pin. See Table 63.
- See Section 6.7.1 to enable the various power down modes.



Turn off the low-power oscillator in deep power-down mode if not used for wake-up timing.

Fig 48. WKT clocking

18.4 Pin description

The WKT can use a clock input on the external pin PIO0_28 for clocking the wake-up timer in sleep, deep-sleep, power-down, and deep power-down modes. Select the external clock source by setting bit SET_EXTCLK in the CTRL register (see Table 261).

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 clock sources:

- A 750 kHz clock derived from the IRC oscillator. This is the default clock,
- A 10 kHz, low-power clock with a dedicated on-chip oscillator as clock source.
- An external clock on the WKTCLKIN pin.

The IRC-derived clock is much more accurate than the alternative, low-power clock. However, the IRC 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 IRC is disabled.

The alternative clock source is a (nominally) 10 kHz, low-power clock, sourced from a dedicated oscillator. This oscillator resides in the always-on voltage domain, so it can be programmed to continue operating in Deep power-down mode when power is removed from the rest of the part. This clock is also be available during other low-power modes when the IRC clock is shut-down.

The Low-Power oscillator is not accurate (approximately +/- 40 % over process and temperature). The frequency may still drift while counting is in progress due to reduced chip temperature after a low-power mode is entered.

An external clock on the WKTCLKIN pin can be used to time the self-wake-up timer in all low power modes, including deep power-down.

18.6 Register description

Table 260. 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 261
COUNT	R/W	0xC	Counter register.	-	Table 262

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 261. Control register (CTRL, address 0x4000 8000) bit description

Bit	Symbol	Value	Description	Reset value
0	CLKSEL		Select the self-wake-up timer clock source. **Remark:** This bit only has an effect if the SEL_EXTCLK bit is not set.	0
		0	Divided IRC clock. This clock runs at 750 kHz and provides time-out periods of up to approximately 95 minutes in 1.33 µs increments. **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 (nominally) 10 kHz clock and provides time-out periods of up to approximately 119 hours in 100 µs increments. The accuracy of this clock is limited to +/- 40 % over temperature and processing. **Remark:** This clock is available in all power modes. Prior to use, the low-power oscillator must be enabled. The oscillator must also be set to remain active in Deep power-down if needed.	

Table 261. Control register (CTRL, address 0x4000 8000) bit description

Bit	Symbol	Value	Description	Reset value
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 including Deep power-down if the clock source is the low power oscillator. Writing a 1 clears this status bit.	
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 262. 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. A read reflects the current value of the timer.	-

19.1 How to read this chapter

The MRT is available on all LPC82x parts.

19.2 Features

- 31-bit interrupt timer
- Four 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 35) to enable the clock to the register interface.
- Clear the MRT reset using the PRESETCTRL register (Table 23).
- The global MRT interrupt is connected to interrupt #10 in the NVIC.

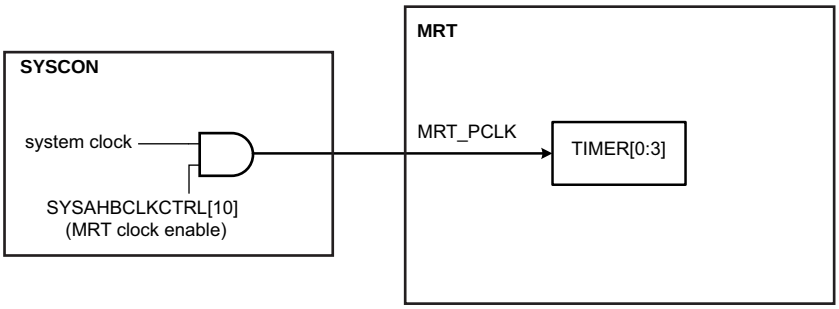


Fig 49. MRT clocking

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 266.

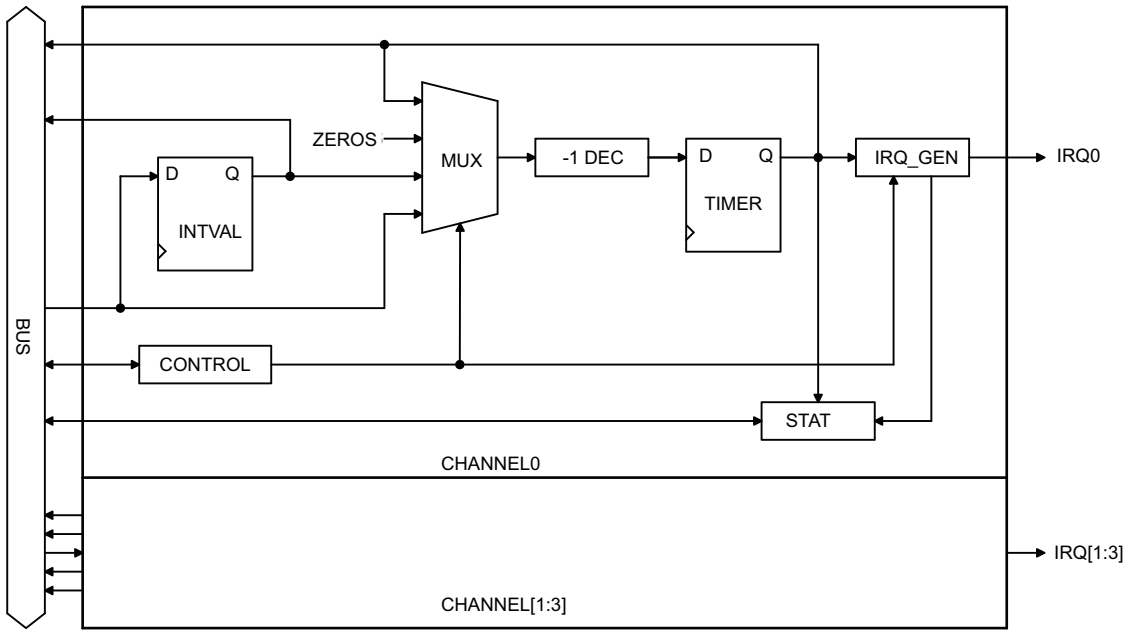


Fig 50. MRT block diagram

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 263 are POR reset values.

Table 263. 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 264
TIMER0	R	0x4	MRT0 Timer register. This register reads the value of the down counter.	0x7FFF FFFF	Table 265
CTRL0	R/W	0x8	MRT0 Control register. This register controls the MRT0 modes.	0	Table 266
STAT0	R/W	0xC	MRT0 Status register.	0	Table 267
INTVAL1	R/W	0x10	MRT1 Time interval value register. This value is loaded into the TIMER1 register.	0	Table 264
TIMER1	R/W	0x14	MRT1 Timer register. This register reads the value of the down counter.	0x7FFF FFFF	Table 265
CTRL1	R/W	0x18	MRT1 Control register. This register controls the MRT1 modes.	0	Table 266
STAT1	R/W	0x1C	MRT1 Status register.	0	Table 267
INTVAL2	R/W	0x20	MRT2 Time interval value register. This value is loaded into the TIMER2 register.	0	Table 264
TIMER2	R/W	0x24	MRT2 Timer register. This register reads the value of the down counter.	0x7FFF FFFF	Table 265
CTRL2	R/W	0x28	MRT2 Control register. This register controls the MRT2 modes.	0	Table 266
STAT2	R/W	0x2C	MRT2 Status register.	0	Table 267
INTVAL3	R/W	0x30	MRT3 Time interval value register. This value is loaded into the TIMER3 register.	0	Table 264
TIMER3	R/W	0x34	MRT3 Timer register. This register reads the value of the down counter.	0x7FFF FFFF	Table 265
CTRL3	R/W	0x38	MRT3 Control register. This register controls the MRT modes.	0	Table 266
STAT3	R/W	0x3C	MRT3 Status register.	0	Table 267
IDLE_CH	R	0xF4	Idle channel register. This register returns the number of the first idle channel.	0	Table 268
IRQ_FLAG	R/W	0xF8	Global interrupt flag register	0	Table 269

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 264. 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. If the timer is idle, writing a non-zero value to this bit field starts the timer immediately. If the timer is running, writing a zero to this bit field does the following: • If LOAD = 1, the timer stops immediately. • 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 265. 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: INTVALn register is updated in the idle state. INTVALn register is updated with LOAD = 1. When the timer is in idle state, reading this bit fields returns -1 (0x00FF FFFF).	0x00FF FFFF
31	-	Reserved.	0

19.6.3 Control register

The control register configures the mode for each MRT and enables the interrupt.

Table 266. 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 267. 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. 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 Idle channel register

The idle channel register returns the lowest idle channel number. The channel is considered idle when both flags is the STATUS register (RUN and INTFLAG) are zero.

In an application with multiple timers running independently, you can calculate the register offset of the next idle timer by reading the idle channel number in this register. The idle channel register allows you set up the next idle timer without checking the idle state of each timer.

Table 268. 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

19.6.6 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 269. 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. 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. Writing a 1 to this bit clears the interrupt request.	
2	GFLAG2		Monitors the interrupt flag of TIMER2.	0
		0	No pending interrupt. Writing a zero is equivalent to no operation.	
		1	Pending interrupt. The interrupt is pending because TIMER2 has reached the end of the time interval. If the INTEN bit in the CONTROL2 register is also set to 1, the interrupt for timer channel 2 and the global interrupt are raised. Writing a 1 to this bit clears the interrupt request.	
3	GFLAG3		Monitors the interrupt flag of TIMER3.	0
		0	No pending interrupt. Writing a zero is equivalent to no operation.	
		1	Pending interrupt. The interrupt is pending because TIMER3 has reached the end of the time interval. If the INTEN bit in the CONTROL3 register is also set to 1, the interrupt for timer channel 3 and the global interrupt are raised. Writing a 1 to this bit clears the interrupt request.	
31:4	-		Reserved.	0

20.1 How to read this chapter

The SysTick timer is available on all LPC82x 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 271). 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 271).
3. The calibration value of the SysTick timer is contained in the SYSTCKCAL register in the system configuration block SYSCON (see Table 46).

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 51.

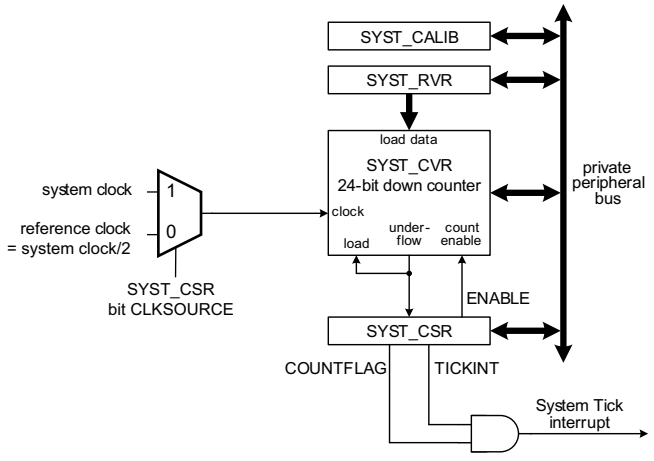


Fig 51. System tick timer block diagram

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 270. 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 271. 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 272. 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 273. 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 46).

Table 274. 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 5) 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 ADC is available on all parts. The number of available ADC channels depends on the package type.

Table 275. Pinout summary

Package	ADC channels available
TSSOP20	ADC_2, ADC_3, ADC_9, ADC_10, ADC_11
HVQFN33	ADC_0 to ADC_11

21.2 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 to VREFP (typically 3 V; not to exceed VDDA voltage level).
- 12-bit conversion rate of up to 1.2 Msamples/s.
- Burst conversion mode for single or multiple inputs.
- DMA support.
- Hardware calibration mode.

21.3 Basic configuration

Configure the ADC as follows:

- Use the PDRUNCFG register to power the ADC. See Table 54. 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 280 and Section 21.7.5.
- Use the SYSAHBCLKCTRL register (Table 35) 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 5.
- The ADC analog inputs are enabled in the switch matrix block.See Table 79.
- The power to the ADC block is controlled by the PDRUNCFG register in the SYSCON block. See Table 54.
- Calibration is required after every power-up or wake-up from Deep power-down mode. See Section 21.3.4 “Hardware self-calibration”.

- For a sampling rate higher than 1 Msamples/s, VDDA must be higher than 2.7 V. See Table 280.
- Configure the ADC for the appropriate analog supply voltage using the TRM register (Table 291). The default setting assumes $V_{DDA} \geq 2.7$ V.

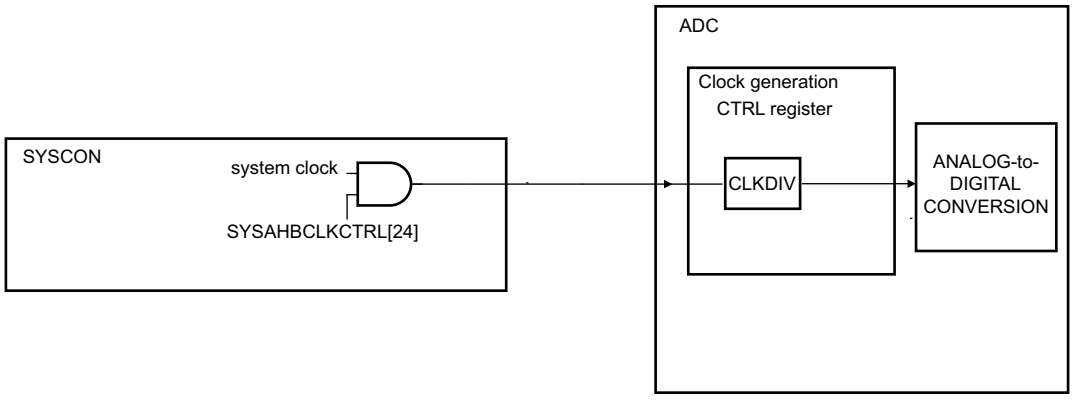


Fig 52. ADC clocking

21.3.1 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 21.7.2.1 “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)/(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 25 MHz and select a CLKDIV value of 0 for a sampling rate of 1 Msamples/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.

21.3.2 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 21.3.3.

To perform a single-step conversion on the first four channels of ADC0 triggered by a rising edge on ADC_PINTRIG0 pin, follow these steps:

1. Enable the analog functions ADC_0 to ADC_3 through the switch matrix. See Table 278.
2. Configure the system clock to be 25 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. Select ADC_PINTRIG0 by writing 0x1 to the TRIGGER bits in the SEQA_CTRL register.
5. Assign the ADC_PINTRIG0 function to pin PIO0_15 through the switch matrix register PINASSIGN10. See Table 278.
6. To generate one interrupt at the end of the entire sequence, set the MODE bit to 1 in the SEQA_CTRL register.
7. Select single-step mode by setting the SINGLESTEP bit in the SEQA_CTRL register to 1.
8. 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.
9. Read the RESULT bits in the DAT0 to DAT3 registers for the conversion result.

21.3.3 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 “A/D Conversion Sequence A Control Register (SEQA_CTRL, address 0x4001 C008) bit description”
- Table “A/D Conversion Sequence B Control Register (SEQB_CTRL, address 0x4001 C00C) bit description”

Table 276. ADC hardware trigger inputs

Input #	Source	Description
0	-	This source is always a logic HIGH. Use this trigger source when running the ADC without hardware triggers or when using a software trigger to avoid spurious conversion triggers.
1	ADC_PINTRG0	ADC pin trigger 0. Connect to an external pin through the switch matrix.
2	ADC_PINTRIG1	ADC pin trigger 1. Connect to an external pin through the switch matrix.
3	SCT0_OUT3	SCT output 3.
4	ACMP_O	Analog comparator output.
5	ARM_TXEV	ARM core TXEV event.

21.3.4 Hardware self-calibration

The A/D converter includes a built-in, hardware self-calibration mode. In order to achieve the specified ADC accuracy, the A/D converter must be recalibrated, at a minimum, following every chip reset before initiating normal ADC operation.

The calibration voltage level is VREFP - VREFN.

To calibrate the ADC follow these steps:

1. Save the current contents of the ADC CTRL register if different from default.
2. In a single write to the ADC CTRL register, do the following to start the calibration:
   - Set the calibration mode bit CALMODE.
   - Write a divider value to the CLKDIV bit field that divides the system clock to yield an ADC clock of about 500 kHz.
   - Clear the LPWR bit.
3. Poll the CALMODE bit until it is cleared.

Before launching a new A/D conversion, restore the contents of the CTRL register or use the default values.

A calibration cycle requires approximately 290 µs to complete. While calibration is in progress, normal ADC conversions cannot be launched, and the ADC Control Register must not be written to. The calibration procedure does not use the CPU or memory, so other processes can be executed during calibration.

21.4 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 in the IOCON register.

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 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 and VREFN at the same voltage levels as V_{DD} and V_{SS}. When selecting VREFP and VREFN different from VDD and VSS, ensure that the voltage midpoints are the same:

$$(VREFP-VREFN)/2 + VREFN = V_{DD}/2$$

Table 277. 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}	Negative voltage reference.
$V_{DDA} = V_{DD}$	The analog supply voltage is internally connected to V_{DD}.
$V_{SSA} = V_{SS}$	ADC ground is internally connected to VSS.

Table 278. ADC pin description

Function	Direction	Type	Connect to	Use register	Reference	Description
ADC_0	AI	external to pin	PIO0_7	PINENABLE0	Table 79	Analog input channel 0.
ADC_1	AI	external to pin	PIO0_6	PINENABLE0	Table 79	Analog input channel 1.
ADC_2	AI	external to pin	PIO0_14	PINENABLE0	Table 79	Analog input channel 2.
ADC_3	AI	external to pin	PIO0_23	PINENABLE0	Table 79	Analog input channel 3.
ADC_4	AI	external to pin	PIO0_22	PINENABLE0	Table 79	Analog input channel 4.
ADC_5	AI	external to pin	PIO0_21	PINENABLE0	Table 79	Analog input channel 5.
ADC_6	AI	external to pin	PIO0_20	PINENABLE0	Table 79	Analog input channel 6.
ADC_7	AI	external to pin	PIO0_19	PINENABLE0	Table 79	Analog input channel 7.
ADC_8	AI	external to pin	PIO0_18	PINENABLE0	Table 79	Analog input channel 8.
ADC_9	AI	external to pin	PIO0_17	PINENABLE0	Table 79	Analog input channel 9.
ADC_10	AI	external to pin	PIO0_13	PINENABLE0	Table 79	Analog input channel 10.
ADC_11	AI	external to pin	PIO0_4	PINENABLE0	Table 79	Analog input channel 11.
ADC_PINTRIG0	I	external to pin	any GPIO	PINASSIGN10	Table 77	ADC pin trigger 0.
ADC_PINTRIG1	I	external to pin	any GPIO	PINASSIGN11	Table 78	ADC pin trigger 1.

21.4.1 ADC vs. 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.

21.5 General description

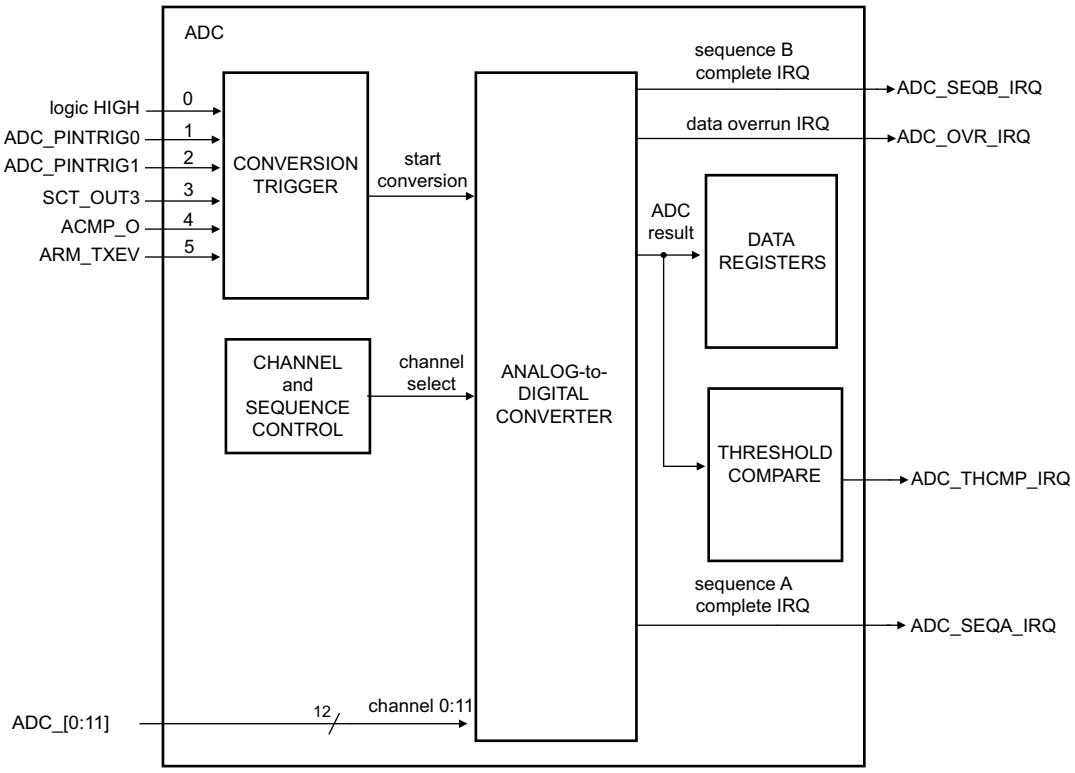


Fig 53. ADC block diagram

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 eight 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 30 MHz. The ADC clock drives the successive approximation process.

A fully accurate conversion requires 25 of these ADC clocks.

21.6 Register description

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

Table 279. 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 280
-	-	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
SEQB_CTRL	R/W	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
SEQA_GDAT	R/W	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 281
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 282
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 283
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 283
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 283
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 283
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 283
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 283
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 283
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 283
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 283
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 283
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 283
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 283
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 284
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 285

Table 279. Register overview : ADC (base address 0x4001 C000)

Name	Access	Address offset	Description	Reset value	Reference
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 286
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 287
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 288
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 289
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 290
TRM	R/W	0x06C	ADC trim register.	0x0000 0F00	Table 291

21.6.1 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 280. 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 30 MHz for 1.2 Msamples/s. 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 280. A/D Control Register (CTRL, addresses 0x4001 C000) bit description

Bit	Symbol	Value	Description	Reset value
10	LPWRMODE		Select low-power ADC mode. 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. Using this mode can save an appreciable amount of current when conversions are required relatively infrequently. 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. **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. 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
30	CALMODE		Writing a 1 to this bit initiates a self-calibration cycle. This bit will be automatically cleared by hardware after the calibration cycle is complete. To calibrate the ADC, set the ADC clock to 500 kHz. **Remark:** Other bits of this register may be written to concurrently with setting this bit, however once this bit has been set no further writes to this register are permitted until the full calibration cycle has ended.	0
31	-		Reserved.	0

21.6.2 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 21.3.1 “Perform a single ADC conversion using a software trigger”.

Remark: Set the BURST and SEQU_ENA bits at the same time.

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. 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. **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. **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. **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: 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. 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. **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

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. 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. 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. 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. 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. Impacts when conversion-complete interrupt/DMA 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/DMA 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. 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/DMA flag will be set when the entire set of sequence-A conversions completes. This flag will need to be explicitly cleared by software or by the DMA-clear signal in this mode. The OVERRUN bit in the SEQA_GDAT register will NOT contribute to generation of an overrun interrupt/DMA trigger since it is assumed this register may not be utilized in this mode.	

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.	

21.6.3 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 21.3.1 “Perform a single ADC conversion using a software trigger”.

Remark: Set the BURST and SEQU_ENA bits at the same time.

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. 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. **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. **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. **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: 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. 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.	

A/D Conversion Sequence B Control Register (SEQB_CTRL, address 0x4001 C00C) bit description

Bit	Symbol	Value	Description	Reset value
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. **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. 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. 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. Impacts when conversion-complete interrupt/DMA trigger for sequence-B will be generated and which overrun conditions contribute to an overrun interrupt as described below:	0
		0	End of conversion. The sequence B interrupt/DMA 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. 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/DMA flag will be set when the entire set of sequence B conversions completes. This flag will need to be explicitly cleared by software or by the DMA-clear signal in this mode. 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.	

A/D Conversion Sequence B Control Register (SEQB_CTRL, address 0x4001 C00C) bit description

Bit	Symbol	Value	Description	Reset value
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.	

21.6.4 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.

The global registers are useful in conjunction with DMA operation - particularly when the channels selected for conversion are not sequential (hence the addresses of the individual result registers will not be sequential, making it difficult for the DMA engine to address them). For interrupt-driven code it will more likely be advantageous to wait for an entire sequence to complete and then retrieve the results from the individual channel registers.

Remark: The method to be employed for each sequence should be reflected in the MODE bit in the corresponding ADSEQn_CTRL register since this will impact interrupt and overrun flag generation.

Table 281. 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. 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}. 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. 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. 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 282. 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. 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}. 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). Threshold Range Comparison result. 0x0 = In Range: The last completed conversion was greater than or equal to the value programmed into the designated LOW threshold register (THRn_LOW) but less than or equal to the value programmed into the designated HIGH threshold register (THRn_HIGH). 0x1 = Below Range: The last completed conversion on was less than the value programmed into the designated LOW threshold register (THRn_LOW). 0x2 = Above Range: The last completed conversion was greater than the value programmed into the designated HIGH threshold register (THRn_HIGH). 0x3 = Reserved.	
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. 0x0 = No threshold Crossing detected: The most recent completed conversion on this channel had the same relationship (above or below) to the threshold value established by the designated LOW threshold register (THRn_LOW) as did the previous conversion on this channel. 0x1 = Reserved. 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. 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 282. 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. 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. 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

21.6.5 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 283. 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. 0x0 = In Range: The last completed conversion was greater than or equal to the value programmed into the designated LOW threshold register (THRn_LOW) but less than or equal to the value programmed into the designated HIGH threshold register (THRn_HIGH). 0x1 = Below Range: The last completed conversion on was less than the value programmed into the designated LOW threshold register (THRn_LOW). 0x2 = Above Range: The last completed conversion was greater than the value programmed into the designated HIGH threshold register (THRn_HIGH). 0x3 = Reserved.	NA
19:18	THCMPCROSS	Threshold Crossing Comparison result. 0x0 = No threshold Crossing detected: The most recent completed conversion on this channel had the same relationship (above or below) to the threshold value established by the designated LOW threshold register (THRn_LOW) as did the previous conversion on this channel. 0x1 = Reserved. 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. 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 283. 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. 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. This bit (in any of the 12 registers) will cause an overrun interrupt request to be asserted if the overrun interrupt is enabled. **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. 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. **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

21.6.6 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 THCMP_RANGE 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 THCMP_CROSS status bits will indicate that a threshold crossing has occurred and the LSB will indicate the direction of the crossing. A threshold crossing event will also generate an interrupt request if enabled to do so via the ADCMPINTEN bits associated with each channel in the INTEN register.

Table 284. 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 285. 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

21.6.7 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 286. 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 287. 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

21.6.8 A/D Channel Threshold Select register

For each channel, this register indicates which pair of threshold registers conversion results should be compared to.

Table 288. 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 288. 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

21.6.9 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. The two conversion/sequence-complete interrupts can also serve as DMA triggers.

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 289. 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/DMA trigger is disabled.	
		1	Enabled. The sequence A interrupt/DMA trigger is enabled and will be asserted either upon completion of each individual conversion performed as part of sequence A, or upon completion of the entire A sequence of conversions, depending on the MODE bit in the SEQA_CTRL register.	
1	SEQB_INTEN		Sequence B interrupt enable.	0
		0	Disabled. The sequence B interrupt/DMA trigger is disabled.	
		1	Enabled. The sequence B interrupt/DMA trigger is enabled and will be asserted either upon completion of each individual conversion performed as part of sequence B, or upon completion of the entire B sequence of conversions, depending on the MODE bit in the SEQB_CTRL register.	
2	OVR_INTEN		Overrun interrupt enable.	0
		0	Disabled. The overrun interrupt is disabled.	
		1	Enabled. The overrun interrupt is enabled. Detection of an overrun condition on any of the 12 channel data registers will cause an overrun interrupt request. 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.	

Table 289. A/D Interrupt Enable register (INTEN, address 0x4001 C064) bit description

Bit	Symbol	Value	Description	Reset value
8:7	ADCMPINTEN2		Threshold comparison interrupt enable.	00
		0x0	Disabled.	
		0x1	Outside threshold.	
		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	

Table 289. A/D Interrupt Enable register (INTEN, address 0x4001 C064) bit description

Bit	Symbol	Value	Description	Reset value
24:23	ADCMPINTEN10		Threshold comparison interrupt enable.	00
		0x0	Disabled.	
		0x1	Outside threshold.	
		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

21.6.10 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).

Remark: The SEQn_INT conversion/sequence-complete flags also serve as DMA triggers.

Table 290. 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

Table 290. A/D Flags register (FLAGS, address 0x4001 C068) bit description

Bit	Symbol	Description	Reset value
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
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/DMA flag. 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. 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. This interrupt must be enabled in the INTEN register.	0

Table 290. A/D Flags register (FLAGS, address 0x4001 C068) bit description

Bit	Symbol	Description	Reset value
29	SEQB_INT	Sequence A interrupt/DMA flag. 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. 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. This interrupt must be enabled in the INTEN register.	0
30	THCMP_INT	Threshold Comparison Interrupt/DMA flag. 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). Each type of threshold comparison interrupt on each channel must be individually enabled in the INTEN register to cause this interrupt. 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. 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. This interrupt must be enabled in the INTEN register. This bit will be cleared when all of the individual overrun bits have been cleared via reading the corresponding data registers.	0

21.6.11 A/D trim register

The A/D trim register configures the ADC for the appropriate operating range of the analog supply voltage VDDA.

Remark: Failure to set the VRANGE bit correctly causes the ADC to return incorrect conversion results.

Table 291. A/D Flags register (TRM, addresses 0x4001 C06C) bit description

Bit	Symbol	Value	Description	Reset value
4:0	-		Reserved.	-
5	VRANGE		Reserved.	0
		0	High voltage. VDD = 2.7 V to 3.6 V.	
		1	Low voltage. VDD = 2.4 V to 2.7 V.	
31:6	-		Reserved.	-

21.7 Functional description

21.7.1 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.

21.7.2 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.

21.7.2.1 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.

21.7.3 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.

21.7.4 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.

21.7.4.1 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 (except when used as a DMA trigger, in which case it will be cleared in hardware by the DMA engine).

21.7.4.2 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.

21.7.4.3 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.

21.7.5 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.

21.7.6 DMA control

The sequence-A or sequence-B conversion/sequence-complete interrupts may also be used to generate a DMA trigger. To trigger a DMA transfer, the same conditions must be met as the conditions for generating an interrupt (see Section 21.7.4 and Section 21.6.9).

Remark: If the DMA is used, the ADC interrupt must be disabled in the NVIC.

For DMA transfers, only burst requests are supported. The burst size can be set to one in the DMA channel control register (see Table 172). If the number of ADC channels is not equal to one of the other DMA-supported burst sizes (applicable DMA burst sizes are 1, 4, 8), set the burst size to one.

The DMA transfer size determines when a DMA interrupt is generated. The transfer size can be set to the number of ADC channels being converted. Non-contiguous channels can be transferred by the DMA using the scatter/gather linked lists.

21.7.7 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.

22.1 How to read this chapter

The analog comparator is available on all LPC82x parts.

Remark: The external voltage ladder reference VDDCMP is not available on the TSSOP20 package.

22.2 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

22.3 Basic configuration

Configure the analog comparator using the following registers:

- In the SYSAHBCLKCTRL register, set bit 19 (Table 35) to enable the clock to the register interface.
- You can enable or disable the power to the analog comparator through the PDRUNCFG register (Table 54).
- Clear the analog comparator peripheral reset using the PRESETCTRL register (Table 23).
- The analog comparator interrupt is connected to interrupt #11 in the NVIC.
- Configure the analog comparator pin functions through the switch matrix. See Section 22.4.

22.3.1 Connect the comparator output to the SCT

The comparator output function (ACMP_O) can be used to start or stop the SCTimer/PWM or, more generally, to create an SCT event without assigning a pin through the switch matrix. To create an SCT event internally connected to the comparator output, select the comparator output as one of the SCT inputs through the INPUT MUX (see Table 147 “SCT input mux registers 0 to 3 (SCT0_INMUX[0:3], address 0x4002 C020 (SCT0_INMUX0) to 0x4002 C02C (SCT0_INMUX3)) bit description”.

22.4 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 7.3.1 “Connect an internal signal to a package pin” to assign the analog comparator output to any pin on the LPC82x package.

Table 292. Analog comparator pin description

Function	Type	Pin	Description	SWM register	Reference
ACMP_I1	I	PIO0_0	Comparator input 1	PINENABLE0	Table 79
ACMP_I2	I	PIO0_1	Comparator input 2	PINENABLE0	Table 79
ACMP_I3	I	PIO0_14	Comparator input 3	PINENABLE0	Table 79
ACMP_I4	I	PIO0_23	Comparator input 4	PINENABLE0	Table 79
ACMP_O	O	any	Comparator output	PINASSIGN8	Table 78
VDDCMP	I	PIO0_6	External reference voltage source for 32-stage Voltage Ladder.	PINENABLE0	Table 79

22.5 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 54 and Table 294).

Input 0 of the multiplexer is the programmable voltage ladder output.

Inputs 1 to 4 connect the external inputs ACMP_I[4:1].

Input 5 of the multiplexer connects the internal reference voltage input.

Input 6 of the multiplexer connects the ADC channel 0 input.

Input 7 is tied to ground.

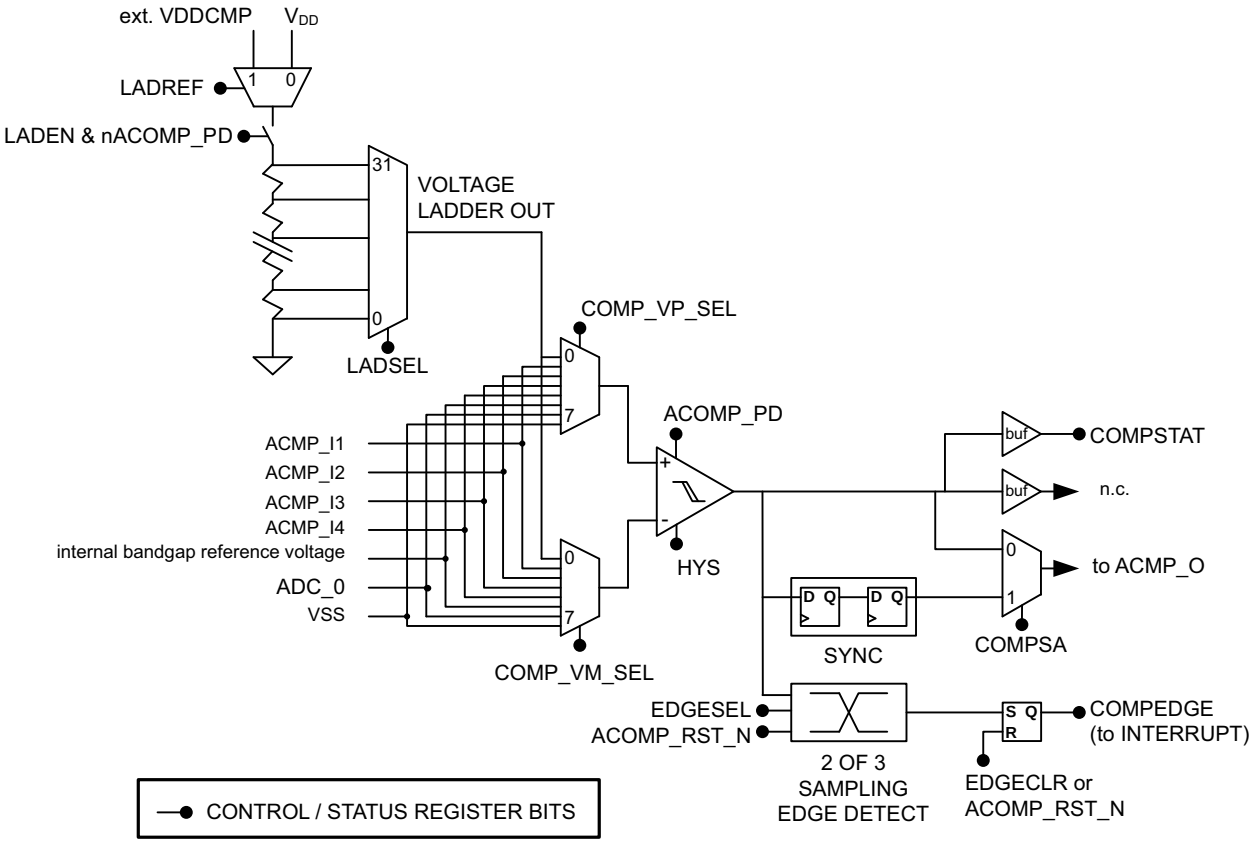


Fig 54. Comparator block diagram

22.5.1 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.

22.5.2 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.

22.5.3 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.

22.5.4 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 35 “System clock control register (SYSAHBCLKCTRL, 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.

The comparator output can be routed to the SCT via the switch matrix allowing to capture the time of a voltage crossing or to count crossings in either or both directions. See Section 22.3.1 “Connect the comparator output to the SCT”.

22.6 Register description

Table 293. 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 294
LAD	R/W	0x004	Voltage ladder register	0	Table 295

22.6.1 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 294 are reserved and should be written as 0.

Table 294. 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 294. 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	Band gap. Internal reference voltage.	
		0x6	ADC_0. ADC channel 0 input.	
		0x7	Reserved	
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	Band gap. Internal reference voltage.	
		0x6	ADC_0. ADC channel 0 input.	
		0x7	Reserved	
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	-		Reserved. Write as 0.	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	-

22.6.2 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 295. 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. 00000 = V_{SS} 00001 = 1 ´ Vref/31 00010 = 2 ´ Vref/31 ... 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

23.1 How to read this chapter

The CRC engine is available on all LPC82x parts.

23.2 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)

23.3 Basic configuration

Enable the clock to the CRC engine in the SYSAHBCLKCTRL register (Table 35, bit 13).

23.4 Pin description

The CRC engine has no configurable pins.

23.5 General description

The Cyclic Redundancy Check (CRC) generator with programmable polynomial settings supports several CRC standards commonly used.

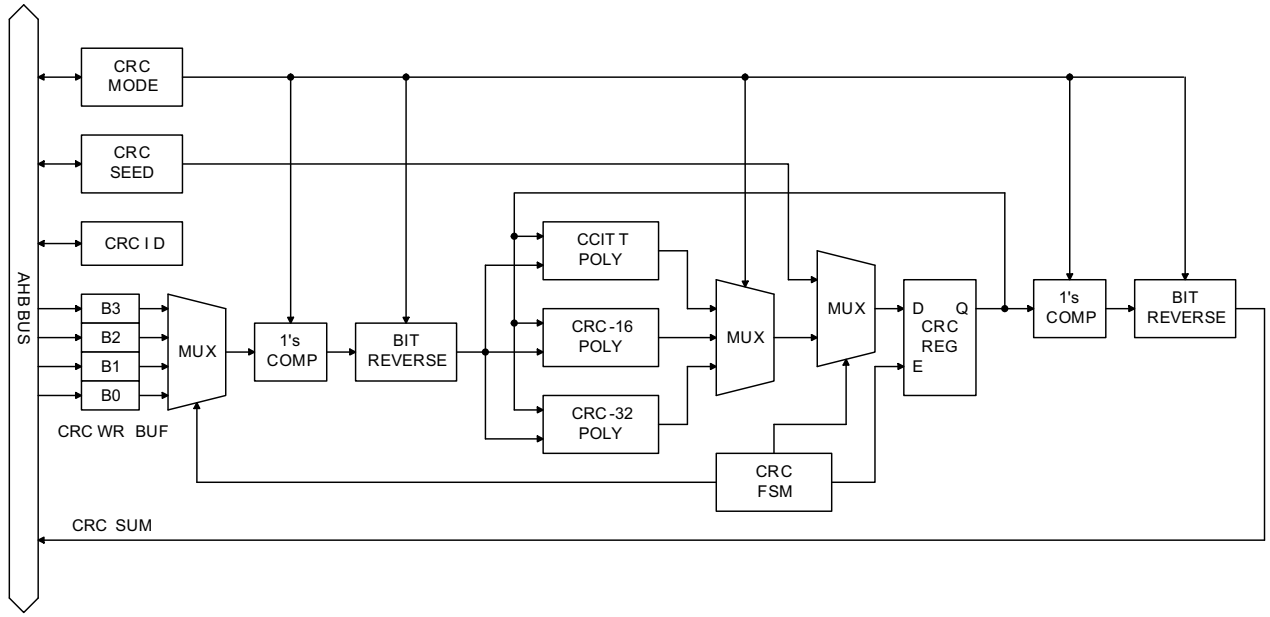


Fig 55. CRC block diagram

23.6 Register description

Table 296. 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 297
SEED	R/W	0x004	CRC seed register	0x0000 FFFF	Table 298
SUM	RO	0x008	CRC checksum register	0x0000 FFFF	Table 299
WR_DATA	WO	0x008	CRC data register	-	Table 300

23.6.1 CRC mode register

Table 297. CRC mode register (MODE, address 0x5000 0000) bit description

Bit	Symbol	Description	Reset value
1:0	CRC_POLY	CRC polynom: 1X= CRC-32 polynomial 01= CRC-16 polynomial 00= CRC-CCITT polynomial	00
2	BIT_RVS_WR	Data bit order: 1= Bit order reverse for CRC_WR_DATA (per byte) 0= No bit order reverse for CRC_WR_DATA (per byte)	0
3	CMPL_WR	Data complement: 1= 1's complement for CRC_WR_DATA 0= No 1's complement for CRC_WR_DATA	0
4	BIT_RVS_SUM	CRC sum bit order: 1= Bit order reverse for CRC_SUM 0= No bit order reverse for CRC_SUM	0
5	CMPL_SUM	CRC sum complement: 1= 1's complement for CRC_SUM 0=No 1's complement for CRC_SUM	0
31:6	Reserved	Always 0 when read	0x0000000

23.6.2 CRC seed register

Table 298. 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. **Remark:** A write access to this register will overrule the CRC calculation in progresses.	0x0000 FFFF

23.6.3 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 299. 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

23.6.4 CRC data register

This register is a Write-only register containing the data block for which the CRC sum will be calculated.

Table 300. 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.	-

23.7 Functional description

The following sections describe the register settings for each supported CRC standard:

23.7.1 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

23.7.2 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

23.7.3 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

24.1 How to read this chapter

The flash controller is identical on all LPC82x parts.

24.2 Features

- Controls flash access time.
- Provides registers for flash signature generation.

24.3 General description

The flash controller is accessible for programming flash wait states and for generating the flash signature.

24.4 Register description

Table 301. Register overview: FMC (base address 0x4004 0000)

Name	Access	Address offset	Description	Reset value	Reference
FLASHCFG	R/W	0x010	Flash configuration register	-	Table 302
FMSSTART	R/W	0x020	Signature start address register	0	Table 303
FMSSTOP	R/W	0x024	Signature stop-address register	0	Table 304
FMSW0	R	0x02C	Signature word	-	Table 305

24.4.1 Flash configuration register

Access time to the flash memory can be configured independently of the system frequency by writing to the FLASHCFG register.

Remark: When using the Power API, do not change the waitstates when in efficiency, low-current, or performance modes.

Table 302. Flash configuration register (FLASHCFG, address 0x4004 0010) bit description

Bit	Symbol	Value	Description	Reset value
1:0	FLASHTIM		Flash memory access time. FLASHTIM +1 is equal to the number of system clocks used for flash access.	0x1
		0x0	1 system clock flash access time.	
		0x1	2 system clocks flash access time.	
		0x2	Reserved.	
		0x3	Reserved.	
31:2	-	-	Reserved. **User software must not change the value of these bits. Bits 31:2 must be written back exactly as read**.	-

24.4.2 Flash signature start address register

Table 303. Flash Module Signature Start register (FMSSTART, 0x4004 0020) bit description

Bit	Symbol	Description	Reset value
16:0	START	Signature generation start address (corresponds to AHB byte address bits[20:4]).	0
31:17	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

24.4.3 Flash signature stop address register

Table 304. Flash Module Signature Stop register (FMSSTOP, 0x4004 0024) bit description

Bit	Symbol	Value	Description	Reset value
16:0	STOPA		Stop address for signature generation (the word specified by STOPA is included in the address range). The address is in units of memory words, not bytes.	0
30:17	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
31	STRTBIST		When this bit is written to 1, signature generation starts. At the end of signature generation, this bit is automatically cleared.	0

24.4.4 Flash signature generation result register

The signature generation result register returns the flash signature produced by the embedded signature generator.

The generated flash signature can be used to verify the flash memory contents. The generated signature can be compared with an expected signature and thus makes saves time and code space. The method for generating the signature is described in Section 24.5.1.

Table 305. FMSW0 register bit description (FMSW0, address: 0x4004 002C)

Bit	Symbol	Description	Reset value
31:0	SIG	32-bit signature.	-

24.5 Functional description

24.5.1 Flash signature generation

The flash module contains a built-in signature generator. This generator can produce a 32-bit signature from a range of flash memory. A typical usage is to verify the flashed contents against a calculated signature (e.g. during programming).

The address range for generating a signature must be aligned on flash-word boundaries, i.e. 32-bit boundaries. Once started, signature generation completes independently. While signature generation is in progress, the flash memory cannot be accessed for other purposes, and an attempted read will cause a wait state to be asserted until signature

generation is complete. Code outside of the flash (e.g. internal RAM) can be executed during signature generation. This can include interrupt services, if the interrupt vector table is re-mapped to memory other than the flash memory. The code that initiates signature generation should also be placed outside of the flash memory.

24.5.1.1 Signature generation address and control registers

These registers control automatic signature generation. A signature can be generated for any part of the flash memory contents. The address range to be used for generation is defined by writing the start address to the signature start address register (FMSSTART) and the stop address to the signature stop address register (FMSSTOP. The start and stop addresses must be aligned to 32-bit boundaries.

Signature generation is started by setting the STRTBIST bit in the FMSSTOP register. Setting the STRTBIST bit is typically combined with the signature stop address in a single write.

Table 303 and Table 304 show the bit assignments in the FMSSTART and FMSSTOP registers respectively.

24.5.1.2 Signature generation

A signature can be generated for any part of the flash contents. The address range to be used for signature generation is defined by writing the start address to the FMSSTART register, and the stop address to the FMSSTOP register.

The signature generation is started by writing a 1 to the SIG_START bit in the FMSSTOP register. Starting the signature generation is typically combined with defining the stop address, which is done in the STOP bits of the same register.

The time that the signature generation takes is proportional to the address range for which the signature is generated. Reading of the flash memory for signature generation uses a self-timed read mechanism and does not depend on any configurable timing settings for the flash. A safe estimation for the duration of the signature generation is:

Duration = int((60 / tcy) + 3) x (FMSSTOP - FMSSTART + 1)

When signature generation is triggered via software, the duration is in AHB clock cycles, and tcy is the time in ns for one AHB clock.

After signature generation, a 32-bit signature can be read from the FMSW0 register. The 32-bit signature reflects the corrected data read from the flash and the flash parity bits and check bit values.

24.5.1.3 Content verification

The signature as it is read from the FMSW0 register must be equal to the reference signature. The following pseudo-code shows the algorithm to derive the reference signature:

```
sign = 0
FOR address = FMSSTART.START to FMSSTOP.STOPA
{
      FOR i = 0 TO 30
      {
            nextSign[i] = f_Q[address][i] XOR sign[i + 1]
```

```
        }
        nextSign[31] = f_Q[address][31] XOR sign[0] XOR sign[10] XOR sign[30] XOR
        sign[31]
        sign = nextSign
    }
    signature32 = sign
```

25.1 How to read this chapter

See Table 306 for different flash configurations.

25.2 Features

- In-System Programming: In-System programming (ISP) is programming or reprogramming the on-chip flash memory, using the bootloader software and UART serial port.
- 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.
- You can use ISP and IAP when the part resides in the end-user board.
- Flash page write and erase supported.

25.3 Basic configuration

To use the IAP calls, enable the IRC and the IRC output in the PDRUNCFG register (see Table 54 “Power configuration register (PDRUNCFG, address 0x4004 8238) bit description”).

25.4 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. In ISP mode, pin PIO0_0 is connected to function U0_RXD and pin PIO0_4 is connected to function U0_TXD on the USART0 block.

25.5 General description

25.5.1 Flash configuration

Most IAP and ISP commands operate on sectors and specify sector numbers. In addition a page erase command is supported. The following table shows the correspondence between page numbers, sector numbers, and memory addresses.

The size of a sector is 1 KB and the size of a page is 64 Byte. One sector contains 16 pages.

Table 306. LPC82x flash configuration

Sector number	Sector size [KB]	Page number	Address range	16 KB flash	32 KB flash
0	1	0 -15	0x0000 0000 - 0x0000 03FF	yes	yes
1	1	16 - 31	0x0000 0400 - 0x0000 07FF	yes	yes
2	1	32 - 47	0x0000 0800 - 0x0000 0BFF	yes	yes
3	1	48 - 63	0x0000 0C00 - 0x0000 0FFF	yes	yes
4	1	64 - 79	0x0000 1000 - 0x0000 13FF	yes	yes
5	1	80 - 95	0x0000 1400 - 0x0000 17FF	yes	yes
6	1	96 - 111	0x0000 1800 - 0x0000 1BFF	yes	yes
7	1	112 - 127	0x0000 1C00 - 0x0000 1FFF	yes	yes
8	1	128 - 143	0x0000 2000 - 0x0000 23FF	yes	yes
9	1	144 - 159	0x0000 2400 - 0x0000 27FF	yes	yes
10	1	160 - 175	0x0000 2800 - 0x0000 2BFF	yes	yes
11	1	176 - 191	0x0000 2C00 - 0x0000 2FFF	yes	yes
12	1	192 - 207	0x0000 3000 - 0x0000 33FF	yes	yes
13	1	208 - 223	0x0000 3400 - 0x0000 37FF	yes	yes
14	1	224 - 239	0x0000 3800 - 0x0000 3BFF	yes	yes
15	1	240 - 255	0x0000 3C00 - 0x0000 3FFF	yes	yes
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 - 349	0x0000 5400 - 0x0000 57FF	-	yes
22	1	350 - 365	0x0000 5800 - 0x0000 5BFF	-	yes
23	1	366 - 381	0x0000 5C00 - 0x0000 5FFF	-	yes
24	1	382 - 397	0x0000 6000 - 0x0000 63FF	-	yes
25	1	398 - 403	0x0000 6400 - 0x0000 67FF	-	yes
26	1	404 - 419	0x0000 6800 - 0x0000 6BFF	-	yes
27	1	420 - 435	0x0000 6C00 - 0x0000 6FFF	-	yes
28	1	436 - 451	0x0000 7000 - 0x0000 73FF	-	yes
29	1	452 - 467	0x0000 7400 - 0x0000 77FF	-	yes
30	1	468 - 483	0x0000 7800 - 0x0000 7BFF	-	yes
31	1	484 - 499	0x0000 7C00 - 0x0000 7FFF	-	yes

25.5.2 Flash content protection mechanism

The part is equipped with the Error Correction Code (ECC) capable Flash memory. The purpose of an error correction module is twofold:

The ECC first decodes data words read from the memory into output data words. Then, the ECC 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 the user accessible 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 accessible Flash is made, writing the 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 bits are also erased. Once a 6-bit ECC 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.

25.5.3 Code Read Protection (CRP)

Code Read Protection is a mechanism that allows the user to enable different levels of security in the system so that access to the on-chip flash and use of the ISP can be restricted. When needed, CRP is invoked by programming a specific pattern in flash location at 0x0000 02FC. IAP commands are not affected by the code read protection.

Important: any CRP change becomes effective only after the device has gone through a power cycle.

Table 307. Code Read Protection options

Name	Pattern programmed in 0x0000 02FC	Description
NO_ISP	0x4E69 7370	Prevents sampling of the ISP entry pin for entering ISP mode. The ISP entry pin is available for other uses.
CRP1	0x12345678	Access to chip via the SWD pins is disabled. This mode allows partial flash update using the following ISP commands and restrictions: • Write to RAM command should not access RAM below 0x1000 0300. Access to addresses below 0x1000 0200 is disabled. • Copy RAM to flash command can not write to Sector 0. • Erase command can erase Sector 0 only when all sectors are selected for erase. • Compare command is disabled. • Read Memory command is disabled. 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	0x87654321	Access to chip via the SWD pins is disabled. The following ISP commands are disabled: • Read Memory • Write to RAM • Go • Copy RAM to flash • Compare When CRP2 is enabled the ISP erase command only allows erasure of all user sectors.
CRP3	0x43218765	Access to chip via the SWD pins is disabled. ISP entry by pulling the ISP entry pin LOW is disabled if a valid user code is present in flash sector 0. This mode effectively disables ISP override using the ISP entry pin. It is up to the user's application to provide a flash update mechanism using IAP calls or call re-invoke ISP command to enable flash update via UART. **Caution: If CRP3 is selected, no future factory testing can be performed on the device.**

Table 308. Code Read Protection hardware/software interaction

CRP option	User Code Valid	ISP entry pin at reset	SWD enabled	Part enters ISP mode	partial flash update in ISP mode
None	No	x	Yes	Yes	Yes
None	Yes	High	Yes	No	NA
None	Yes	Low	Yes	Yes	Yes
CRP1	Yes	High	No	No	NA
CRP1	Yes	Low	No	Yes	Yes
CRP2	Yes	High	No	No	NA
CRP2	Yes	Low	No	Yes	No
CRP3	Yes	x	No	No	NA
CRP1	No	x	No	Yes	Yes
CRP2	No	x	No	Yes	No
CRP3	No	x	No	Yes	No

Table 309. 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 0300 only	no	n/a
Read Memory	no	no	n/a
Prepare sector(s) 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
Blank check sector(s)	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

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.

25.5.3.1 ISP entry protection

In addition to the three CRP modes, the user can prevent the sampling of the ISP entry pin for entering ISP mode and thereby release the ISP entry pin for other uses. This is called the NO_ISP mode. The NO_ISP mode can be entered by programming the pattern 0x4E69 7370 at location 0x0000 02FC.

25.6 API description

25.6.1 UART 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 310. UART ISP command summary

ISP Command	Usage	Described in
Unlock	U <Unlock Code>	Table 311
Set Baud Rate	B <Baud Rate> <stop bit>	Table 312
Echo	A <setting>	Table 313
Write to RAM	W <start address> <number of bytes>	Table 314
Read Memory	R <address> <number of bytes>	Table 315
Prepare sector(s) for write operation	P <start sector number> <end sector number>	Table 316
Copy RAM to flash	C <Flash address> <RAM address> <number of bytes>	Table 317
Go	G <address> <Mode>	Table 318
Erase sector(s)	E <start sector number> <end sector number>	Table 319
Blank check sector(s)	I <start sector number> <end sector number>	Table 320
Read Part ID	J	Table 321
Read Boot code version	K	Table 323
Compare	M <address1> <address2> <number of bytes>	Table 324
ReadUID	N	Table 325
Read CRC checksum	S <address> <number of bytes>	Table 326

25.6.1.1 Unlock <Unlock code>

Table 311. UART ISP Unlock command

Command	U
Input	Unlock code: 23130 (decimal)
Return Code	CMD_SUCCESS | INVALID_CODE | 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.

25.6.1.2 Set Baud Rate <Baud Rate> <stop bit>

Table 312. UART ISP Set Baud Rate command

Command	B
Input	Baud Rate: 9600 | 19200 | 38400 | 57600 | 115200 Stop bit: 1 | 2
Return Code	CMD_SUCCESS | INVALID_BAUD_RATE | INVALID_STOP_BIT | PARAM_ERROR
Description	This command is used to change the baud rate. The new baud rate is effective after the command handler sends the CMD_SUCCESS return code.
Example	"B 57600 1<CR><LF>" sets the serial port to baud rate 57600 bps and 1 stop bit.

25.6.1.3 Echo <setting>

Table 313. UART ISP Echo command

Command	A
Input	Setting: ON = 1 | OFF = 0
Return Code	CMD_SUCCESS | PARAM_ERROR
Description	The default setting for echo command is ON. When ON the ISP command handler sends the received serial data back to the host.
Example	"A 0<CR><LF>" turns echo off.

25.6.1.4 Write to RAM <start address> <number of bytes>

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 314. UART ISP Write to RAM command

Command	W
Input	**Start Address:** RAM address where data bytes are to be written. This address should be a word boundary. **Number of Bytes:** Number of bytes to be written. Count should be a multiple of 4
Return Code	CMD_SUCCESS | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to download data to RAM. This command is blocked when code read protection levels 2 or 3 are enabled. Writing to addresses below 0x1000 0300 is disabled for CRP1.
Example	"W 268436224 4<CR><LF>" writes 4 bytes of data to address 0x1000 0300.

25.6.1.5 Read Memory <address> <number of bytes>

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

Table 315. UART ISP Read Memory command

Command	R
Input	**Start Address:** Address from where data bytes are to be read. This address should be a word boundary. **Number of Bytes:** Number of bytes to be read. Count should be a multiple of 4.
Return Code	CMD_SUCCESS followed by <actual data (plain binary)> | ADDR_ERROR (Address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not a multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to read data from RAM or flash memory. This command is blocked when code read protection is enabled.
Example	"R 268435456 4<CR><LF>" reads 4 bytes of data from address 0x1000 0000.

25.6.1.6 Prepare sector(s) for write operation <start sector number> <end sector number>

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

Table 316. UART ISP Prepare sector(s) for write operation command

Command	P
Input	**Start Sector Number** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR | PARAM_ERROR
Description	This command must be executed before executing "Copy RAM to flash" or "Erase Sector(s)" command. Successful execution of the "Copy RAM to flash" or "Erase Sector(s)" command causes relevant sectors to be protected again. The boot block can not be prepared by this command. To prepare a single sector use the same "Start" and "End" sector numbers.
Example	"P 0 0<CR><LF>" prepares the flash sector 0.

25.6.1.7 Copy RAM to flash <Flash address> <RAM address> <no of bytes>

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 follows from the application of ECC to the flash write operation, see Section 25.5.2.
3. To avoid write disturbance (a mechanism intrinsic to flash memories), an erase should be performed after following 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 317. UART ISP Copy RAM to flash 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. **RAM Address (SRC):** Source RAM address from where data bytes are to be read. **Number of Bytes:** Number of bytes to be written. Should be 64 | 128 | 256 | 512 | 1024.
Return Code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not on word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 64 | 128 | 256 | 512 | 1024) | SECTOR_NOT_PREPARED_FOR WRITE_OPERATION | BUSY | CMD_LOCKED | PARAM_ERROR | 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. The boot block cannot be written by this command. This command is blocked when code read protection is enabled.
Example	"C 0 268437504 512<CR><LF>" copies 512 bytes from the RAM address 0x1000 0800 to the flash address 0.

25.6.1.8 Go <address> <mode>

Table 318. UART ISP Go command

Command	G
Input	**Address:** Flash or RAM address from which the code execution is to be started. This address should be on a word boundary. **Mode:** T (Execute program in Thumb Mode).
Return Code	CMD_SUCCESS | ADDR_ERROR | ADDR_NOT_MAPPED | CMD_LOCKED | PARAM_ERROR | 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. The command must be used with an address of 0x0000 0200 or greater.
Example	"G 512 T<CR><LF>" branches to address 0x0000 0200 in Thumb mode.

The GO command is usually used after the flash image has been updated and a RESET is desired. For this, the GO command should point to the RESET handler. Since the device is still in ISP, the RESET handler should do the following:

- Re-initialize the SP pointer to the application default
- Set the SYSMEMREMAP to either 0x01 or 0x02

While in the ISP mode, the SYSMEMREMAP is set to 0x00.

Alternatively, the following snippet can be loaded into the RAM for execution:

```
SCB->AIRCR = 0x05FA0004;                          //issue system reset
while(1);                                         //should never come here
```

The snippet will issue a system reset request to the core.

25.6.1.9 Erase sector(s) <start sector number> <end sector number>

Table 319. UART ISP Erase sector command

Command	E
Input	**Start Sector Number** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | BUSY | INVALID_SECTOR | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | CMD_LOCKED | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to erase one or more sector(s) of on-chip flash memory. The boot block can not be erased using this command. 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.

25.6.1.10 Blank check sector(s) <sector number> <end sector number>

Table 320. UART ISP Blank check sector command

Command	I
Input	**Start Sector Number:** **End Sector Number:** Should be greater than or equal to start sector number.
Return Code	CMD_SUCCESS | SECTOR_NOT_BLANK (followed by <Offset of the first non blank word location> <Contents of non blank word location>) | INVALID_SECTOR | PARAM_ERROR
Description	This command is used to blank check one or more sectors of on-chip flash memory. Blank check on sector 0 always fails as first 64 bytes are re-mapped to flash boot block. When CRP is enabled, the blank check command returns 0 for the offset and value of sectors which are not blank. Blank sectors are correctly reported irrespective of the CRP setting.
Example	"I 2 3<CR><LF>" blank checks the flash sectors 2 and 3.

25.6.1.11 Read Part Identification number

Table 321. UART ISP Read Part Identification command

Command	J
Input	None.
Return Code	CMD_SUCCESS followed by part identification number in ASCII (see Table 322).
Description	This command is used to read the part identification number.

Table 322. Part identification numbers

Device	Hex coding
LPC824M201JHI33	0x0000 8241
LPC822M101JHI33	0x0000 8221
LPC824M201JDH20	0x0000 8242
LPC822M101JDH20	0x0000 8222

25.6.1.12 Read Boot code version number

Table 323. UART 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.

25.6.1.13 Compare <address1> <address2> <no of bytes>

Table 324. UART ISP Compare command

Command	M
Input	**Address1 (DST):** Starting flash or RAM address of data bytes to be compared. This address should be a word boundary. **Address2 (SRC):** Starting flash or RAM address of data bytes to be compared. This address should be a word boundary. **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) COMPARE_ERROR | (Followed by the offset of first mismatch) COUNT_ERROR (Byte count is not a multiple of 4) | ADDR_ERROR | ADDR_NOT_MAPPED | PARAM_ERROR
Description	This command is used to compare the memory contents at two locations.
Example	"M 8192 268468224 4<CR><LF>" compares 4 bytes from the RAM address 0x1000 8000 to the 4 bytes from the flash address 0x2000.

25.6.1.14 ReadUID

Table 325. UART ISP ReadUID command

Command	N
Input	None
Return Code	CMD_SUCCESS followed by four 32-bit words of E-sort test information in ASCII format. The word sent at the lowest address is sent first.
Description	This command is used to read the unique ID.

25.6.1.15 Read CRC checksum <address> <no of bytes>

Get the CRC checksum of a block of RAM or flash. CMD_SUCCESS followed by 8 bytes of CRC checksum in ASCII 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

No 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

Table 326. UART 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. **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 plain binary format ADDR_ERROR (address not on word boundary) | ADDR_NOT_MAPPED | COUNT_ERROR (byte count is not a multiple of 4) | PARAM_ERROR | CODE_READ_PROTECTION_ENABLED
Description	This command is used to read the CRC checksum of a block of RAM or flash memory. This command is blocked when code read protection is enabled.
Example	"S 268436736 4<CR><LF>" reads the CRC checksum for 4 bytes of data from address 0x1000 0500. If checksum value is 0xCBF43926, then the host will receive: "3421780262 <CR><LF>"

25.6.1.16 UART ISP Return Codes

Table 327. UART ISP Return Codes Summary

Return Code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully. Sent by ISP handler only when command given by the host has been completely and successfully executed.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on word boundary.
3	DST_ADDR_ERROR	Destination address is not on a correct boundary.
4	SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken in to consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken in to consideration where applicable.
6	COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
7	INVALID_SECTOR	Sector number is invalid or end sector number is greater than start sector number.
8	SECTOR_NOT_BLANK	Sector is not blank.
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
10	COMPARE_ERROR	Source and destination data not equal.
11	BUSY	Flash programming hardware interface is busy.
12	PARAM_ERROR	Insufficient number of parameters or invalid parameter.
13	ADDR_ERROR	Address is not on word boundary.
14	ADDR_NOT_MAPPED	Address is not mapped in the memory map. Count value is taken in to consideration where applicable.
15	CMD_LOCKED	Command is locked.
16	INVALID_CODE	Unlock code is invalid.
17	INVALID_BAUD_RATE	Invalid baud rate setting.
18	INVALID_STOP_BIT	Invalid stop bit setting.
19	CODE_READ_PROTECTION_ENABLED	Code read protection enabled.

25.6.2 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. 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 56. 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 4, 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 0x1FFF 1FF0 location and it is thumb code.

To call an IAP function, do the following:

Define the IAP location entry point. Since the 0th bit of the IAP location is set there will be a change to Thumb instruction set when the program counter branches to this address.

```
#define IAP_LOCATION 0x1fff1ff1
```

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[4];
```

or

```
unsigned int * command_param;
unsigned int * status_result;
command_param = (unsigned int *) 0x...
status_result =(unsigned int *) 0x...
```

Define pointer to function type, which takes two parameters and returns void. Note the IAP returns the result with the base address of the table residing in R1.

```
typedef void (*IAP)(unsigned int [],unsigned int[]);
IAP iap_entry;
```

Setting the function pointer:

```
iap_entry=(IAP) IAP_LOCATION;
```

To call the IAP, use the following statement.

```
iap_entry (command_param,status_result);
```

As per the ARM specification (The ARM Thumb Procedure Call Standard SWS ESPC 0002 A-05) up to 4 parameters can be passed in the r0, r1, r2 and r3 registers respectively. Additional parameters are passed on the stack. Up to 4 parameters can be returned in the r0, r1, r2 and r3 registers respectively. Additional parameters are returned indirectly via memory. Some of the IAP calls require more than 4 parameters. If the ARM suggested scheme is used for the parameter passing/returning then it might create problems due to difference in the C compiler implementation from different vendors. The suggested parameter passing scheme reduces such risk.

The flash memory is not accessible during a write or erase operation. IAP commands, which results in a flash write/erase operation, use 32 bytes of space in the top portion of the on-chip RAM for execution. The user program should not be use this space if IAP flash programming is permitted in the application.

Table 328. IAP Command Summary

IAP Command	Command Code	Described in
Prepare sector(s) for write operation	50 (decimal)	Table 329
Copy RAM to flash	51 (decimal)	Table 330
Erase sector(s)	52 (decimal)	Table 331
Blank check sector(s)	53 (decimal)	Table 332
Read Part ID	54 (decimal)	Table 333
Read Boot code version	55 (decimal)	Table 334
Compare	56 (decimal)	Table 335
Reinvoke ISP	57 (decimal)	Table 336
Read UID	58 (decimal)	Table 337
Erase page	59 (decimal)	Table 338

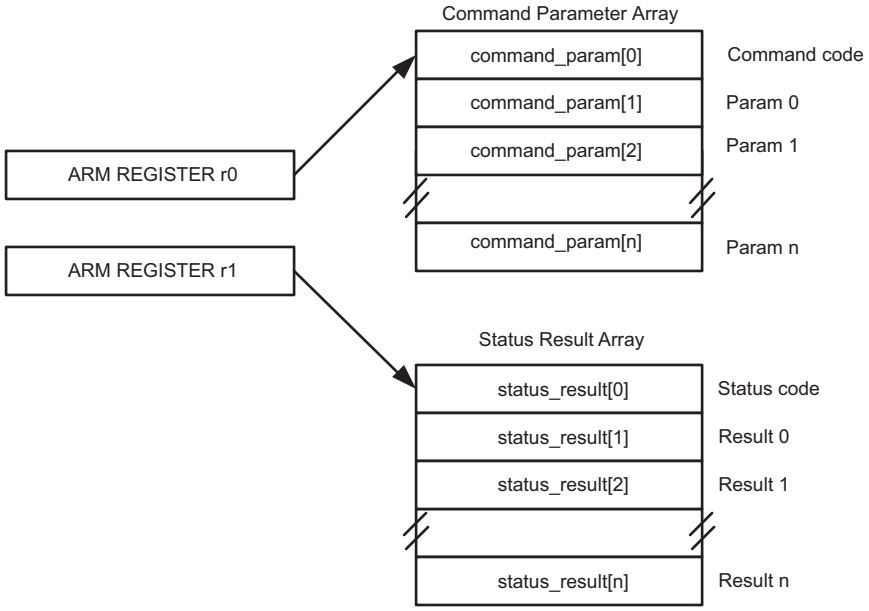


Fig 56. IAP parameter passing

25.6.2.1 Prepare sector(s) for write operation (IAP)

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

Table 329. IAP Prepare sector(s) for write operation command

Command	Prepare sector(s) for write operation
Input	**Command code: 50** **(decimal)** **Param0:** Start Sector Number **Param1:** End Sector Number (should be greater than or equal to start sector number).
Status code	CMD_SUCCESS | BUSY | INVALID_SECTOR
Result	None
Description	This command must be executed before executing "Copy RAM to flash" or "Erase Sector(s)" command. Successful execution of the "Copy RAM to flash" or "Erase Sector(s)" command causes relevant sectors to be protected again. The boot sector can not be prepared by this command. To prepare a single sector use the same "Start" and "End" sector numbers.

25.6.2.2 Copy RAM to flash (IAP)

See Section 25.6.1.4 for limitations on the write-to-flash process.

Table 330. IAP Copy RAM to flash command

Command	Copy RAM to flash
Input	**Command code: 51** **(decimal)** **Param0(DST):** Destination flash address where data bytes are to be written. This address should be a 64 byte boundary. **Param1(SRC):** Source RAM address from which data bytes are to be read. This address should be a word boundary. **Param2:** Number of bytes to be written. Should be 64 | 128 | 256 | 512 | 1024. **Param3:** NULL
Status code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not a word boundary) | DST_ADDR_ERROR (Address not on correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 256 | 512 | 1024 | 4096) | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | 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. The boot sector can not be written by this command. Param3 is overwritten by the fixed value of 12 MHz, which is the IRC reference clock used by the flash controller.

25.6.2.3 Erase Sector(s) (IAP)

Table 331. IAP Erase Sector(s) command

Command	Erase Sector(s)
Input	**Command code: 52** **(decimal)** **Param0:** Start Sector Number **Param1:** End Sector Number (should be greater than or equal to start sector number). **Param2:** NULL
Status code	CMD_SUCCESS | BUSY | SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION | INVALID_SECTOR
Result	None
Description	This command is used to erase a sector or multiple sectors of on-chip flash memory. The boot sector can not be erased by this command. To erase a single sector use the same "Start" and "End" sector numbers. Param2 is overwritten by the fixed value of 12 MHz, which is the IRC reference clock used by the flash controller.

25.6.2.4 Blank check sector(s) (IAP)

Table 332. IAP Blank check sector(s) command

Command	Blank check sector(s)
Input	**Command code: 53** **(decimal)** **Param0:** Start Sector Number **Param1:** End Sector Number (should be greater than or equal to start sector number).
Status code	CMD_SUCCESS | BUSY | SECTOR_NOT_BLANK | INVALID_SECTOR
Result	**Result0:** Offset of the first non blank word location if the Status Code is SECTOR_NOT_BLANK. **Result1:** Contents of non blank word location.
Description	This command is used to blank check a sector or multiple sectors of on-chip flash memory. To blank check a single sector use the same "Start" and "End" sector numbers.

25.6.2.5 Read Part Identification number (IAP)

Table 333. IAP Read Part Identification command

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

25.6.2.6 Read Boot code version number (IAP)

Table 334. IAP Read Boot Code version number command

Command	Read boot code version number
Input	**Command code: 55** **(decimal)** **Parameters:** None
Status code	CMD_SUCCESS
Result	**Result0:** 2 bytes of boot code version number. Read as <byte1(Major)>.<byte0(Minor)>
Description	This command is used to read the boot code version number.

25.6.2.7 Compare <address1> <address2> <no of bytes> (IAP)

Table 335. IAP Compare command

Command	Compare
Input	**Command code: 56** **(decimal)** **Param0(DST):** Starting flash or RAM address of data bytes to be compared. This address should be a word boundary. **Param1(SRC):** Starting flash or RAM address of data bytes to be compared. This address should be a word boundary. **Param2:** Number of bytes to be compared; should be a multiple of 4.
Status code	CMD_SUCCESS | COMPARE_ERROR | COUNT_ERROR (Byte count is not a multiple of 4) | ADDR_ERROR | ADDR_NOT_MAPPED
Result	**Result0:** Offset of the first mismatch if the Status Code is COMPARE_ERROR.
Description	This command is used to compare the memory contents at two locations.

25.6.2.8 Reinvoke ISP (IAP)

Table 336. IAP Reinvoke ISP

Command	Compare
Input	**Command code: 57 (decimal)**
Status code	None
Result	None.
Description	This command is used to invoke the bootloader in ISP mode. It maps boot vectors, sets PCLK = CCLK, and configures USART0 pins U0_RXD and U0_TXD. This command may be used when a valid user program is present in the internal flash memory and the ISP entry pin is not accessible to force the ISP mode.

25.6.2.9 ReadUID (IAP)

Table 337. IAP ReadUID command

Command	Compare
Input	**Command code: 58 (decimal)**
Status code	CMD_SUCCESS
Result	**Result0:** The first 32-bit word (at the lowest address). **Result1:** The second 32-bit word. **Result2:** The third 32-bit word. **Result3:** The fourth 32-bit word.
Description	This command is used to read the unique ID.

25.6.2.10 Erase page

Table 338. IAP Erase page command

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

25.6.2.11 IAP Status codes

Table 339. IAP Status codes Summary

Status Code	Mnemonic	Description
0	CMD_SUCCESS	Command is executed successfully.
1	INVALID_COMMAND	Invalid command.
2	SRC_ADDR_ERROR	Source address is not on a word boundary.
3	DST_ADDR_ERROR	Destination address is not on a correct boundary.
4	SRC_ADDR_NOT_MAPPED	Source address is not mapped in the memory map. Count value is taken in to consideration where applicable.
5	DST_ADDR_NOT_MAPPED	Destination address is not mapped in the memory map. Count value is taken in to consideration where applicable.
6	COUNT_ERROR	Byte count is not multiple of 4 or is not a permitted value.
7	INVALID_SECTOR	Sector number is invalid.
8	SECTOR_NOT_BLANK	Sector is not blank.
9	SECTOR_NOT_PREPARED_FOR_WRITE_OPERATION	Command to prepare sector for write operation was not executed.
10	COMPARE_ERROR	Source and destination data is not same.
11	BUSY	Flash programming hardware interface is busy.
17	ERR_ISP_IRC_NO_POWER	IRC is turned off. The IRC must be running to use the IAP calls.
18	ERR_ISP_FLASH_NO_POWER	-
1B	ERR_ISP_FLASH_NO_CLOCK	-

25.7 Functional description

25.7.1 UART Communication protocol

All UART 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.

25.7.1.1 UART ISP command format

"Command Parameter_0 Parameter_1 ... Parameter_n<CR><LF>" "Data" (Data only for Write commands).

25.7.1.2 UART 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).

25.7.1.3 UART ISP data format

The data stream is in plain binary format.

25.7.2 Memory and interrupt use for ISP and IAP

25.7.2.1 Interrupts during UART ISP

The boot block interrupt vectors located in the boot block of the flash are active after any reset.

25.7.2.2 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.

25.7.2.3 RAM used by ISP command handler

The stack of ISP commands is located at 0x1000 0270. The maximum stack usage is 540 byte and grows downwards.

25.7.2.4 RAM used by IAP command handler

The maximum stack usage in the user allocated stack space is 148 bytes and grows downwards.

25.7.3 Debugging

25.7.3.1 Comparing flash images

Depending on the debugger used and the IDE debug settings, the memory that is visible when the debugger connects might be the boot ROM, the internal SRAM, or the flash. To help determine which memory is present in the current debug environment, check the value contained at flash address 0x0000 0004. This address contains the entry point to the code in the ARM Cortex-M0+ vector table, which is the bottom of the boot ROM, the internal SRAM, or the flash memory respectively.

Table 340. Memory mapping in debug mode

Memory mapping mode	Memory start address visible at 0x0000 0004
Bootloader mode	0x1FFF 0000
User flash mode	0x0000 0000
User SRAM mode	0x1000 0000

25.7.3.2 Serial Wire Debug (SWD) flash programming interface

Debug tools can write parts of the flash image to RAM and then execute the IAP call "Copy RAM to flash" repeatedly with proper offset.

26.1 How to read this chapter

The power profiles are available for all LPC82x parts.

26.2 Features

- Includes ROM-based application services
- Power Management services
- Clocking services

26.3 Basic configuration

Specific power profile settings are required in the following situations:

- Disable all interrupts before making calls to the power profile API. You can re-enable the interrupts after the power profile API calls have completed.
- When using IAP commands, configure the power profiles in Default mode.

26.4 General description

The power consumption in Active and Sleep modes can be optimized for the application through simple calls to the power profile. The power configuration routine configures the LPC82x for one of the following power modes:

- Default mode corresponding to power configuration after reset.
- CPU performance mode corresponding to optimized processing capability.
- Efficiency mode corresponding to optimized balance of current consumption and CPU performance.
- Low-current mode corresponding to lowest power consumption.

In addition, the power profile includes routines to select the optimal PLL settings for a given system clock and PLL input clock.

Remark: Disable all interrupts before making calls to the power profile API. You can re-enable the interrupts after the power profile API calls have completed.

The API calls to the ROM are performed by executing functions which are pointed by a pointer within the ROM Driver Table. Figure 57 shows the pointer structure used to call the Power Profiles API.

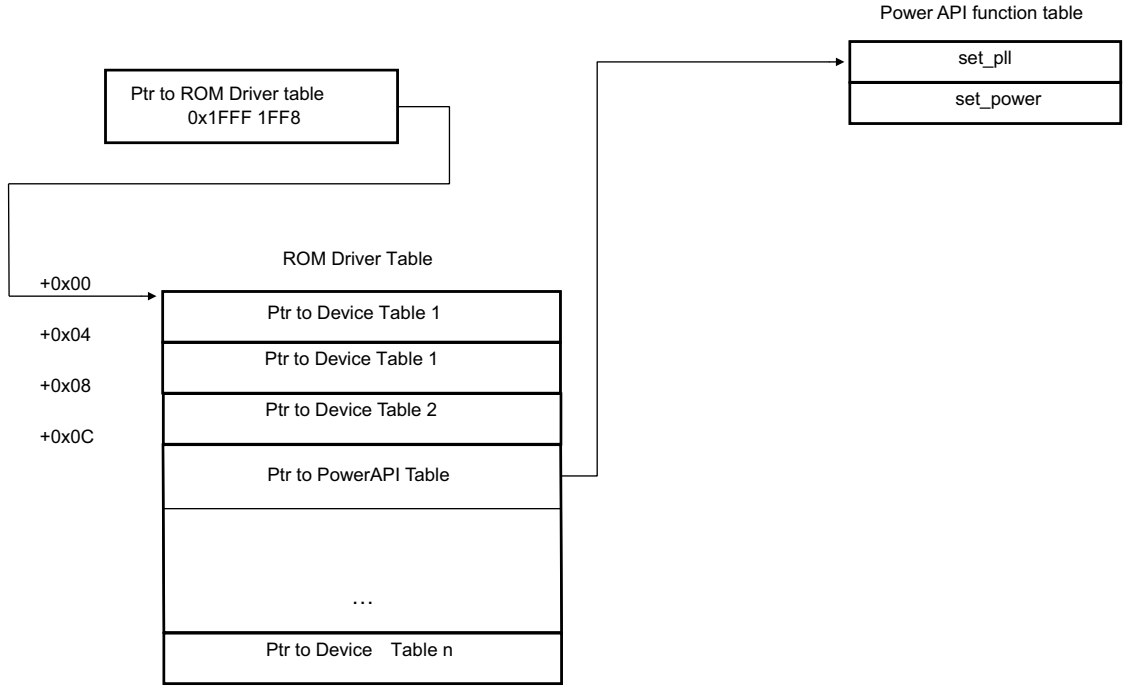


Fig 57. Power profiles pointer structure

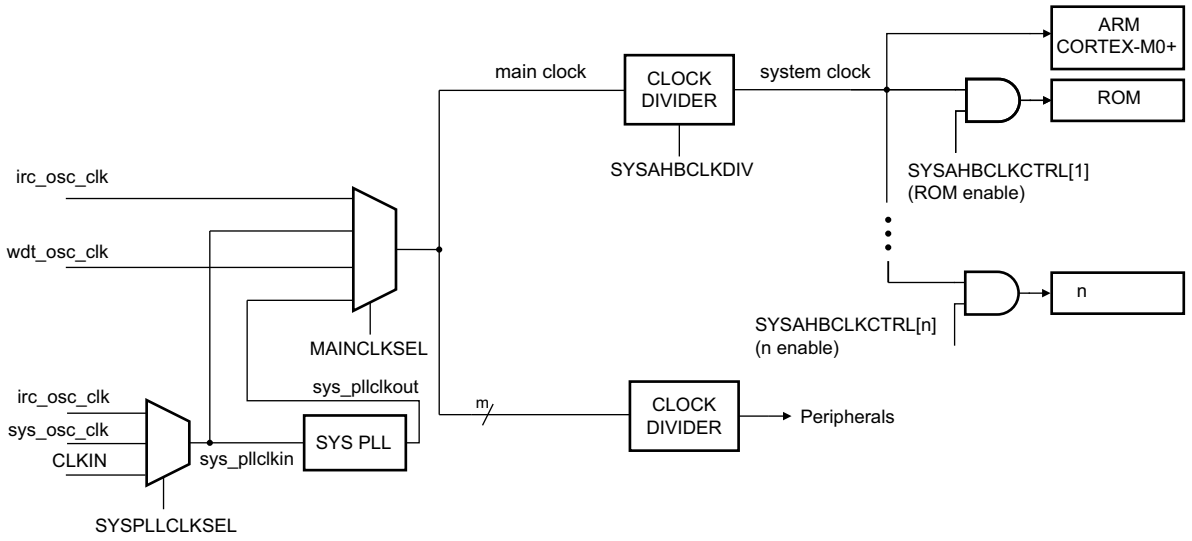


Fig 58. LPC82x clock configuration for power API use

26.5 API description

The power profile API provides functions to configure the system clock and optimize the system setting for lowest power consumption.

Table 341. Power profile API calls

API call	Description	Reference
`set_pll(command, result);`	Power API set pll routine	Table 342
`set_power(command, result);`	Power API set power routine	Table 343

The following elements have to be defined in an application that uses the power profiles:

```
typedef struct _PWRD {
     void (*set_pll)(unsigned int cmd[], unsigned int resp[]);
     void (*set_power)(unsigned int cmd[], unsigned int resp[]);
} PWRD;

#define ROM_DRIVER_BASE (0x1FFF1FF8UL)
#define LPC_PWRD_API    ((PWRD_API_T *) ((*(ROM_API_T * *) (ROM_DRIVER_BASE))->pPWRD))
```

See Section 3.5.2 for how to include the ROM driver structure.

26.5.1 set_pll

This routine sets up the system PLL according to the calling arguments. If the expected clock can be obtained by simply dividing the system PLL input, set_pll bypasses the PLL to lower system power consumption.

Remark: Before this routine is invoked, the PLL clock source (IRC/system oscillator) must be selected (Table 30), the main clock source must be set to the input clock to the system PLL (Table 32). and the system/AHB clock divider must be set to 1 (Table 34).

set_pll attempts to find a PLL setup that matches the calling parameters. Once a combination of a feedback divider value (SYSPLLCTRL, M), a post divider ratio (SYSPLLCTRL, P) and the system/AHB clock divider (SYSAHBCLKDIV) is found, set_pll applies the selected values and switches the main clock source selection to the system PLL clock out (if necessary).

The routine returns a result code that indicates if the system PLL was successfully set (PLL_CMD_SUCCESS) or not (in which case the result code identifies what went wrong). The current system frequency value is also returned. The application should use this information to adjust other clocks in the device (the SSP, UART, and WDT clocks, and/or the clock output).

Table 342. set_pll routine

Routine	set_pll
Input	**Param0:** system PLL input frequency (in kHz) **Param1:** expected system clock (in kHz) **Param2:** mode (CPU_FREQ_EQU, CPU_FREQ_LTE, CPU_FREQ_GTE, CPU_FREQ_APPROX) **Param3:** system PLL lock time-out
Result	**Result0:** PLL_CMD_SUCCESS | PLL_INVALID_FREQ | PLL_INVALID_MODE | PLL_FREQ_NOT_FOUND | PLL_NOT_LOCKED **Result1:** system clock (in kHz)

The following definitions are needed when making set_pll power routine calls:

```
/* set_pll mode options */
```

```
#define    CPU_FREQ_EQU          0
#define    CPU_FREQ_LTE          1
#define    CPU_FREQ_GTE          2
#define    CPU_FREQ_APPROX       3
/* set_pll result0 options */
#define    PLL_CMD_SUCCESS       0
#define    PLL_INVALID_FREQ      1
#define    PLL_INVALID_MODE      2
#define    PLL_FREQ_NOT_FOUND    3
#define    PLL_NOT_LOCKED        4
```

For a simplified clock configuration scheme see Figure 58. For more details see Figure 5.

26.5.1.1 Param0: system PLL input frequency and Param1: expected system clock

set_pll configures a setup in which the main clock does not exceed 30 MHz (see Figure 58). It easily finds a solution when the ratio between the expected system clock and the system PLL input frequency is an integer value, but it can also find solutions in other cases.

The system PLL input frequency (Param0) must be between 10000 to 25000 kHz (10 MHz to 25 MHz) inclusive. The expected system clock (Param1) must be between 1 and 30000 kHz inclusive. If either of these requirements is not met, set_pll returns PLL_INVALID_FREQ and returns Param0 as Result1 since the PLL setting is unchanged.

26.5.1.2 Param2: mode

The first priority of set_pll is to find a setup that generates the system clock at exactly the rate specified in Param1. If it is unlikely that an exact match can be found, input parameter mode (Param2) should be used to specify if the actual system clock can be less than or equal, greater than or equal or approximately the value specified as the expected system clock (Param1).

A call specifying CPU_FREQ_EQU will only succeed if the PLL can output exactly the frequency requested in Param1.

CPU_FREQ_LTE can be used if the requested frequency should not be exceeded (such as overall current consumption and/or power budget reasons).

CPU_FREQ_GTE helps applications that need a minimum level of CPU processing capabilities.

CPU_FREQ_APPROX results in a system clock that is as close as possible to the requested value (it may be greater than or less than the requested value).

If an illegal mode is specified, set_pll returns PLL_INVALID_MODE. If the expected system clock is out of the range supported by this routine, set_pll returns PLL_FREQ_NOT_FOUND. In these cases the current PLL setting is not changed and Param0 is returned as Result1.

26.5.1.3 Param3: system PLL lock time-out

It should take no more than 100 μs for the system PLL to lock if a valid configuration is selected. If Param3 is zero, set_pll will wait indefinitely for the PLL to lock. A non-zero value indicates how many times the code will check for a successful PLL lock event before it returns PLL_NOT_LOCKED. In this case the PLL settings are unchanged and Param0 is returned as Result1.

Remark: The time it takes the PLL to lock depends on the selected PLL input clock source (IRC/system oscillator) and its characteristics. The selected source can experience more or less jitter depending on the operating conditions such as power supply and/or ambient temperature. This is why it is suggested that when a good known clock source is used and a PLL_NOT_LOCKED response is received, the set_pll routine should be invoked several times before declaring the selected PLL clock source invalid.

Hint: setting Param3 equal to the system PLL frequency [Hz] divided by 10000 will provide more than enough PLL lock-polling cycles.

26.5.2 set_power

This routine configures the device's internal power control settings according to the calling arguments. The goal is to reduce active power consumption while maintaining the feature of interest to the application close to its optimum.

Remark: Use the set_power routine with SYSAHBCLKDIV = 1 (System clock divider register, see Table 34 and Figure 58).

set_power returns a result code that reports whether the power setting was successfully changed or not.

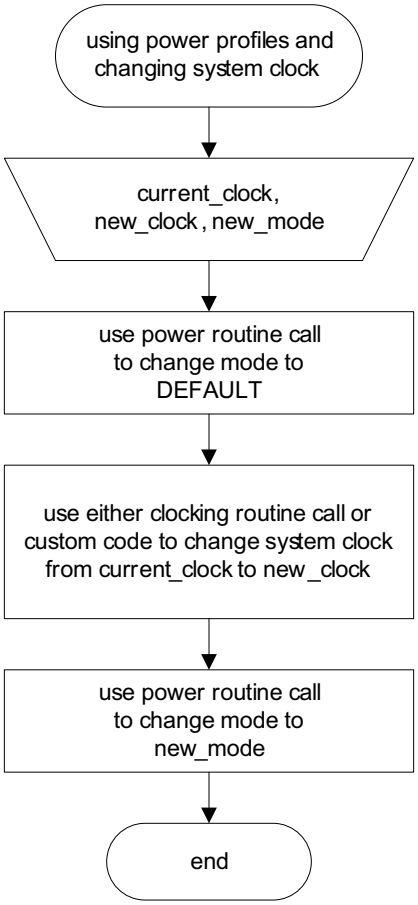


Fig 59. Power profiles usage

Table 343. set_power routine

Routine	set_power
Input	**Param0:** main clock (in MHz) **Param1:** mode (PWR_DEFAULT, PWR_CPU_PERFORMANCE, PWR_EFFICIENCY, PWR_LOW_CURRENT) **Param2:** system clock (in MHz)
Result	**Result0:** PWR_CMD_SUCCESS | PWR_INVALID_FREQ | PWR_INVALID_MODE

The following definitions are needed for set_power routine calls:

```
/* set_power mode options */
#define    PWR_DEFAULT          0
#define    PWR_CPU_PERFORMANCE  1
#define    PWR_EFFICIENCY       2
#define    PWR_LOW_CURRENT      3
/* set_power result0 options */
#define    PWR_CMD_SUCCESS      0
#define    PWR_INVALID_FREQ     1
#define    PWR_INVALID_MODE     2
```

For a simplified clock configuration scheme see Figure 58. For more details see Figure 5.

26.5.2.1 Param0: main clock

The main clock is the clock rate the microcontroller uses to source the system's and the peripherals' clock. It is configured by either a successful execution of the clocking routine call or a similar code provided by the user. This operand must be an integer between 1 to 30 MHz inclusive. If a value out of this range is supplied, set_power returns PWR_INVALID_FREQ and does not change the power control system.

26.5.2.2 Param1: mode

The input parameter mode (Param1) specifies one of four available power settings. If an illegal selection is provided, set_power returns PWR_INVALID_MODE and does not change the power control system.

PWR_DEFAULT keeps the device in a baseline power setting similar to its reset state.

PWR_CPU_PERFORMANCE configures the microcontroller so that it can provide more processing capability to the application. CPU performance is 30% better than the default option.

PWR_EFFICIENCY setting was designed to find a balance between active current and the CPU's ability to execute code and process data. In this mode the device outperforms the default mode both in terms of providing higher CPU performance and lowering active current.

PWR_LOW_CURRENT is intended for those solutions that focus on lowering power consumption rather than CPU performance.

26.5.2.3 Param2: system clock

The system clock is the clock rate at which the microcontroller core is running when set_power is called. This parameter is an integer between from 1 and 30 MHz inclusive.

26.6 Functional description

26.6.1 Clock control

See Section 26.6.1.1 to Section 26.6.1.6 for examples of the clock control API.

26.6.1.1 Invalid frequency (device maximum clock rate exceeded)

```
command[0] = 12000;
command[1] = 60000;
command[2] = CPU_FREQ_EQU;
command[3] = 0;
LPC_PWRD_API->set_pll(command, result);
```

The above code specifies a 12 MHz PLL input clock and a system clock of exactly 60 MHz. The application was ready to infinitely wait for the PLL to lock. But the expected system clock of 60 MHz exceeds the maximum of 30 MHz. Therefore set_pll returns PLL_INVALID_FREQ in result[0] and 12000 in result[1] without changing the PLL settings.

26.6.1.2 Invalid frequency selection (system clock divider restrictions)

```
command[0] = 12000;
command[1] = 40;
command[2] = CPU_FREQ_LTE;
command[3] = 0;
LPC_PWRD_API->set_pll(command, result);
```

The above code specifies a 12 MHz PLL input clock, a system clock of no more than 40 kHz and no time-out while waiting for the PLL to lock. Since the maximum divider value for the system clock is 255 and running at 40 kHz would need a divide by value of 300, set_pll returns PLL_INVALID_FREQ in result[0] and 12000 in result[1] without changing the PLL settings.

26.6.1.3 Exact solution cannot be found (PLL)

```
command[0] = 12000;
command[1] = 25000;
command[2] = CPU_FREQ_EQU;
command[3] = 0;
LPC_PWRD_API->set_pll(command, result);
```

The above code specifies a 12 MHz PLL input clock and a system clock of exactly 25 MHz. The application was ready to infinitely wait for the PLL to lock. Since there is no valid PLL setup within earlier mentioned restrictions, set_pll returns PLL_FREQ_NOT_FOUND in result[0] and 12000 in result[1] without changing the PLL settings.

26.6.1.4 System clock less than or equal to the expected value

```
command[0] = 12000;
command[1] = 25000;
command[2] = CPU_FREQ_LTE;
command[3] = 0;
LPC_PWRD_API->set_pll(command, result);
```

The above code specifies a 12 MHz PLL input clock, a system clock of no more than 25 MHz and no locking time-out. set_pll returns PLL_CMD_SUCCESS in result[0] and 24000 in result[1]. The new system clock is 24 MHz.

26.6.1.5 System clock greater than or equal to the expected value

```
command[0] = 12000;
command[1] = 20000;
command[2] = CPU_FREQ_GTE;
command[3] = 0;
LPC_PWRD_API->set_pll(command, result);
```

The above code specifies a 12 MHz PLL input clock, a system clock of at least 20 MHz and no locking time-out. set_pll returns PLL_CMD_SUCCESS in result[0] and 24000 in result[1]. The new system clock is 24 MHz.

26.6.1.6 System clock approximately equal to the expected value

```
command[0] = 12000;
command[1] = 16500;
command[2] = CPU_FREQ_APPROX;
command[3] = 0;
LPC_PWRD_API->set_pll(command, result);
```

The above code specifies a 12 MHz PLL input clock, a system clock of approximately 16.5 MHz and no locking time-out. set_pll returns PLL_CMD_SUCCESS in result[0] and 16000 in result[1]. The new system clock is 16 MHz.

26.6.2 Power control

See Section 26.6.1.1 and Section 26.6.2.2 for examples of the power control API.

26.6.2.1 Invalid frequency (device maximum clock rate exceeded)

```
command[0] = 30;
command[1] = PWR_CPU_PERFORMANCE;
command[2] = 40;
LPC_PWRD_API->set_power(command, result);
```

The above setup would be used in a system running at the main and system clock of 30 MHz, with a need for maximum CPU processing power. Since the specified 40 MHz clock is above the 30 MHz maximum, set_power returns PWR_INVALID_FREQ in result[0] without changing anything in the existing power setup.

26.6.2.2 An applicable power setup

```
command[0] = 24;
command[1] = PWR_CPU_EFFICIENCY;
command[2] = 24;
LPC_PWRD_API->set_power(command, result);
```

The above code specifies that an application is running at the main and system clock of 24 MHz with emphasis on efficiency. set_power returns PWR_CMD_SUCCESS in result[0] after configuring the microcontroller's internal power control features.

27.1 How to read this chapter

The USART ROM driver routines are available on all LPC82x parts.

27.2 Features

- Send and receive characters in asynchronous or synchronous mode
- Send and receive multiple characters (line) in asynchronous or synchronous UART mode

27.3 General description

The UART API handles sending and receiving characters using any of the USART blocks in asynchronous mode.

Remark: Because all USARTS share a common fractional divider, the uart_init routine returns the value for the common divider.

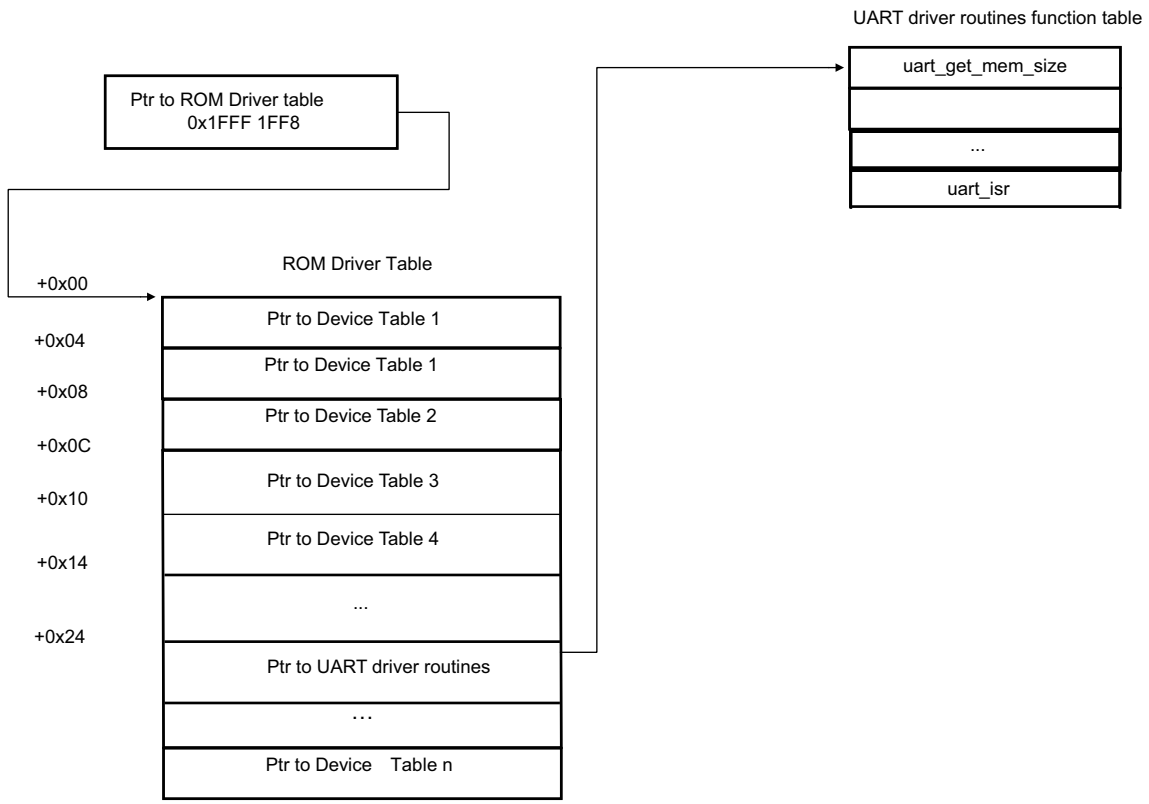


Fig 60. USART driver routines pointer structure

27.4 API description

The UART API contains functions to send and receive characters via any of the USART blocks.

Table 344. UART API calls

API call	Description	Reference
`uint32_t ramsize_in_bytes uart_get_mem_size( void) ;`	UART get memory size	Table 345
`UART_HANDLE_T* uart_setup(uint32_t base_addr, uint8_t *ram) ;`	UART set-up	Table 346
`uint32_t uart_init(UART_HANDLE_T* handle, UART_CONFIG set);`	UART init	Table 347
`uint8_t uart_get_char(UART_HANDLE_T* handle);`	UART get character	Table 348
`void uart_put_char(UART_HANDLE_T* handle, uint8_t data);`	UART put character	Table 349
`uint32_t uart_get_line(UART_HANDLE_T* handle, UART_PARAM_T param);`	UART get line	Table 350
`uint32_t uart_put_line(UART_HANDLE_T* handle, UART_PARAM_T param);`	UART put line	Table 351
`void uart_isr(UART_HANDLE_T* handle);`	UART interrupt service routine	Table 352

The following structure has to be defined to use the UART API:

```
typedef struct  UARTD_API {         // index of all the uart driver functions
                uint32_t (*uart_get_mem_size)(void);
                UART_HANDLE_T (*uart_setup)(uint32_t base_addr, uint8_t *ram);
                uint32_t (*uart_init)(UART_HANDLE_T handle, UART_CONFIG_T *set);
                //--polling functions--//
                uint8_t (*uart_get_char)(UART_HANDLE_T handle);
                void (*uart_put_char)(UART_HANDLE_T handle, uint8_t data);
               uint32_t (*uart_get_line)(UART_HANDLE_T handle, UART_PARAM_T * param);
               uint32_t (*uart_put_line)(UART_HANDLE_T handle, UART_PARAM_T * param);
                //--interrupt functions--//
                void (*uart_isr)(UART_HANDLE_T handle);
} UARTD_API_T  ;             // end of structure

#define ROM_DRIVER_BASE (0x1FFF1FF8UL)
#define LPC_UART_API    ((UARTD_API_T *) ((*(ROM_API_T * *)
     (ROM_DRIVER_BASE))->pUARTD))
```

See Section 3.5.2 for how to include the ROM driver structure.

27.4.1 UART get memory size

Table 345. uart_get_mem_size

Routine	uart_get_mem_size
Prototype	`uint32_t ramsize_in_bytes uart_get_mem_size( void) ;`
Input parameter	None.
Return	Memory size in bytes.
Description	Get the memory size needed by one UART instance.

27.4.2 UART setup

Table 346. uart_setup

Routine	uart_setup
Prototype	`UART_HANDLE_T* uart_setup(uint32_t base_addr, uint8_t *ram) ;`
Input parameter	base_addr: Base address of register for this uart block. ram: Pointer to the memory space for uart instance. The size of the memory space can be obtained by the uart_get_mem_size function.
Return	The handle to corresponding uart instance.
Description	Setup UART instance with provided memory and return the handle to this instance.

27.4.3 UART init

Table 347. uart_init

Routine	uart_init
Prototype	`uint32_t uart_init(UART_HANDLE_T* handle, UART_CONFIG set);`
Input parameter	handle: The handle to the uart instance. set: configuration for uart operation.
Return	Fractional divider value if System clock is not integer multiples of baud rate.
Description	Setup baud rate and operation mode for uart, then enable uart.

27.4.4 UART get character

Table 348. uart_get_char

Routine	uart_get_char
Prototype	`uint8_t uart_get_char(UART_HANDLE_T* handle);`
Input parameter	handle: The handle to the uart instance.
Return	Received data
Description	Receive one Char from uart. This functions is only returned after Char is received. In case Echo is enabled, the received data is sent out immediately.

27.4.5 UART put character

Table 349. uart_put_char

Routine	uart_put_char
Prototype	`void uart_put_char(UART_HANDLE_T* handle, uint8_t data);`
Input parameter	handle: The handle to the uart instance. data: data to be sent out.
Return	None.
Description	Send one Char through uart. This function is only returned after data is sent.

27.4.6 UART get line

Table 350. uart_get_line

Routine	uart_get_line
Prototype	`uint32_t uart_get_line(UART_HANDLE_T* handle, UART_PARAM_T param);`
Input parameter	handle: The handle to the uart instance. param: Refer to UART_PARAM_T definition.
Return	Error code: ERR_UART_RECEIVE_ON - UART receive is ongoing.
Description	Receive multiple bytes from UART.

27.4.7 UART put line

Table 351. uart_put_line

Routine	uart_put_line
Prototype	`uint32_t uart_put_line(UART_HANDLE_T* handle, UART_PARAM_T param);`
Input parameter	handle: The handle to the uart instance. param: Refer to UART_PARAM_T definition.
Return	Error code: ERR_UART_SEND_ON - UART sending is ongoing.
Description	Send string (end with \0) or raw data through UART.

27.4.8 UART interrupt service routine

Table 352. uart_isr

Routine	uart_isr
Prototype	`void uart_isr(UART_HANDLE_T* handle);`
Input parameter	handle: The handle to the uart instance.
Return	None.
Description	UART interrupt service routine. To use this routine, the corresponding USART interrupt must be enabled. This function is invoked by the user ISR.

27.4.9 Error codes

Table 353. Error codes

Return code	Error Code	Description
0x0008 0001	ERR_UART_RXD_BUSY = ERR_UART_BASE+1,	UART receive is busy
0x0008 0002	ERR_UART_TXD_BUSY	UART transmit is busy
0x0008 0003	ERR_UART_OVERRUN_FRAME_PARITY_NOISE	Overrun error, Frame error, parity error, RxNoise error
0x0008 0004	ERR_UART_UNDERRUN	Underrun error
0x0008 0005	ERR_UART_PARAM	Parameter error

27.4.10 UART ROM driver variables

27.4.10.1 UART_CONFIG structure

```
typdef  struct UART_CONFIG {
     uint32_t sys_clk_in_hz; // main clock/UARTCLKDIV in Hz
     uint32_t baudrate_in_hz; // Baudrate in hz
     uint8_t  config; //bit 1:0
               // 00: 7 bits length, 01: 8 bits lenght, others: reserved
               //bit3:2
               //   00: No Parity, 01: reserved, 10: Even, 11: Odd
               //bit4
               //   0: 1 Stop bit, 1: 2 Stop bits
     uint8_t sync_mod;  //bit0: 0(Async mode), 1(Sync mode)
                    //bit1: 0(Un_RXD is sampled on the falling edge of SCLK)
                    //       1(Un_RXD is sampled on the rising edge of SCLK)
                    //bit2: 0(Start and stop bits are transmitted as in asynchronous
                    mode)
                    //       1(Start and stop bits are not transmitted)
                    //bit3: 0(the UART is a slave on Sync mode)
                    //       1(the UART is a master on Sync mode)
     uint16_t error_en; //Bit0: OverrunEn, bit1: UnderrunEn, bit2: FrameErrEn,
                 // bit3: ParityErrEn, bit4: RxNoiseEn
}
```

27.4.10.2 UART_HANDLE_T

The handle to the instance of the UART driver. Each UART has one handle, so there can be several handles for up to three UART blocks. This handle is created by Init API and used by the transfer functions for the corresponding UART block.

```
typedef  void     *UART_HANDLE_T   ;    // define TYPE for uart handle pointer
```

27.4.10.3 UART_PARAM_T

```
typedef struct  uart_A { // parms passed to uart driver function
     uint8_t  *  buffer ; // The pointer of buffer.
                    // For uart_get_line function, buffer for receiving data.
                    // For uart_put_line function, buffer for transmitting data.
     uint32_t   size;     // [IN] The size of buffer.
                    //[OUT] The number of bytes transmitted/received.
     uint16_t    transfer_mode ;
                    // 0x00:  For uart_get_line function, transfer without
                    // termination.
                    // For uart_put_line function, transfer without termination.
                    // 0x01:  For uart_get_line function, stop transfer when
                    // <CR><LF> are received.
                    // For uart_put_line function, transfer is stopped after
                    // reaching \0. <CR><LF> characters are sent out after that.
                    // 0x02:  For uart_get_line function, stop transfer when <LF>
                    // is received.
                    // For uart_put_line function, transfer is stopped after
                    // reaching \0. A <LF> character is sent out after that.
```

```
                    //0x03:  For uart_get_line function, RESERVED.
                    // For uart_put_line function, transfer is stopped after
                    // reaching \0.
    uint16_t   driver_mode;
                    //0x00: Polling mode, function is blocked until transfer is
                    // finished.
                    // 0x01: Intr mode, function exit immediately, callback function
                    // is invoked when transfer is finished.
                    //0x02: RESERVED
    UART_CALLBK_T     callback_func_pt; // callback function
}  UART_PARAM_T  ;
```

28.1 How to read this chapter

The SPI ROM driver routines are available on all parts.

28.2 Features

- Send and receive data in slave or master mode.
- Support for DMA mode.

28.3 General description

The SPI API handles SPI data transfer in master and slave modes.

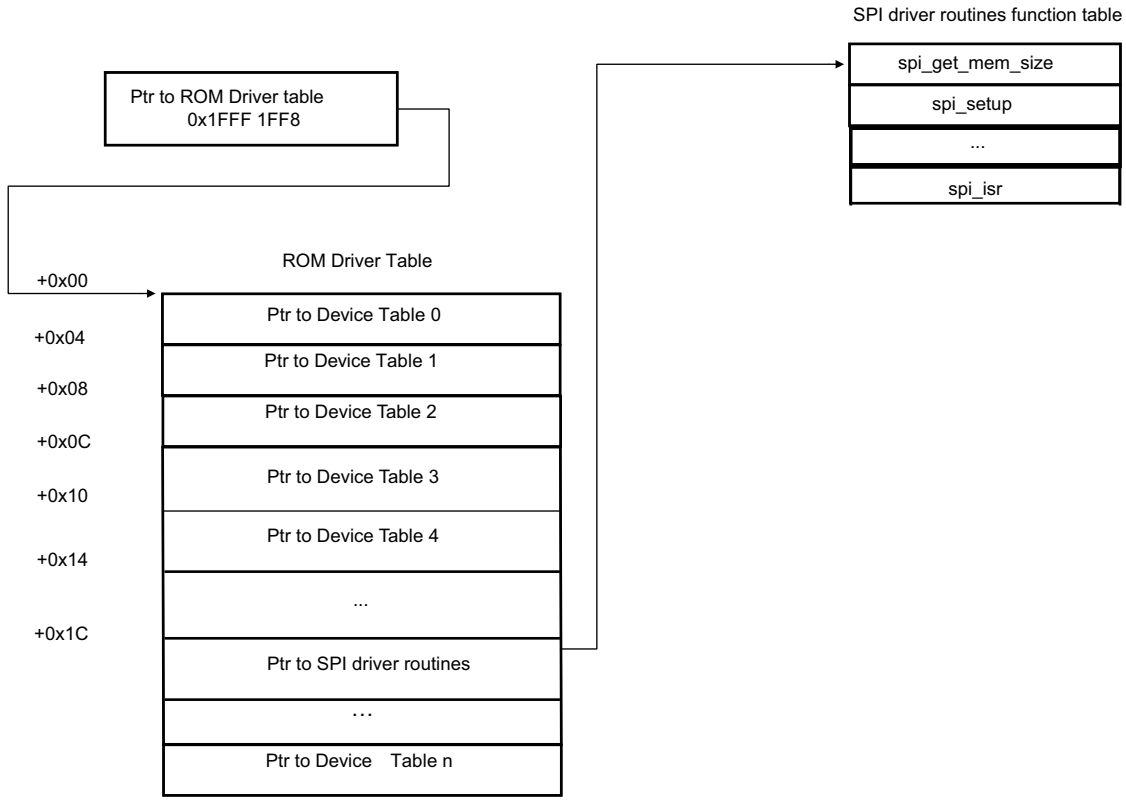


Fig 61. SPI driver routines pointer structure

28.4 API description

The SPI API contains functions to send and receive data via any of the SPI interfaces in master and slave modes.

Table 354. SPI API calls

API call	Description	Reference
`uint32_t spi_get_mem_size(void);`	Memory size for one SPI instance	Table 355
`SPI_HANDLE_T* spi_setup(uint32_t base_addr, uint8_t *ram);`	Set up SPI instance and return handle	Table 356
`uint32_t spi_init(SPI_HANDLE_T* handle, SPI_CONFIG set);`	Set up SPI operating mode	Table 357
`uint32_t spi_master_transfer(SPI_HANDLE_T* handle, SPI_PARAM_T param);`	Send or receive data in master mode	Table 358
`uint8_t spi_slave_transfer(SPI_HANDLE_T* handle, SPI_PARAM_T param);`	Send or receive data in slave mode	Table 359
`void spi_isr(SPI_HANDLE_T* handle);`	Interrupt service routine	Table 360

The following structure has to be defined to use the SPI API:

```
typedef struct {   // index of all the SPI driver functions
     uint32_t (*spi_get_mem_size)(void);
     SPI_HANDLE_T (*spi_setup)(uint32_t base_addr, uint8_t *ram);
     void (*spi_init)(SPI_HANDLE_T handle, SPI_CONFIG_T *set);
     uint32_t (*spi_master_transfer)(SPI_HANDLE_T handle, SPI_PARAM_T * param);
     uint32_t (*spi_slave_transfer)(SPI_HANDLE_T handle, SPI_PARAM_T * param);
     //--interrupt functions--//
     void (*spi_isr)(SPI_HANDLE_T handle);
} SPID_API_T  ;
```

28.4.1 SPI get memory size

Table 355. spi_get_mem_size

Routine	spi_get_mem_size
Prototype	`uint32_t spi_get_mem_size(void);`
Input parameter	None.
Return	Memory size in bytes.
Description	Get the memory size needed by one SPI instance.

28.4.2 SPI setup

Table 356. spi_setup

Routine	spi_setup
Prototype	`SPI_HANDLE_T* spi_setup(uint32_t base_addr, uint8_t *ram);`
Input parameter	base_addr: Register base address for this SPI block. ram: Pointer to the memory space for SPI instance; the memory size is obtained from the spi_get_mem_size() function.
Return	The handle to corresponding SPI instance.
Description	Set up SPI instance with provided memory and return the handle to this instance.

28.4.3 SPI init

See Section 28.4.8.2 for the SPI_CONFIG and Section 28.4.8.1 for SPI_HANDLE_T variables.

Table 357. spi_init

Routine	spi_init
Prototype	`uint32_t spi_init(SPI_HANDLE_T* handle, SPI_CONFIG set);`
Input parameter	handle: The handle to the SPI instance. set: configuration for SPI operation.
Return	None.
Description	Set up operation mode for SPI, then enable SPI.

28.4.4 SPI master data transfer

See Section 28.4.8.1 for SPI_HANDLE_T variable.

Table 358. spi_master_transfer

Routine	spi_master_transfer
Prototype	`uint32_t spi_master_transfer(SPI_HANDLE_T* handle, SPI_PARAM_T param);`
Input parameter	handle: The handle to the SPI instance. param: Refer to SPI_PARAM_T definition.
Return	Error code.
Description	Master send or receive data through SPI.

28.4.5 SPI slave data transfer

See Section 28.4.8.1 for SPI_HANDLE_T variable.

Table 359. spi_slave_transfer

Routine	spi_slave_transfer
Prototype	`uint8_t spi_slave_transfer(SPI_HANDLE_T* handle, SPI_PARAM_T param);`
Input parameter	handle: The handle to the SPI instance. param: Refer to SPI_PARAM_T definition.
Return	Error code.
Description	Slave send or receive data to SPI.

28.4.6 SPI interrupt service routine

See Section 28.4.8.1 for SPI_HANDLE_T variable.

Table 360. spi_isr

Routine	spi_isr
Prototype	`void spi_isr(SPI_HANDLE_T* handle);`
Input parameter	handle: The handle to the SPI instance.
Return	None.
Description	SPI interrupt service routine. To use this routine, the corresponding SPI interrupt must be enabled. This function is invoked by the user ISR.

28.4.7 Error codes

Table 361. Error codes

Return code	Error Code	Description
0x000E 0001	ERR_SPI_RXOVERRUN	
0x000E 0002	ERR_SPI_TXUNDERRUN	
0x000E 0003	ERR_SPI_SELNASSERT	
0x000E 0004	ERR_SPI_SELNDEASSERT	
0x000E 0005	ERR_SPI_CLKSTALL	
0x000E 0006	ERR_SPI_PARAM	
0x000E 0007	ERR_SPI_INVALID_LENGTH	

```
typedef enum
{
 ERR_SPI_BASE = 0x000E0000,
  /*0x000E0001*/ ERR_SPI_RXOVERRUN=ERR_SPI_BASE+1,
  /*0x000E0002*/ ERR_SPI_TXUNDERRUN,
  /*0x000E0003*/ ERR_SPI_SELNASSERT,
  /*0x000E0004*/ ERR_SPI_SELNDEASSERT,
  /*0x000E0005*/ ERR_SPI_CLKSTALL,
  /*0x000E0006*/ ERR_SPI_PARAM,
  /*0x000E0007*/ ERR_SPI_INVALID_LENGTH
} ErrorCode_t;
```

28.4.8 SPI ROM driver variables

28.4.8.1 SPI_HANDLE_T

The handle to the instance of the SPI driver. Each SPI has one handle, so there can be several handles for each SPI block. This handle is created by Init API and used by the transfer functions for the corresponding SPI block.

```
typedef  void     SPI_HANDLE_T   ;    // define TYPE for SPI handle pointer
```

28.4.8.2 SPI_CONFIG_T

```
Typdef  struct {
     uint32_t delay;
     uint32_t divider;
     uint32_t  config;// config register
     uint32_t error_en;//Bit0: OverrunEn, bit1: UnderrunEn,
}SPI_CONFIG_T;
```

The following variables are used in this structure:

`delay`: Configures the delay between SSEL and data transfers and between frames. The value is the content of the SPI DLY register. See Table 191.

`divider`: Clock divider value DIVVAL in the SPI DIV register. See Table 199.

`config`: Enable SPI, configure master/slave, configure signal phase and polarity. The value is the content of the SPI CFG register. See Table 190.

error_en: Enables the receive overrun and transmit underrun error interrupts.

28.4.8.3 SPI_PARAM_T

This structure defines the SPI configuration.

```
typedef struct {     // params passed to SPI driver function
     uint16_t *tx_buffer;// SPI TX buffer, needed in master and slave TX only
                         //mode
     uint16_t *rx_buffer;// SPI RX buffer, needed in master RX only, TX and RX,
                         //and slave RX mode,
     uint32_t size;// total number of SPI frames
     uint32_t fsize_sel;// data lenth of one transfer and SPI SSELx select in TXCTL
     uint32_t tx_rx_flag;
     uint32_t driver_mode;
     SPI_DMA_CFG_T *dma_cfg;// DMA configuration
     SPI_CALLBK_Tcb; // callback function
     SPI_DMA_REQ_T dma_cb; // SPI DMA channel setup callback
} SPI_PARAM_T;
```

The following variables are used in this structure:

size: number of SPI frames. A transfer can consist of several transmits of the TXCTLDAT register and of several frames.

fsize_sel: write the contents of the SPI TXCTL register to select the data length and the slave select lines. In slave mode, you need to only select the data length. See Table 198 “SPI Transmitter Control register (TXCTL, addresses 0x4005 8020 (SPI0), 0x4005 C020 (SPI1)) bit description”.

tx_rx_flag: select receive/transmit mode.

0x0 = SPI transmit only mode

0x1 = SPI receive only mode

0x2 = SPI transmit and receive mode, master mode only, transmit data transfer on MOSI and receive data transfer on MISO.

driver_mode: select the driver mode.

0x0 = Polling mode. Function is blocked until transfer has finished.

0x1 = Interrupt mode. Function exits immediately and a call back function is invoked when the transfer has finished.

0x2 = DMA mode. Data transferred by SPI is processed by DMA. The DMA_req function is called for SPI DMA channel set up. The callback function indicates when the transfer is complete.

28.4.8.4 SPI_DMA_CFG_T

This structure configures the channels used for DMA transfer between the SPI and memory. See Table 148 “DMA requests”, for the DMA channels connected to the SPI receive and transmit request lines.

To perform a DMA transfer for receive data, also enable the SPI transmit DMA channel, so that a SPI clock is generated.

Configure the DMA transfer using the DMA API. The handle to the DMA instance returned by the DMA API is used by the SPI DMA for the SPI transfer.

```
typedef struct {
     uint32_t        dma_txd_num; // SPI TX DMA channel number.
     uint32_t        dma_rxd_num;// SPI RX DMA channel number. In order to do a SPI RX
                                 // DMA, a SPI TX DMA is also needed to generated SPI
                                 // clock.
     DMA_HANDLE_T hDMA;// DMA handle
} SPI_DMA_CFG_T;
```

28.4.8.5 CALLBK_T

The callback is either invoked in the SPI interrupt handler or the DMA interrupt handler depending on the driver mode.

```
typedef void  (*CALLBK_T) (ErrorCode_t error_code, uint32_t num_transfer );
```

`error_code`: SPI error code

`num_transfer`: In interrupt mode, this parameter indicates the number of SPI frames. In DMA mode, this parameter is always zero.

In interrupt mode, error code and number of transfers will be passed to the callback.

In DMA mode, the callback indicates the completion of the DMA transfer.

In master mode, the `frame_size` parameter must be set so that the EOT (End of the transfer) bit in the SPICTL register is set to assert the SPI slave select line.

28.4.8.6 SPI_DMA_REQ_T

When the driver mode is DMA, the following DMA set-up callback is invoked:

```
typedef ErrorCode_t  (*SPI_DMA_REQ_T) (SPI_HANDLE_T handle, SPI_DMA_CFG_T *dma_cfg);
```

To set up the DMA channel, the source address, destination address, DMA transfer length, DMA request information must be retrieved from the driver structure which has been originally passed from the SPI_PARAM_T structure.

28.5 Functional description

28.5.1 Example (no DMA)

Send and receive characters in interrupt mode. Use the SPI API as follows:

1. Assign the SCK, MOSI, MISO, and SSEL0 functions to pins in the switch matrix and set up the system clock.
2. Global defines:

```
#define rom_drivers_ptr (* (ROM **) 0x03000200)
SPID_API_T *  pSpiApi ;  //define pointer to type API function addr table
SPI_HANDLE_T* spi_handle; //handle to SPI API

SPI_PARAM_T     param;
```

```
#define RAMBLOCK_H  60
uint32_t  start_of_ram_block0[ RAMBLOCK_H ] ;

#define BUFFER_SIZE   100
uint32_t spi_buffer[BUFFER_SIZE];
```

3. Define configuration structure and initialize pointer to the API:

```
SPI_CONFIG_T    spi_set;
 pSpiApi = (SPID_API_T *) (rom_drivers_ptr->pSPID);
```

4. Initialize SPI:

```
size_in_bytes = pSpiApi->spi_get_mem_size() ; //size_in_bytes/4 must be
 // < RAMBLOCK_H
spi_handle = pSpiApi->spi_setup(LPC_SPI0_BASE, (uint8_ *)start_of_ram_block0);
```

5. Set up the SPI0 in master mode by defining the SPI_CONFIG_T structure:

```
spi_set.delay =
DLY_PREDELAY(0x0)|DLY_POSTDELAY(0x0)|DLY_FRAMEDELAY(0x0)|DLY_INTERDELAY(0x0);
spi_set.divider = 0xFFFF; // divide system clock by 0xFFFF for SCK
spi_set.config = CFG_MASTER; //master mode
spi_set.error_en = STAT_RXOVERRUN | STAT_TXUNDERRUN;
```

6. Initialize SPI transfer:

```
pSpiApi->spi_init(spi_handle, &spi_set);
```

7. Enable the SPI interrupt in the NVIC.

8. Set up the SPI parameter structure SPI_PARAM_T:

```
param.driver_mode = 0x01; //interrupt mode.
param.tx_rx_flag  = 0x02; // TX AND RX
param.fsize_sel = TXDATCTL_SSELN(SLAVE0) | TXDATCTL_FSIZE(MASTER_FRAME_SIZE);
param.tx_buffer = (uint16_t *)tx_buffer;
param.rx_buffer = (uint16_t *)rx_buffer;
param.eof_flag  = 0;                    // no framedelay used
param.size = 10;  //10 transfers of TXDATCTL_FSIZE size data.
// If receive callback is invoked, transmit follows automatically. Only one
// callback is needed.
param.callback_func_pt = receive_callback;
```

9. Define the receive callback function invoked when the transmit has finished:

```
void  receive_callback(uint32_t err_code, uint32_t n ) {
error_code = (ErrorCode_t)err_code;
     if (err_code != LPC_OK)
     while(1);
     receive_tag = 1;
}
```

10. Master transmits data. Slave sends some data back.

```
receive_tag = 0;
pSpiApi->spi_master_transfer(spi_handle, &param);
while(!receivetag); //wait until receivetag is set
```

28.5.2 Example (using DMA)

The DMA sends and receives characters via the SPI in master mode. Use the SPI API and DMA API as follows:

1. Assign the SCK, MOSI, MISO, and SSEL0 functions to pins in the switch matrix.
2. Global defines:

```
SPID_API_T *  pSpiApi ;  //define pointer to type API function addr table
SPI_HANDLE_T* spi_handle; //handle to SPI API

SPI_PARAM_T     param;
SPI_CONFIG_T    spi_cfg;
SPI_DMA_CFG_T  dmc_cfg;

#define RAMBLOCK_H  60
uint32_t  start_of_ram_block0[ RAMBLOCK_H ] ;

#define BUFFER_SIZE   100
uint32_t adc_buffer[BUFFER_SIZE];
```

3. Initialize pointer to the SPI API:

```
 pSpiApi = (SPID_API_T *) (rom_drivers_ptr->pSPID);
```

4. Initialize SPI:

```
size_in_bytes = pSpiApi->spi_get_mem_size() ; //size_in_bytes/4 must be
 // < RAMBLOCK_H
spi_handle = pSpiApi->spi_setup(LPC_SPI0_BASE, (uint8_ *)start_of_ram_block0);
```

5. Set up the DMA:

```
size_in_bytes =  pDmaApi->dma_get_mem_size(); //size_in_bytes/4 must be
     // < RAMBLOCK_H

pDmaApi = (DMAD_API_T *) (rom_drivers_ptr->pDMAD);
align_location = (uint32_t *)0x02008000;
dma_handle = pDmaApi->dma_setup(LPC_DMA_BASE, (uint8_t *)align_location);
```

6. Set up the SPI0 in DMA mode by defining the SPI_CONFIG_T structure:

```
spi_set.delay =
DLY_PREDELAY(0x0)|DLY_POSTDELAY(0x0)|DLY_FRAMEDELAY(0x0)|DLY_INTERDELAY(0x0);
spi_set.divider = 0xFFFF; // divide system clock by 0xFFFF for SCK
spi_set.config = CFG_MASTER; //master mode
spi_set.error_en = STAT_RXOVERRUN | STAT_TXUNDERRUN; //disable
     // error interrupts - these are only needed in slave mode
```

7. Set-up the DMA_CONFIG_T structure for this DMA transfer (see Section 28.4.8.4):

```
dma_cfg.dma_txd_num = DMAREQ_SPI0_TX;
dma_cfg.dma_rxd_num = DMAREQ_SPI0_RX;
dma_cfg.dma_handle  = (uint32_t)dma_handle;
```

8. Initialize SPI transfer:

```
pSpiApi->spi_init(spi_handle, &spi_set);
```

9. Set up the SPI parameter structure SPI_PARAM_T for the SPI:

```
param.driver_mode = 0x02; //DMA mode.
param.tx_rx_flag  = 0x02; // TX AND RX
param.fsize_sel = TXDATCTL_SSELN(SLAVE0) | TXDATCTL_FSIZE(MASTER_FRAME_SIZE);
param.tx_buffer = (uint16_t *)tx_buffer;
param.rx_buffer = (uint16_t *)rx_buffer;
param.eof_flag  = 1; // after data transfer introduce frame delay.
param.size = 10;  //10 transfers of TXDATCTL_FSIZE size data.
// If receive callback is invoked, transmit follows automatically. Only one
// callback is needed.
param.callback_func_pt = receive_callback;

// DMA set-up

 param.dma_cfg       = &dma_cfg;
 param.dma_req_func_pt = dma_transfer_callback;
```

10. Define the DMA receive callback function invoked when the transmit has finished:

```
ErrorCode_t dma_transfer_callback( SPI_HANDLE_T handle, SPI_DMA_CFG_T *dma_cfg )
{
    DMA_CHANNEL_T chn;
    DMA_TASK_T tsk;
    ErrorCode_t error_code;

    LSPI_REGS_T *lspi = ((LSPI_REGS_T *)((SPI_DRIVER_TypeDef*)handle)->base_addr);
      SPI_DRIVER_TypeDef *driver = (SPI_DRIVER_TypeDef *)handle;

    chn.event = DMA_PER_REQ; //enable peripheral request
    chn.hd_trigger = 0;
    chn.priority = 0;
    tsk.config = DMA_SW_ON | DMA_INT_A; //enable trigger for peripheral request and
           // DMA interrupt
    tsk.data_length = driver->buffer_size - 1;
```

```
    // Enable DMA for SPI RX first. For RX only or TX_RX, only callback_rxd is
     // needed.
    if (driver->callback_rxd != NULL) {
      chn.callback_func_pt = driver->callback_rxd; //set callback function
    }

   tsk.ch_num = dma_cfg->dma_rxd_num;
    tsk.data_type = DMA_8_BIT | DMA_DST_INC_1;
 // peripheral to memory
    tsk.src = (uint32_t)&lspi->RXDAT;
    tsk.dst = (uint32_t)driver->buffer_rxd + tsk.data_length;
    tsk.task_addr = NULL; // for task head, no momery is needed.
    error_code = pDmaApi->dma_init((DMA_HANDLE_T*)dma_cfg->dma_handle, &chn,
 &tsk);
    if ( error_code != LPC_OK )
      return ( error_code );

    tsk.ch_num = dma_cfg->dma_txd_num;
   tsk.data_type = DMA_8_BIT | DMA_SRC_INC_1;
 // memory to peripheral
    tsk.src = (uint32_t)driver->buffer_txd + tsk.data_length;
    tsk.dst = (uint32_t)&lspi->TXDAT;
    tsk.task_addr = NULL; // for task head, no momery is needed.
    error_code = pDmaApi->dma_init((DMA_HANDLE_T*)dma_cfg->dma_handle, &chn,
 &tsk);
    return( error_code ); //init spi dma channel
}
```

11. Define the receive callback function. This function is called when the DMA transfer has finished and sets the EOT bit to one in the SP TXDATCTL register (see Table 198) to set the SSEL signal to HIGH.

```
void  receive_callback( uint32_t err_code, uint32_t n ) {
  LSPI_REGS_T *lspi = ((LSPI_REGS_T
     *)((SPI_DRIVER_TypeDef*)spi_handle)->base_addr);
  if (err_code != LPC_OK)
    while(1);
  // set the EOT flag in the TXCTL register
  if ( lspi->CFG & CFG_MASTER )
    lspi->TXCTL |= TXDATCTL_EOT;
    receive_tag = 1;
}
```

12. DMA transmits data. Slave sends some data back to the DMA.

```
receive_tag = 0;
pSpiApi->spi_master_transfer(spi_handle, &param);
while(!receivetag); //wait for receive tag to be set
```

29.1 How to read this chapter

The I2C-bus ROM API is available on all LPC82x parts.

29.2 Features

- Simple I2C drivers to send and receive data on the I2C-bus.
- Polled and interrupt-driven receive and transmit functions for master and slave modes.

29.3 General description

The drivers are callable for use by any application program to send or receive data on the I2C bus. With the I2C drivers it is easy to produce working projects using the I2C interface.

The ROM routines allow the user to operate the I2C interface as a Master or a Slave. The software routines do not implement arbitration to make a Master switch to a Slave mode in the midst of a transmission.

Although multi-master arbitration is not implemented in these I2C drivers, it is possible to use them in a system design with more than one master. If the flag returned from the driver indicates that the message was not successful due to loss of arbitration, the application just re-sends the message.

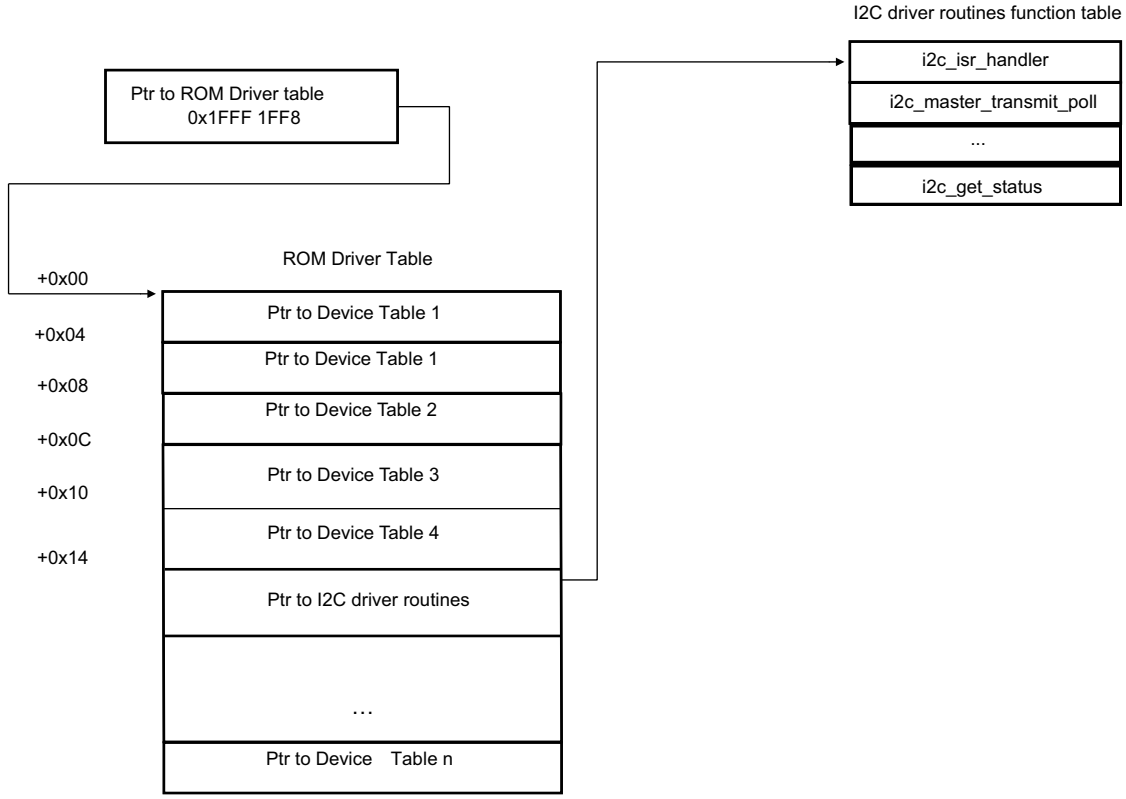


Fig 62. I2C-bus driver routines pointer structure

29.4 API description

The I2C API contains functions to configure the I2C and send and receive data in master and slave modes.

Table 362. I2C API calls

API call	Description	Reference
`void i2c_isr_handler(I2C_HANDLE_T*);`	I2C ROM Driver interrupt service routine.	Table 363
`ErrorCode_t i2c_master_transmit_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT* );`	I2C Master Transmit Polling	Table 364
`ErrorCode_t i2c_master_receive_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);`	I2C Master Receive Polling	Table 365
`ErrorCode_t i2c_master_tx_rx_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);`	I2C Master Transmit and Receive Polling	Table 366
`ErrorCode_t i2c_master_transmit_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);`	I2C Master Transmit Interrupt	Table 367
`ErrorCode_t i2c_master_receive_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);`	I2C Master Receive Interrupt	Table 368
`ErrorCode_t i2c_master_tx_rx_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);`	I2C Master Transmit Receive Interrupt	Table 369

Table 362. I2C API calls

API call	Description	Reference
`ErrorCode_t i2c_slave_receive_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);`	I2C Slave Receive Polling	Table 370
`ErrorCode_t i2c_slave_transmit_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);`	I2C Slave Transmit Polling	Table 371
`ErrorCode_t i2c_slave_receive_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);`	I2C Slave Receive Interrupt	Table 372
`ErrorCode_t i2c_slave_transmit_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*);`	I2C Slave Transmit Interrupt	Table 373
`ErrorCode_t i2c_set_slave_addr(I2C_HANDLE_T*, slave_addr_0_3, slave_mask_0_3);`	I2C Set Slave Address	Table 374
`uint32_t i2c_get_mem_size(void);`	I2C Get Memory Size	Table 375
`I2C_HANDLE_T* i2c_setup(i2c_base_addr, *start_of_ram);`	I2C Setup	Table 376
`ErrorCode_t i2c_set_bitrate(I2C_HANDLE_T*, P_clk_in_hz, bitrate_in_bps);`	I2C Set Bit Rate	Table 377
`uint32_t i2c_get_firmware_version(void );`	I2C Get Firmware Version	Table 378
`I2C_MODE_T i2c_get_status(I2C_HANDLE_T* );`	I2C Get Status	Table 379
`ErrorCode_t i2c_set_timeout(I2C_HANDLE_T* h_i2c, uint32_t timeout);`	I2C time-out value	Table 380

The following structure has to be defined to use the I2C API:

```
typedef struct I2CD_API {   // index of all the i2c driver functions
void (*i2c_isr_handler) (I2C_HANDLE_T* h_i2c) ; // ISR interrupt service request
// MASTER functions ***
ErrorCode_t (*i2c_master_transmit_poll)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
                                                    I2C_RESULT* ptr );
ErrorCode_t (*i2c_master_receive_poll)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
                                                     I2C_RESULT* ptr );
ErrorCode_t (*i2c_master_tx_rx_poll)(I2C_HANDLE_T* h_i2c,I2C_PARAM* ptp,
                                                    I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_master_transmit_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
     I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_master_receive_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
     I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_master_tx_rx_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp, I2C_RESULT*
     ptr ) ;
// SLAVE functions ***
ErrorCode_t (*i2c_slave_receive_poll)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp, I2C_RESULT*
     ptr ) ;
ErrorCode_t (*i2c_slave_transmit_poll)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
     I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_slave_receive_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp, I2C_RESULT*
     ptr ) ;
ErrorCode_t (*i2c_slave_transmit_intr)(I2C_HANDLE_T* h_i2c, I2C_PARAM* ptp,
     I2C_RESULT* ptr ) ;
ErrorCode_t (*i2c_set_slave_addr)(I2C_HANDLE_T* h_i2c,
                                uint32_t slave_addr_0_3, uint32_t slave_mask_0_3);
// OTHER functions
uint32_t (*i2c_get_mem_size)(void) ; //ramsize_in_bytes memory needed by I2C drivers
```

```
I2C_HANDLE_T*   (*i2c_setup)(uint32_t i2c_base_addr, uint32_t *start_of_ram ) ;
ErrorCode_t   (*i2c_set_bitrate)(I2C_HANDLE_T* h_i2c, uint32_t P_clk_in_hz,
                           uint32_t bitrate_in_bps) ;
uint32_t (*i2c_get_firmware_version)() ;
I2C_MODE_T (*i2c_get_status)(I2C_HANDLE_T* h_i2c ) ;
} I2CD_API_T ;


#define ROM_DRIVER_BASE (0x1FFF1FF8UL)
#define LPC_I2CD_API    ((I2CD_API_T *) ((*(ROM_API_T * *) (ROM_DRIVER_BASE))->pI2CD))
```

See Section 3.5.2 for how to include the ROM driver structure.

29.4.1 ISR handler

Table 363. ISR handler

Routine	ISR handler
Prototype	`void i2c_isr_handler(I2C_HANDLE_T*)`
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area.
Return	None.
Description	I2C ROM Driver interrupt service routine. This function must be called from the I2C ISR when using I2C Rom Driver interrupt mode.

29.4.2 I2C Master Transmit Polling

Table 364. I2C Master Transmit Polling

Routine	I2C Master Transmit Polling
Prototype	`ErrorCode_t i2c_master_transmit_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT* )`
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	Transmits bytes in the send buffer to a slave. The slave address with the R/W bit =0 is expected in the first byte of the send buffer. STOP condition is sent at end unless stop_flag =0. When the task is completed, the function returns to the line after the call.

29.4.3 I2C Master Receive Polling

Table 365. I2C Master Receive Polling

Routine	I2C Master Receive Polling
Prototype	`ErrorCode_t i2c_master_receive_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)`

Table 365. I2C Master Receive Polling

Routine	I2C Master Receive Polling
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	Receives bytes from slave and put into receive buffer. The slave address with the R/W bit =0 is expected in the first byte of the send buffer. After the task is finished, the slave address with the R/W bit =1 is in the first byte of the receive buffer. STOP condition is sent at end unless stop_flag =0. When the task is completed, the function returns to the line after the call.

29.4.4 I2C Master Transmit and Receive Polling

Table 366. I2C Master Transmit and Receive Polling

Routine	I2C Master Transmit and Receive Polling
Prototype	`ErrorCode_t i2c_master_tx_rx_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)`
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	First, transmit bytes in the send buffer to a slave and secondly, receives bytes from slave and store it in the receive buffer. The slave address with the R/W bit =0 is expected in the first byte of the send buffer. After the task is finished, the slave address with the R/W bit =1 is in the first byte of the receive buffer. STOP condition is sent at end unless stop_flag =0. When the task is completed, the function returns to the line after the call.

29.4.5 I2C Master Transmit Interrupt

Table 367. I2C Master Transmit Interrupt

Routine	I2C Master Transmit Interrupt
Prototype	`ErrorCode_t i2c_master_transmit_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)`
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	Transmits bytes in the send buffer to a slave. The slave address with the R/W bit =0 is expected in the first byte of the send buffer. STOP condition is sent at end unless stop_flag =0. Program control will be returned immediately and task will be completed on an interrupt-driven basis. When task is completed, the callback function is called.

29.4.6 I2C Master Receive Interrupt

Table 368. I2C Master Receive Interrupt

Routine	I2C Master Receive Interrupt
Prototype	`ErrorCode_t i2c_master_receive_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)`
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	Receives bytes from slave and put into receive buffer. After the task is finished, the slave address with the R/W bit =1 is in the first byte of the receive buffer. STOP condition is sent at end unless stop_flag =0. Program control will be returned immediately and task will be completed on an interrupt-driven basis. When task is completed, the callback function is called.

29.4.7 I2C Master Transmit Receive Interrupt

Table 369. I2C Master Transmit Receive Interrupt

Routine	I2C Master Transmit Receive Interrupt
Prototype	`ErrorCode_t i2c_master_tx_rx_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)`
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	First, transmits bytes in the send buffer to a slave and secondly, receives bytes from slave and store it in the receive buffer. The slave address with the R/W bit =0 is expected in the first byte of the send buffer. After the task is finished, the slave address with the R/W bit =1 is in the first byte of the receive buffer. STOP condition is sent at end unless stop_flag =0. Program control will be returned immediately and task will be completed on an interrupt-driven basis. When task is completed, the callback function is called.

29.4.8 I2C Slave Receive Polling

Table 370. I2C Slave Receive Polling

Routine	I2C Slave Receive Polling
Prototype	`ErrorCode_t i2c_slave_receive_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)`
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	Receives data from master. When the task is completed, the function returns to the line after the call.

29.4.9 I2C Slave Transmit Polling

Table 371. I2C Slave Transmit Polling

Routine	I2C Slave Transmit Polling
Prototype	`ErrorCode_t i2c_slave_transmit_poll(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)`
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	Sends data bytes back to master. When the task is completed, the function returns to the line after the call.

29.4.10 I2C Slave Receive Interrupt

Table 372. I2C Slave Receive Interrupt

Routine	I2C Slave Receive Interrupt
Prototype	`ErrorCode_t i2c_slave_receive_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)`
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	Receives data from master. Program control will be returned immediately and task will be completed on an interrupt-driven basis. When task is completed, the callback function is called.

29.4.11 I2C Slave Transmit Interrupt

Table 373. I2C Slave Transmit Interrupt

Routine	I2C Slave Transmit Interrupt
Prototype	`ErrorCode_t i2c_slave_transmit_intr(I2C_HANDLE_T* , I2C_PARAM* , I2C_RESULT*)`
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. I2C_PARAM - Pointer to the I2C PARAM struct. I2C_RESULT - Pointer to the I2C RESULT struct.
Return	ErrorCode.
Description	Sends data to the Master. Program control will be returned immediately and task will be completed on an interrupt-driven basis. When task is completed, the callback function is called.

29.4.12 I2C Set Slave Address

Table 374. I2C Set Slave Address

Routine	I2C Set Slave Address
Prototype	`ErrorCode_t i2c_set_slave_addr(I2C_HANDLE_T*, slave_addr_0_3, slave_mask_0_3)`
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. Slave_addr_0_3 - unint32 variable. 7-bit slave address . Slave_mask_0_3 - unint32 variable. Slave address mask.
Return	ErrorCode.
Description	Sets the slave address and associated mask. The set_slave_addr() function supports four 7-bit slave addresses and masks.

29.4.13 I2C Get Memory Size

Table 375. I2C Get Memory Size

Routine	I2C Get Memory Size
Prototype	`uint32_t i2c_get_mem_size(void)`
Input parameter	None.
Return	uint32.
Description	Returns the number of bytes in SRAM needed by the I2C driver.

29.4.14 I2C Setup

Table 376. I2C Setup

Routine	I2C Setup
Prototype	`I2C_HANDLE_T* i2c_setup(i2c_base_addr, *start_of_ram)`
Input parameter	I2C_base addr - unint32 variable. Base address for I2C peripherals. Start_of_ram - unint32 pointer. Pointer to allocated SRAM.
Return	I2C_Handle.
Description	Returns a handle to the allocated SRAM area.

29.4.15 I2C Set Bit Rate

Table 377. I2C Set Bit Rate

Routine	I2C Set Bit Rate
Prototype	`ErrorCode_t i2c_set_bitrate(I2C_HANDLE_T*, P_clk_in_hz, bitrate_in_bps)`
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. P_clk_in_hz - unint32 variable. The Peripheral Clock in Hz. Bitrate_in_bps - unint32 variable. Requested I2C operating frequency in Hz.
Return	ErrorCode.
Description	Configures the I2C duty-cycle registers (SCLH and SCLL).

29.4.16 I2C Get Firmware Version

Table 378. I2C Get Firmware Version

Routine	I2C Get Firmware Version
Prototype	`uint32_t i2c_get_firmware_version(void )`
Input parameter	None.
Return	I2C ROM Driver version number.
Description	Returns the version number. The firmware version is an unsigned 32-bit number.

29.4.17 I2C Get Status

Table 379. I2C Get Status

Routine	I2C Get Status
Prototype	`I2C_MODE_T i2c_get_status(I2C_HANDLE_T* )`
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area.
Return	Status code.
Description	Returns status code. The status code indicates the state of the I2C bus. Refer to I2C Status Code Table.

29.4.18 I2C time-out value

Table 380. I2C time-out value

Routine	I2C time-out value
Prototype	`ErrorCode_t i2c_set_timeout(I2C_HANDLE_T* h_i2c, uint32_t timeout)`
Input parameter	I2C_HANDLE_T - Handle to the allocated SRAM area. uint32_t timeout - time value is timeout*16 i2c function clock. If timeout = 0, timeout feature is disabled.
Return	Status code.
Description	Returns status code. The status code indicates the state of the I2C bus. Refer to I2C Status Code Table.

29.4.19 Error codes

Table 381. Error codes

Error Code	Description	Comment
0	Successful completion	Function was completed successfully.
1	General error	-
0x0006 0001	ERR_I2C_NAK	-
0x0006 0002	ERR_I2C_BUFFER_OVERFLOW	-
0x0006 0003	ERR_I2C_BYTE_COUNT_ERR	-
0x0006 0004	ERR_I2C_LOSS_OF_ARBRITRATION	-
0x0006 0005	ERR_I2C_SLAVE_NOT_ADDRESSED	-

Table 381. Error codes

Error Code	Description	Comment
0x0006 0006	ERR_I2C_LOSS_OF_ARBRITRATION_NAK_BIT	-
0x0006 0007	ERR_I2C_GENERAL_FAILURE	Failure detected on I2C bus.
0x0006 0008	ERR_I2C_REGS_SET_TO_DEFAULT	I2C clock frequency could not be set. Default value of 0x04 is loaded into SCLH and SCLL.

29.4.20 I2C Status code

Table 382. I2C Status code

Status code	Description
0	IDLE
1	MASTER_SEND
2	MASTER_RECEIVE
3	SLAVE_SEND
4	SLAVE_RECEIVE

29.4.21 I2C ROM driver variables

The I2C ROM driver requires specific variables to be declared and initialized for proper usage. Depending on the operating mode, some variables can be omitted.

29.4.21.1 I2C Handle

The I2C handle is a pointer allocated for the I2C ROM driver. The handle needs to be defined as an I2C handle TYPE:

```
typedef void* I2C_HANDLE_T
```

After the definition of the handle, the handle must be initialized with I2C base address and RAM reserved for the I2C ROM driver by making a call to the i2c_setup() function.

The callback function type must be defined if interrupts for the I2C ROM driver are used:

```
typedef void (*I2C_CALLBK_T) (uint32_t err_code, uint32_t n)
```

The callback function will be called by the I2C ROM driver upon completion of a task when interrupts are used.

29.4.22 PARAM and RESULT structure

The I2C ROM driver input parameters consist of two structures, a PARAM structure and a RESULT structure. The PARAM structure contains the parameters passed to the I2C ROM driver and the RESULT structure contains the results after the I2C ROM driver is called.

The PARAM structure is as follows:

```
typedef struct i2c_A { //parameters passed to ROM function
     uint32_t num_bytes_send ;
     uint32_t num_bytes_rec ;
     uint8_t *buffer_ptr_send ;
     uint8_t *buffer_ptr_rec ;
```

```
     I2C_CALLBK_T func_pt; // callback function pointer
     uint8_t stop_flag;
     uint8_t dummy[3] ;         // required for word alignment
} I2C_PARAM ;
```

The RESULT structure is as follows:

```
typedef struct i2c_R {     // RESULTs struct--results are here when returned
     uint32_t n_bytes_sent ;
     uint32_t n_bytes_recd ;
} I2C_RESULT ;
```

29.4.23 Error structure

The error code returned by the I2C ROM driver is an enum structure. The Error structure is as follows:

```
typedef enum
{
                LPC_OK=0, /**< enum value returned on Success */
                 ERROR,
 ERR_I2C_BASE = 0x00060000,
/*0x00060001*/ ERR_I2C_NAK=ERR_I2C_BASE+1,
/*0x00060002*/ ERR_I2C_BUFFER_OVERFLOW,
/*0x00060003*/ ERR_I2C_BYTE_COUNT_ERR,
/*0x00060004*/ ERR_I2C_LOSS_OF_ARBRITRATION,
/*0x00060005*/ ERR_I2C_SLAVE_NOT_ADDRESSED,
/*0x00060006*/ ERR_I2C_LOSS_OF_ARBRITRATION_NAK_BIT,
/*0x00060007*/ ERR_I2C_GENERAL_FAILURE,
/*0x00060008*/ ERR_I2C_REGS_SET_TO_DEFAULT
} ErrorCode_t;
```

29.4.24 I2C Mode

The i2c_get_status() function returns the current status of the I2C engine. The return codes can be defined as an enum structure:

```
typedef enum I2C_mode {
 IDLE,
 MASTER_SEND,
 MASTER_RECEIVE,
 SLAVE_SEND,
 SLAVE_RECEIVE
 } I2C_MODE_T ;
```

29.5 Functional description

29.5.1 I2C Set-up

Before calling any setup functions in the I2C ROM, the application program is responsible for doing the following:

1. Enable the clock to the I2C peripheral.

2. Enable the two pins required for the SCL and SDA outputs of the I2C peripheral.
3. Allocate a RAM area for dedicated use of the I2C ROM Driver.

After the I2C block is configured, the I2C ROM driver variables have to be set up:

1. Initialize pointer to the I2C API function table.
2. Declare the PARAM and RESULT struct.
3. Declare Error Code struct.
4. Declare the transmit and receive buffer.

If interrupts are used, then additional driver variables have to be set up:

1. Declare the I2C_CALLBK_T type.
2. Declare callback functions.
3. Declare I2C ROM Driver ISR within the I2C ISR.
4. Enable I2C interrupt.

29.5.2 I2C Master mode set-up

The I2C ROM Driver support polling and interrupts. In the master mode, 7-bit and 10-bit addressing are supported. The setup is as follows:

1. Allocate SRAM for the I2C ROM Driver by making a call to the i2c_get_mem_size() function.
2. Create the I2C handle by making a call to the i2c_setup() function.
3. Set the I2C operating frequency by making a call to the i2c_set_bitrate() function.

```
size_in_bytes = LPC_I2CD_API->i2c_get_mem_size();
i2c_handle = LPC_I2CD_API->i2c_setup(LPC_I2C_BASE, (uint32_t *)&I2C_Handle[0] );
error_code = LPC_I2CD_API->i2c_set_bitrate((I2C_HANDLE_T*)i2c_handle, PCLK_in_Hz,
     bps_in_hz);
```

29.5.3 I2C Slave mode set-up

The I2C ROM Driver support polling and interrupts in the slave mode. In the slave mode, only 7-bit addressing is supported. The set-up is as follows:

1. Allocate SRAM for the I2C ROM Driver by making a call to the i2c_get_mem_size() function.
2. Create the I2C handle by making a call to the i2c_setup() function.
3. Set the I2C operating frequency by making a call to the i2c_set_bitrate() function.
4. Set the slave address by making a call to the i2c_set_slave_addr() function.

The I2C ROM driver allows setting up to 4 slave addresses and 4 address masks as well as possibly enabling the General Call address.

The four slave address bytes are packed into the 4 byte variable. Slave address byte 0 is the least significant byte and Slave address byte 3 is the most significant byte. The Slave address mask bytes are ordered the same way in the other 32 bit variable. When in slave

receive mode, all of these addresses (or groups if masks are used) will be monitored for a match. If the General Call bit (least significant bit of any of the four slave address bytes) is set, then the General Call address of 0x00 is monitored as well.

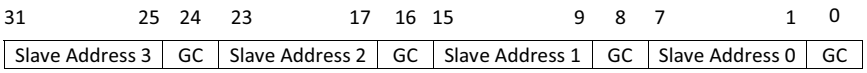


Fig 63. I2C slave mode set-up address packing

```
size_in_bytes = LPC_I2CD_API->i2c_get_mem_size();
i2c_handle = LPC_I2CD_API->i2c_setup(LPC_I2C_BASE, (uint32_t *)&I2C_Handle[0] );
error_code = LPC_I2CD_API->i2c_set_bitrate((I2C_HANDLE_T*)i2c_handle, PCLK_in_Hz,
     bps_in_hz);
error_code = LPC_I2CD_API->i2c_set_slave_addr((I2C_HANDLE_T*)i2c_handle, slave_addr,
     slave_addr_mask) ;
```

29.5.4 I2C Master Transmit/Receive

The Master mode drivers give the user the choice of either polled (wait for the message to finish) or interrupt driven routines (non-blocking). Polled routines are recommended for testing purposes or very simple I2C applications. These routines allow the Master to send to Slaves with 7-bit or 10-bit addresses.

The following routines are polled routines:

```
err_code i2c_master_transmit_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

err_code i2c_master_receive_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

err_code i2c_master_tx_rx_poll (I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
```

The following routines are interrupt driven routines:

```
err_code i2c_master_transmit_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

err_code i2c_master_receive_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

err_code i2c_master_tx_rx_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
```

Where:

- err_code is the return state of the function. An “0” indicates success. All non-zero indicates an error. Refer to Error Table.
- I2C_PARM* is a structure with parameters passed to the function. Refer to Section 29.4.22.
- I2C_RESULT* is a containing the results after the function executes.

To initiate a master mode write/read the I2C_PARAM has to be setup. The I2C_PARAM is a structure with various variables needed by the I2C ROM Driver to operate correctly. The structure contains the following:

- Number of bytes to be transmitted.
- Number of bytes to be receive.
- Pointer to the transmit buffer.
- Pointer to the receive buffer.
- Pointer to callback function.
- Stop flag.

The RESULT structure contains the results after the function executes. The structure contains the following:

- Number of bytes transmitted.
- Number of bytes received.

Remark: The number of bytes transmitted will be updated for i2c_master_transmit_intr() and i2c_master_transmit_poll(). The number of bytes received will only be update on i2c_master_receive_poll(), i2c_master_receive_intr(), i2c_master_tx_rx_poll(), and i2c_master_tx_rx_intr().

In all the master mode routines, the transmit buffer's first byte must be the slave address with the R/W bit set to "0". To enable a master read, the receive buffer's first byte must be the slave address with the R/W bit set to "1".

The following conditions must be fulfilled to use the I2C driver routines in master mode:

- For 7-bit addressing, the first byte of the send buffer must have the slave address in the most significant 7 bits and the least significant (R/W) bit = 0. Example: Slave address 0x53, first byte is 0xA6.
- For 7-bit addressing, the first byte of the receive buffer must have the slave address in the most significant 7 bits and the least significant (R/W) bit = 1. Example: Slave Addr 0x53, first byte 0xA7.
- For 10-bit address, the first byte of the transmit buffer must have the slave address most significant 2 bits with the (R/W) bit =0. The second byte must contain the remaining 8-bit of the slave address.
- For 10-bit address, the first byte of the receive buffer must have the slave address most significant 2 bits with the (R/W) bit =1. The second byte must contain the remaining 8-bit of the slave address.
- The number of bytes to be transmitted should include the first byte of the buffer which is the slave address byte. Example: 2 data bytes + 7-bit slave addr = 3.
- The application program must enable I2C interrupts. When I2C interrupt occurs, the i2c_isr_handler function must be called from the application program.

When using the interrupt function calls, the callback functions must be define. Upon the completion of a read/write as specified by the PARAM structure, the callback functions will be invoked.

29.5.5 I2C Slave Mode Transmit/Receive

In slave mode, polled routines are intended for testing purposes. It is up to the user to decide whether to use the polled or interrupt driven mode. While operating the Slave driver in polled mode can be useful for program development and debugging, most applications will need the interrupt-driven versions of Slave Receive and Transmit in the final software.

The following routines are polled routines:

```
err_code i2c_slave_receive_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

err_code i2c_slave_transmit_poll(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
```

The following routines are interrupt driven routines:

```
err_code i2c_slave_receive_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)

err_code i2c_slave_transmit_intr(I2C_HANDLE_T*, I2C_PARAM*, I2C_RESULT*)
```

Where:

- err_code is the return state of the function. An 0 indicates success. All non-zero indicates an error. Refer to the Error Code Table.
- I2C_PARM is a structure with parameters passed to the function. Section 29.4.22.
- I2C_RESULT is a containing the results after the function executes. Section 29.4.22.

To initiate a master-mode write/read the I2C_PARAM has to be setup. The I2C_PARAM is a structure with various variables needed by the I2C ROM Driver to operate correctly. The structure contains the following:

- Number of bytes to be transmitted.
- Number of bytes to be received.
- Pointer to the transmit buffer.
- Pointer to the receive buffer.
- Pointer to callback function.
- Stop flag.

The RESULT structure contains the results after the function executes. The structure contains the following:

- Number of bytes transmitted.
- Number of bytes received.

Remark: The number of bytes transmitted is updated only for i2c_slave_send_poll() and i2c_slave_send_intr(). The number of bytes received is updated only for i2c_slave_receive_poll() and i2c_slave_receive_intr().

To initiate a slave mode communication, the receive function is called. This can be either the polling or interrupt driven function, i2c_slave_receive_poll() or i2c_slave_receive_intr(), respectively. The receive buffer should be as large or larger than any data or command that will be received. If the amount of data exceed the receive buffer size, an error code will be returned.

In slave-receive mode, the driver receives data until one of the following are true:

- Address matching set in the set_slave_addr() function with the R/W bit set to 1
- STOP or repeated START is received
- An error condition is detected

When using the interrupt function calls, the callback functions must be define. Upon the completion of a read/write as specified by the PARAM structure, the callback functions will be invoked.

29.5.6 I2C time-out feature

```
//timeout: Timeout time value. Specifies the timeout interval value in increments of
// 16 I2C function clocks (Min value is 16).
//                    if timeout = 0,  timeout feature is disabled
//         if timeout != 0, time value is timeout*16 i2c function clock.
ErrorCode_t i2c_set_timeout(I2C_HANDLE_T*  h_i2c, uint32_t timeout)
{
  I2C_DRIVER_TypeDef *h ;  // declare pointer to i2c structure [handle]
  h = (I2C_DRIVER_TypeDef*) h_i2c ;   //assign handle pointer address
     if (timeout != 0){
          h->i2c_base->TimeOut = (timeout - 1)<<4;
          // Enable timeout feature
          h->i2c_base->CFG |= BI2C_TIMEOUT_EN;
     }
     else
          // disable timeout feature
          h->i2c_base->CFG &= ~BI2C_TIMEOUT_EN;

     return(LPC_OK) ;
}//i2c_set_timeout
```

30.1 How to read this chapter

The ADC ROM driver routines are available on all parts.

30.2 Features

- ADC calibration
- Triggered ADC conversions
- ADC interrupt handling

30.3 General description

The ADC API handles the calibration and set-up of the ADC and allows the user to perform analog-to-digital conversion using the 12-bit ADC.

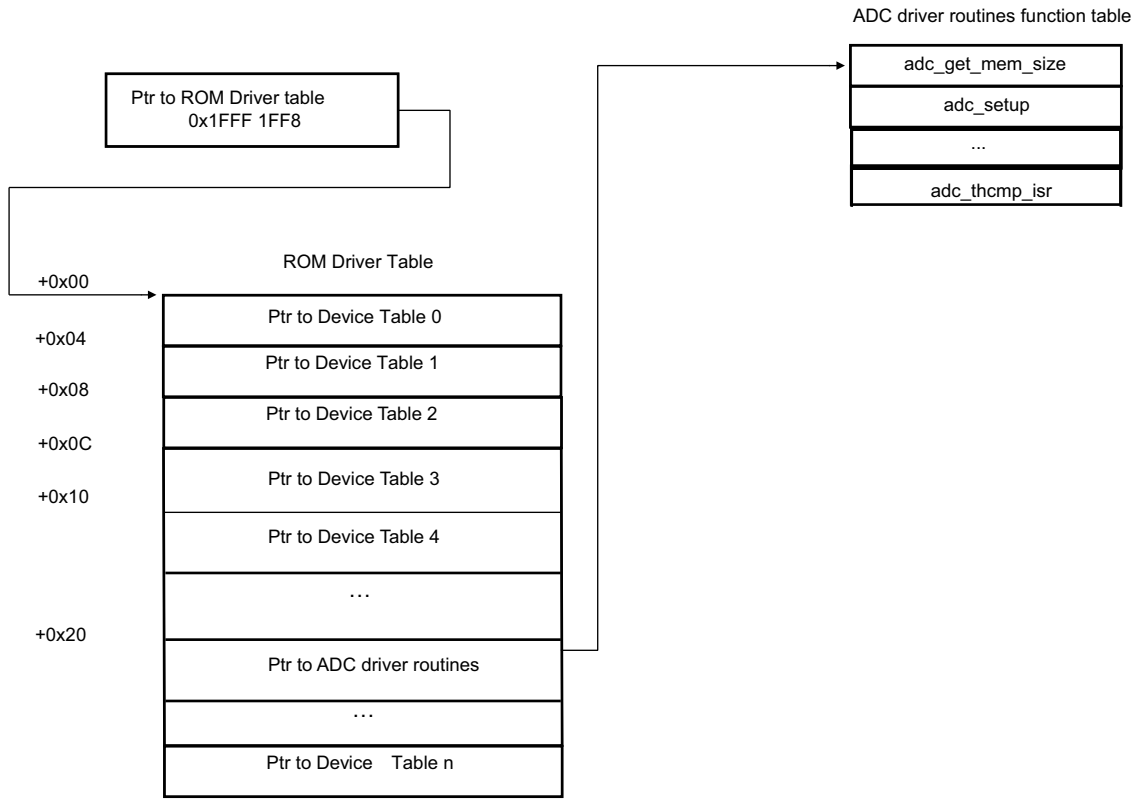


Fig 64. ADC driver routines pointer structure

30.4 API description

The ADC API contains functions set up and operate the 12-bit ADC.

Table 383. ADC API calls

API call	Description	Reference
ADC set-up		
`uint32_t ramsize_in_bytes adc_get_mem_size(void);`	Get memory size for one ADC instance	Table 384
`ADC_HANDLE_T* adc_setup (uint32_t base_addr, uint8_t *ram);`	Set-up ADC instance	Table 385
`void adc_calibration (ADC_HANDLE_T handle, ADC_CONFIG_T *adc_set);`	Calibrate ADC	Table 386
`void adc_init (ADC_HANDLE_T *handle, ADC_CONFIG_T adc_set);`	Set-up operation mode and enable ADC	Table 387
ADC operation		
`uint32_t adc_seqa_read (ADC_HANDLE_T* handle, ADC_PARAM_T param);`	Sequence A conversion	Table 388
`uint32_t adc_seqb_read (ADC_HANDLE_T* handle, ADC_PARAM_T param);`	Sequence B conversion	Table 389
Interrupt service		
`void adc_seqa_isr (ADC_HANDLE_T* handle);`	Sequence A interrupt service	Table 390
`void adc_seqb_isr (ADC_HANDLE_T* handle);`	Sequence B interrupt service	Table 391
`void adc_ovr_isr (ADC_HANDLE_T* handle);`	Overrun interrupt service	Table 392
`void adc_thcmp_isr (ADC_HANDLE_T* handle);`	Threshold compare interrupt service	Table 393

The following structure must be defined to use the ADC API:

```
typedef struct  ADCD_API {   // index of ADC driver functions
     uint32_t (*adc_get_mem_size)(void);
     ADC_HANDLE_T (*adc_setup)(uint32_t base_addr, uint8_t *ram);
     void (*adc_calibration)(ADC_HANDLE_T handle, ADC_CONFIG_T *set);
     void (*adc_init)(ADC_HANDLE_T handle, ADC_CONFIG_T *set);
     uint32_t (*adc_seqa_read)(ADC_HANDLE_T handle, ADC_PARAM_T * param);
     uint32_t (*adc_seqb_read)(ADC_HANDLE_T handle, ADC_PARAM_T * param);
     //--interrupt functions--//
     void (*adc_seqa_isr)(ADC_HANDLE_T handle);
     void (*adc_seqb_isr)(ADC_HANDLE_T handle);
     void (*adc_ovr_isr)(ADC_HANDLE_T handle);
     void (*adc_thcmp_isr)(ADC_HANDLE_T handle);
} ADCD_API_T  ;
```

30.4.1 ADC get memory size

Table 384. adc_get_mem_size

Routine	adc_get_mem_size
Prototype	`uint32_t ramsize_in_bytes adc_get_mem_size(void);`
Input parameter	None.
Return	Memory size in bytes.
Description	The memory size for one ADC instance.

30.4.2 ADC set-up

Table 385. adc_setup

Routine	adc_setup
Prototype	`ADC_HANDLE_T* adc_setup (uint32_t base_addr, uint8_t *ram);`
Input parameter	base_addr: Base address of register for the ADC block. ram: Pointer to the memory space for ADC instance; the size is obtained from adc_get_mem_size() function.
Return	The handle to corresponding ADC instance.
Description	Sets up the ADC instance with provided memory and the base address of the ADC instance, and returns the handle to this instance.

30.4.3 ADC calibration

The conditions under which self-calibration of the ADC should be performed are described in Section 21.3.4 “Hardware self-calibration”.

Remark: Once the calibration is complete, revert the calibration-related clock settings in the ADC_CONFIG_T structure to the clock settings needed for A/D conversion.

Table 386. adc_calibration

Routine	adc_calibration
Prototype	`void adc_calibration (ADC_HANDLE_T handle, ADC_CONFIG_T *adc_set);`
Input parameter	handle: The handler of ADC driver from adc_setup(). adc_set: Configuration for ADC operation.
Return	None.
Description	Calibrate ADC controller. Setup the ADC_CONFIG_T structure with the parameters `system_clock` and `adc_clock`. Only the system clock and the ADC clock are needed for ADC calibration. Set the ADC clock to about 500 kHz for calibration and ensure that the system clock is larger or equal to the ADC clock,

30.4.4 ADC initialize

The ADC_CONFIG_T structure defines the configuration settings for this function. See Section 30.4.12.1 “ADC_CONFIG_T channel configuration structure”.

Table 387. adc_init

Routine	adc_init
Prototype	`void adc_init (ADC_HANDLE_T *handle, ADC_CONFIG_T adc_set);`
Input parameter	handle: The handle to the ADC instance. adc_set: Configuration for ADC operation.
Return	None.
Description	Setup operation mode for ADC, then enable ADC.

30.4.5 ADC start sequence A conversion

Table 388. adc_seqa_read

Routine	adc_seqa_read
Prototype	`uint32_t adc_seqa_read (ADC_HANDLE_T* handle, ADC_PARAM_T param);`
Input parameter	handle: The handle to the ADC instance. param: Pointer to the ADC_PARAM_T structure.
Return	Error code.
Description	Start ADC conversion using sequence A.

30.4.6 ADC start sequence B conversion

Table 389. adc_seqb_read

Routine	adc_seqb_read
Prototype	`uint32_t adc_seqb_read (ADC_HANDLE_T* handle, ADC_PARAM_T param);`
Input parameter	handle: The handle to the ADC instance. param: Pointer to the ADC_PARAM_T structure.
Return	Error code.
Description	Start ADC conversion using sequence B.

30.4.7 ADC service sequence A interrupt

Table 390. adc_seqa_isr

Routine	adc_seqa_isr
Prototype	`void adc_seqa_isr (ADC_HANDLE_T* handle);`
Input parameter	handle: The handle to the ADC instance.
Return	None.
Description	ADC interrupt service routine for sequence A interrupt. When using this function, the corresponding ADC interrupt must be enabled. This function is invoked by the user ISR.

30.4.8 ADC service sequence B interrupt

Table 391. adc_seqb_isr

Routine	adc_seqb_isr
Prototype	`void adc_seqb_isr (ADC_HANDLE_T* handle);`
Input parameter	handle: The handle to the ADC instance.
Return	None.
Description	ADC interrupt service routine for sequence B interrupt. When using this function, the corresponding ADC interrupt must be enabled. This function is invoked by the user ISR.

30.4.9 ADC service overrun error interrupt

Table 392. adc_ovr_isr

Routine	adc_ovr_isr
Prototype	`void adc_ovr_isr (ADC_HANDLE_T* handle);`
Input parameter	handle: The handle to the ADC instance.
Return	None.
Description	ADC interrupt service routine for overrun error interrupt. When using this function, the corresponding ADC interrupt must be enabled. This function is invoked by the user ISR.

30.4.10 ADC service threshold compare interrupt

Table 393. adc_thcmp_isr

Routine	adc_thcmp_isr
Prototype	`void adc_thcmp_isr (ADC_HANDLE_T* handle);`
Input parameter	handle: The handle to the ADC instance.
Return	None.
Description	ADC interrupt service routine for threshold compare interrupt. When using this function, the corresponding ADC interrupt must be enabled. This function is invoked by the user ISR.

30.4.11 Error codes

Table 394. Error codes

Return code	Error Code	Description
0x000F 0001	ERR_ADC_OVERRUN	
0x000F 0002	ERR_ADC_INVALID_CHANNEL	
0x000F 0003	ERR_ADC_INVALID_SEQUENCE	
0x000F 0004	ERR_ADC_INVALID_SETUP	
0x000F 0005	ERR_ADC_PARAM	
0x000F 0006	ERR_ADC_INVALID_LENGTH	
0x000F 0007	ERR_ADC_NO_POWER	

```
typedef enum
{
 ERR_ADC_BASE = 0x000F0000,
  /*0x000F0001*/ ERR_ADC_OVERRUN=ERR_ADC_BASE+1,
  /*0x000F0002*/ ERR_ADC_INVALID_CHANNEL,
  /*0x000F0003*/ ERR_ADC_INVALID_SEQUENCE,
  /*0x000F0004*/ ERR_ADC_INVALID_SETUP,
  /*0x000F0005*/ ERR_ADC_PARAM,
  /*0x000F0006*/ ERR_ADC_INVALID_LENGTH,
  /*0x000F0007*/ ERR_ADC_NO_POWER
} ErrorCode_t;
```

30.4.12 ADC ROM driver variables

30.4.12.1 ADC_CONFIG_T channel configuration structure

```
typedef  struct {
     uint32_t system_clock; // System clock
     uint32_t adc_clock; // ADC clock (set system_clock >= adc_clock)
     uint32_t async_mode;
     uint32_t tenbit_mode;
     uint32_t lpwr_mode;
     uint32_t input_sel;
     uint32_t seqa_ctrl;
     uint32_t seqb_ctrl;
     uint32_t thrsel;
     uint32_t thr0_low;
     uint32_t thr0_high;
     uint32_t thr1_low;
     uint32_t thr1_high;
     uint32_t error_en;
     uint32_t thcmp_en;
     uint32_t channel_num;
} ADC_CONFIG_T;
```

The variables in this structure are defined as follows:

`system_clock`: Frequency of the system clock generated by the SYSCON block in Hz.

`adc_clock`: Frequency of the ADC clock in Hz. This is the clock rate for analog-to-digital conversions. Maximum clock rate is 50 MHz for 12-bit resolution. In synchronous mode, ensure that `system_clock` >= `adc_clock`.

`async_mode`: 0 selects synchronous mode, 1 selects asynchronous mode. See Table 280.

`tenbit_mode`: 0 disables 10-bit mode, 1 enables fast-conversion, 10-bit mode. See Table 280.

`lpwr_mode`: 0 disables low-power mode, 1 enables low-power mode. See Table 280.

`inp_sel`: selects input for channel 0.

0x0 = ADCn_0 pin.

0x1 = Reserved.

0x2 = Reserved.

0x3 = Reserved.

0x4 =Reserved.

`seqa_ctrl`: Register content of the SEQA_CTRL register. See Table . Ensure that the START and SEQA_EN bits are always set to 0.

`seqb_ctrl`: Register content of the SEQB_CTRL register. See Table . Ensure that the START and SEQB_EN bits are always set to 0.

`thrsel`: Register content of the CHAN_THRSEL register. See Table 288.

`thr0_low`: Low threshold 0 value.

`thr0_high`: High threshold 0 value.

`thr1_low`: Low threshold 1 value.

thr1_high: High threshold 1 value.

error_en: 0 disables the overrun interrupt, 1 enables the overrun interrupt.

thcmp_en: Value of bits 26:3 in the INTEN register. See Table 289. Each pair of bits controls the threshold comparison interrupt for one selected channel.

channel_num: The highest channel number used in sequence A or sequence B. If this number is lower than the total available number of ADC channels, only the first channel_num channels can be used in either sequence A or sequence B.

30.4.12.2 ADC_HANDLE_T

This structure is the handle to one instance of the ADC driver. Each ADC has one handle. This handle is created by the init API.

```
typedef  void     ADC_HANDLE_T   ;    // define TYPE for ADC handle pointer
```

30.4.12.3 ADC_DMA_CFG_T

This structure sets up the DMA channel used for DMA transfer. The transfer can use a hardware trigger which is selected in the DMA_ITRIG_INMUX register for the selected channel.

See Table 147 for the implemented hardware triggers and the trigger numbers.

The DMA transfer is configured using the DMA API. After setting up the DMA transfer, a handle is returned and passed to the ADC in this structure.

```
typedef struct{
     uint32_t dma_adc_num;// DMA channel used for ADC data peripheral to memory
                          //transfer
     uint32_t dma_pinmux_num;// H/W trigger number.
     uint32_t dma_handle;// DMA handle passed to ADC
     ADC_CALLBK_T dma_done_callback_pt; // DMA completion callback function
} ADC_DMA_CFG_T;
```

30.4.12.4 ADC_PARAM_T

It's important that some of the parameters in ADC_CONFIG_T such as sequence control register setting and channel numbers needs to be set up before ADC_PARAM_T can be passed to the SEQA and SEQB read routines.

```
typedef struct {// params passed to adc driver function
     uint32_t          *buffer;
     // Considering supporting DMA and non-DMA mode, 32-bit buffer is needed for DMA
     uint32_t          driver_mode;
     // 0x00: Polling mode, function is blocked until transfer is finished.
     // 0x01: Interrupt mode, function exit immediately, callback function is invoked
     // when transfer is finished.
     // 0x02: DMA mode, in case DMA block is available, data transferred by ADC is
     // processed by DMA,
     //       and max buffer size is the
     //total number ADC channels, DMA req function is called for ADC DMA
     //       channel setup, then SEQx
     //completion also used as DMA callback function when that ADC conversion/DMA
     //transfer is finished.
```

```
    uint32_t          seqa_hwtrig;// H/W trigger for sequence A
    uint32_t          seqb_hwtrig;// H/W trigger for sequence B
    ADC_CONFIG_T      *adc_cfg;
    uint32_t          comp_flags;
    uint32_t          overrun_flags;
    uint32_t          thcmp_flags;
    ADC_DMA_CFG_T     *dma_cfg;
    ADC_SEQ_CALLBK_T  seqa_callback_pt;// SEQA callback function/the same callback
    //on DMA completion if DMA is used for ADCx.
    ADC_SEQ_CALLBK_T  seqb_callback_pt;// SEQb callback function/the same callback
    //on DMA completion if DMA is used for ADCx.
    ADC_CALLBK_T      overrun_callback_pt;// Overrun callback function
    ADC_CALLBK_T      thcmp_callback_pt;// THCMP callback function
    ADC_DMA_SETUP_T   dma_setup_func_pt;// ADC DMA channel setup function
} ADC_PARAM_T;
```

The following variables and pointers are used in this structure:

`seqa_buffer`: ADC buffer for the result of the conversion. Two separate buffers are allocated for both SEQA and SEQB. To support DMA mode, the ADC buffer size is set to 32-bit. Once DMA is completed, only bits 4 to 15 of the buffer contain the result of the ADC conversion. For non-DMA mode, the buffers contain the actual data from the conversion.

`seqb_buffer`: See description of `seqa_buffer`.

`driver_mode`: Configures the API function:

0x00: Polling mode, function is blocked until transfer is finished.

0x01: Interrupt mode. Function exits immediately and callback function is invoked when ADC conversion is finished.

0x02: DMA mode. ADC data conversion is handled by DMA. The maximum DMA buffer size is the total number ADC channels. The DMA API function is called for ADC DMA channel setup.

`seqa_hwtrig`: Select the hardware trigger for sequence A conversion. See Section 21.3.3 “ADC hardware trigger inputs” for potential triggers sources.

`seqb_hwtrig`: Select the hardware trigger for sequence A conversion. See Section 21.3.3 “ADC hardware trigger inputs” for potential triggers sources.

`comp_flags`: For each channel with a completed conversion, the corresponding bit is set.

`thcmp_flags`: For each channel incurring an overrun or threshold compare interrupt, the corresponding bit is set.

30.4.12.5 Callback functions

```
typedef void  (*ADC_SEQ_CALLBK_T) (ADC_HANDLE_T handle);
```

`handle`:The handle to the ADC instance.

The following callback function is invoked in the ADC sequence interrupt handler:

```
typedef void  (*ADC_CALLBK_T) (ErrorCode_t error_code, uint32_t num_channel );
```

`error_code`: ADC error code

num_channel: In interrupt mode, number of ADC channels that have been converted.

Depending on the driver mode, the ADC_CALLBAK_T is either invoked in the ADC overrun, threshold compare interrupt handler, or DMA interrupt handler.

In the overrun and threshold compare interrupt mode, the error code and the number of ADC channels are passed to the callback function. In DMA mode, the callback functions indicates the completion of the DMA transfer.

30.4.12.6 ADC_DMA_SETUP_T

This function defines the ADC DMA channel set-up function type.

```
typedef ErrorCode_t  (*ADC_DMA_SETUP_T) (ADC_HANDLE_T handle, ADC_DMA_CFG_T *dma_cfg);
```

When the driver mode is DMA, this DMA set-up callback is invoked.

To set up the DMA channel, the source address, destination address, and the DMA transfer length, the DMA request information must be retrieved from the driver structure, which is originally passed from ADC_PARAM_T structure.

30.5 Functional description

30.5.1 Example

Perform a simple analog-to-digital conversion with the ADC0 in interrupt mode (sequence A done) using a hardware trigger (ADC0_PINTRIG0). Triggering the sequence once starts the conversion. All 12 channels are sampled. Use the ADC API as follows:

Remark: For this example, burst mode must be disabled.

1. Assign the ADC0_PINTRIG0 function to a pin in the switch matrix.
2. Global defines:

```
#define rom_drivers_ptr (* (ROM **) 0x03000200)
ADCD_API_T *  pAdcApi ;  //define pointer to type API function addr table
ADC_HANDLE_T*  adc_handle_0; //handle to ADC0 API

ADC_PARAM_T        param;
ADC_CONFIG_T       adc_cfg;

#define RAMBLOCK_H  60
uint32_t  start_of_ram_block0[ RAMBLOCK_H ] ;

#define BUFFER_SIZE   100
uint32_t adc_buffer[BUFFER_SIZE];
```

3. Initialize pointer to the ADC API:

```
 pAdcApi = (ADCD_API_T *) (rom_drivers_ptr->pADCD);
```

4. Initialize ADC0:

```
size_in_bytes = pAdcApi->adc_get_mem_size() ; //size_in_bytes/4 must be
 // < RAMBLOCK_H
adc_handle_0 = pAdcApi->adc_setup(LPC_ADC0_BASE, (uint8_ *)start_of_ram_block0);
```

5. Set up the ADC0 for calibration by defining the ADC_CONFIG_T structure:

```
ADC_CONFIG_T adc_set;
adc_set.system_clock = SYSTEMCLOCK; //system clock
adc_set.adc_clock = ADC_CALIB_CLK; // ADC clock
```

6. Always calibrate the ADC before performing conversions:

```
pAdcApi->adc_calibration(adc_handle_0, &adc_set);
```

7. Set up the ADC0 for interrupt mode using the hardware trigger pin ADC0_PINTRIG0 by defining the ADC_CONFIG_T structure:

```
adc_set.adc_clock = ADC_ADC_CLK; // this is the ADC clock - in synchronous mode
      //equal to the system clock divided by the CLKDIV value in the CTRL register
     // in synchronous mode set system_clock >= adc_clock.
adc_set.async_mode = 0; //async mode disabled
adc_set.tenbit_mode = 0; //10-bit mode disabled
adc_set.lpwr_mode = 0; //low-power mode disabled
adc_set.thr0_low = 0; //threshold disabled
adc_set.thr0_high = 0; //threshold disabled
adc_set.thr1_low = 0; //threshold disabled
adc_set.thr1_high = 0; //threshold disabled
adc_set.error_en = OVR_ENA; //overrun error interrupt disabled
adc_cfg.seqa_ctrl = CHANNELS(0xFFF) | TRIGGER(0x0); //use all 12 channels and
     // trigger # 0
adc_cfg.channel_num = 0xC; // highest channel number is 12
```

8. Initialize ADC0 conversion:

```
pAdcApi->adc_init(adc_handle_0, &adc_set);
```

9. Enable the ADC_SEQA interrupt in the NVIC.

10. Set up the ADC parameter structure ADC_PARAM_T:

```
param.driver_mode = 0x01; //interrupt mode.
param.seqa_hwtrig = 0x1;
param.adc_cfg = (ADC_CONFIG_T *)&adc_cfg;
param.buffer = (uint32_t *)adc_buffer;
param.comp_flags = 0;
param.seqa_callback_pt = adc_seqa_callback;
```

11. Define the callback function invoked when the sequence A interrupt is raised. The ADC0 handle is passed as an argument to the callback function.

```
void  adc_seqa_callback( ADC_HANDLE_T handle ) {
     ADC_DRIVER_TypeDef *driver = (ADC_DRIVER_TypeDef *)handle;
     ADC_REGS_T *adcr = ((ADC_REGS_T *)((ADC_DRIVER_TypeDef
          *)handle)->base_addr);

     if ((ErrorCode_t)driver->error_code != LPC_OK) {
          while(1);
     }
     adcr->SEQA_CTRL &= ~( ADC_SEQ_ENA );           // stop ADC SEQA now.
     adcr->INTEN &= ~SEQA_ENA;
     completiona_tag = 1;

}
```

12. Perform conversion. The result is returned in the callback function.

```
pAdcApi->adc_seqa_read(adc_handle, &param);
while (!completiona_tag); // wait until ADC conversion is complete
```

31.1 How to read this chapter

The debug functionality is identical for all LPC82x parts.

31.2 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.
- Micro Trace Buffer (MTB) supported.

31.3 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 and Micro Trace Buffer is available. In order to use the micro-trace buffer for debugging, enable the MTB clock in the SYSAHBCLKCTRL register (Table 35).

Only RAM0 can be used as trace buffer by MTB, that means the maximum trace buffer size is 4 KB.

31.4 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 7.3.2 to enable the analog comparator inputs and the reference voltage input.

Table 395. 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 79
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 LPC800. This pin is pulled up internally.	PINENABLE0	Table 79

The boundary scan mode and the pins needed are selected by hardware (see Section 31.5.3). There is no access to the boundary scan pins through the switch matrix.

Table 396. 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.

31.5 Functional description

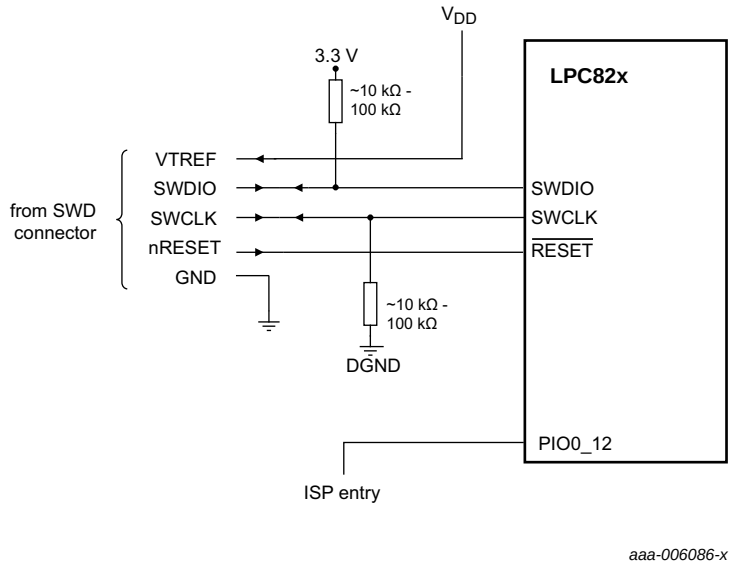
31.5.1 Debug limitations

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.

31.5.2 Debug connections for SWD

For debugging purposes, it is useful to provide access to the ISP entry pin PIO0_1. This pin can be used to recover the part from configurations which would disable the SWD port such as improper PLL configuration, 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 65. Connecting the SWD pins to a standard SWD connector

31.5.3 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 250 μs.
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.

31.5.4 Micro Trace Buffer (MTB)

The MTB registers are located at memory address 0x1400 0000 and are described in Ref. 4. The EXTTRACE register in the SYSCON block (see Section 5.6.21) starts and stops tracing in conjunction with the TSTARTEN and TSTOPEN bits in the MTB MASTER register. The trace is stored in the local SRAM starting at address 0x1000 0000. The trace memory location is configured in the MTB POSITION register.

Remark: The MTB BASE register is not implemented. Reading the BASE register returns 0x0 independently of the SRAM memory area configured for trace.

32.1 How to read this chapter

The ROM-based 32-bit integer division routines are available on all parts.

32.2 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.

32.3 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 66 shows the pointer structure used to call the Integer divider API.

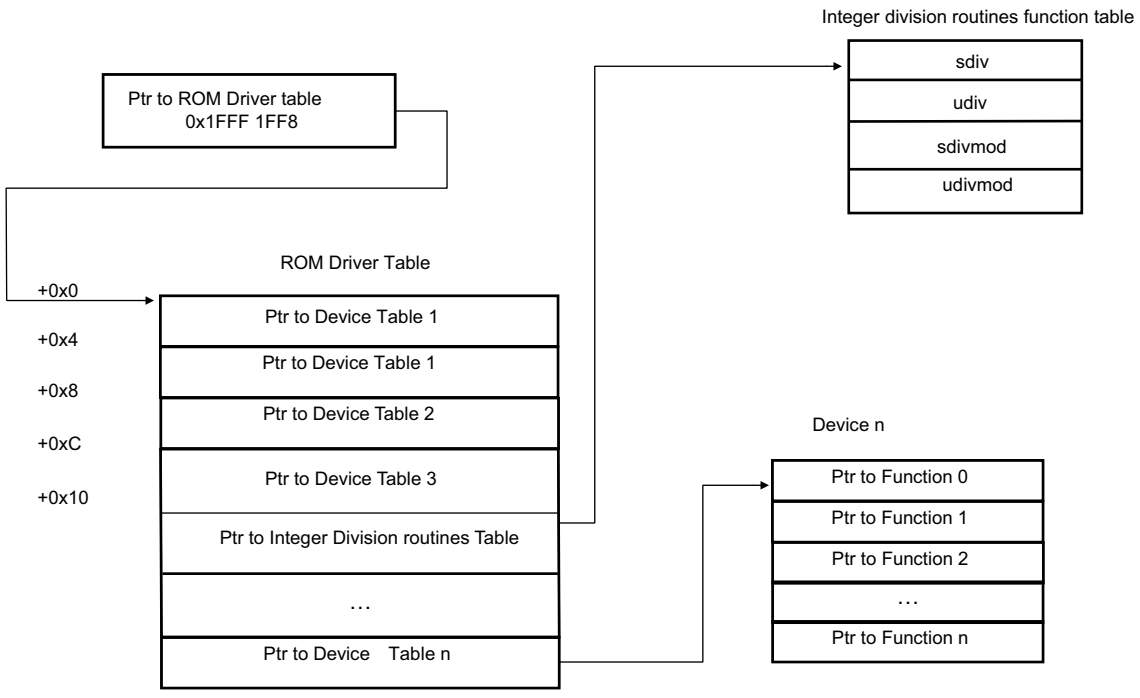


Fig 66. ROM pointer structure

32.4 API description

The integer division routines perform arithmetic integer division operations and can be called in the application code through simple API calls.

Table 397. Divide API calls

API call	Description	Reference
`int(*sdiv)(int numerator, int denominator);`	Signed integer division	Table 398
`unsigned(*udiv) (int numerator, int denominator);`	Unsigned integer division	Table 399
`sdiv_t (*sdivmod)(int numerator, int denominator);`	Signed integer division with remainder	Table 400
`udiv_t (*udivmod)(unsigned numerator, unsigned denominator);`	Unsigned integer division with remainder	Table 401

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 3.5.2 “ROM-based APIs” must be included in the code.

32.4.1 DIV signed integer division

Table 398. sidiv

Routine	sidiv
Prototype	`int(*sidiv)(int32_t numerator, int32_t denominator);`

Table 398. sidiv

Routine	sidiv
Input parameter	numerator: Numerator signed integer. denominator: Denominator signed integer.
Return	Signed division result without remainder.
Description	Signed integer division

32.4.2 DIV unsigned integer division

Table 399. 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

32.4.3 DIV signed integer division with remainder

Table 400. 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

32.4.4 DIV unsigned integer division with remainder

Table 401. 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

32.5 Functional description

32.5.1 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) */
```

32.5.2 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) */
```

33.1 Pin description

The pin description table Table 402 shows the pin functions that are fixed to specific pins on each package. These fixed-pin functions are selectable between GPIO and the comparator, ADC, SWD, RESET, and the XTAL pins. By default, the GPIO function is selected except on pins PIO0_2, PIO0_3, and PIO0_5. JTAG functions are available in boundary scan mode only.

Movable function for the I2C, USART, SPI, and SCT pin functions can be assigned through the switch matrix to any pin that is not power or ground in place of the pin's fixed functions.

The following exceptions apply:

For full I2C-bus compatibility, assign the I2C functions to the open-drain pins PIO0_11 and PIO0_10.

Do not assign more than one output to any pin. However, more than one input can be assigned to a pin. Once any function is assigned to a pin, the pin's GPIO functionality is disabled.

Pin PIO0_4 triggers a wake-up from Deep power-down mode. If you need to wake up from Deep power-down mode via an external pin, do not assign any movable function to this pin.

The JTAG functions TDO, TDI, TCK, TMS, and TRST are selected on pins PIO0_0 to PIO0_4 by hardware when the part is in boundary scan mode.

Table 402. Pin description

Symbol	TSSOP20	HVQFN33		Reset state[1]	Type	Description
PIO0_0/ACMP_I1/ TDO	19	24	[2]	I; PU	IO	**PIO0_0 —** General purpose port 0 input/output 0. In ISP mode, this the U0_RXD pin. In boundary scan mode: TDO (Test Data Out).
					A	**ACMP_I1 —** Analog comparator input 1.
PIO0_1/ACMP_I2/ CLKIN/TDI	12	16	[2]	I; PU	IO	**PIO0_1 —** General purpose port 0 input/output 1. In boundary scan mode: TDI (Test Data In).
					A	**ACMP_I2 —** Analog comparator input 2.
					I	**CLKIN —** External clock input.
SWDIO/PIO0_2/ TMS	8	7	[4]	I; PU	IO	**SWDIO —** Serial Wire Debug I/O. SWDIO is enabled by default on this pin. In boundary scan mode: TMS (Test Mode Select).
					I/O	**PIO0_2 —** General purpose port 0 input/output 2.

Table 402. Pin description

Symbol	TSSOP20	HVQFN33		Reset state[1]	Type	Description
SWCLK/PIO0_3/ TCK	7	6	[4]	I; PU	I	**SWCLK —** Serial Wire Clock. SWCLK is enabled by default on this pin. In boundary scan mode: TCK (Test Clock).
					IO	**PIO0_3 —** General purpose port 0 input/output 3.
PIO0_4/ADC_11/ TRSTN/WAKEUP	6	4	[3]	I; PU	IO	**PIO0_4 —** General purpose port 0 input/output 4. In boundary scan mode: $\overline{\text{TRST}}$ (Test Reset). In ISP mode, this pin is the U0_TXD pin. This pin triggers a wake-up from Deep power-down mode. If you need to wake up from Deep power-down mode via an external pin, do not assign any movable function to this pin. This pin should be pulled HIGH externally before entering Deep power-down mode. A LOW-going pulse as short as 50 ns causes the chip to exit Deep power-down mode and wakes up the part.
					A	**ADC_11 —** ADC input 11.
$\overline{\text{RESET}}$/PIO0_5	5	3	[7]	I; PU	IO	**$\overline{\text{RESET}}$ —** External reset input: A LOW-going pulse as short as 50 ns on this pin resets the device, causing I/O ports and peripherals to take on their default states, and processor execution to begin at address 0. In deep power-down mode, this pin must be pulled HIGH externally. The $\overline{\text{RESET}}$ pin can be left unconnected or be used as a GPIO or for any movable function if an external $\overline{\text{RESET}}$ function is not needed and the Deep power-down mode is not used.
					I	**PIO0_5 —** General purpose port 0 input/output 5.
PIO0_6/ADC_1/ VDDCMP	-	23	[10]	I; PU	IO	**PIO0_6 —** General purpose port 0 input/output 6.
					A	**ADC_1 —** ADC input 1.
					A	**VDDCMP —** Alternate reference voltage for the analog comparator.
PIO0_7/ADC_0	-	22	[2]	I; PU	IO	**PIO0_7 —** General purpose port 0 input/output 7.
					A	**ADC_0 —** ADC input 0.
PIO0_8/XTALIN	14	18	[8]	I; PU	IO	**PIO0_8 —** General purpose port 0 input/output 8.
					A	**XTALIN —** Input to the oscillator circuit and internal clock generator circuits. Input voltage must not exceed 1.95 V.
PIO0_9/XTALOUT	13	17	[8]	I; PU	IO	**PIO0_9 —** General purpose port 0 input/output 9.
					A	**XTALOUT —** Output from the oscillator circuit.
PIO0_10/I2C0_SCL	10	9	[6]	Inactive	I; F	**PIO0_10 —** General purpose port 0 input/output 10 (open-drain).
						I2C0_SCL — Open-drain I^2C-bus clock input/output. High-current sink if I^2C Fast-mode Plus is selected in the I/O configuration register.
PIO0_11/I2C0_SDA	9	8	[6]	Inactive	I; F	**PIO0_11 —** General purpose port 0 input/output 11 (open-drain).
						I2C0_SDA — Open-drain I^2C-bus data input/output. High-current sink if I^2C Fast-mode Plus is selected in the I/O configuration register.

Table 402. Pin description

Symbol	TSSOP20	HVQFN33		Reset state[1]	Type	Description
PIO0_12	4	2	[4]	I; PU	IO	**PIO0_12 —** General purpose port 0 input/output 12. ISP entry pin. A LOW level on this pin during reset starts the ISP command handler.
PIO0_13/ADC_10	3	1	[2]	I; PU	IO	**PIO0_13 —** General purpose port 0 input/output 13.
					A	**ADC_10 —** ADC input 10.
PIO0_14/ ACMP_I3/ADC_2	20	25	[2]	I; PU	IO	**PIO0_14 —** General purpose port 0 input/output 14.
					A	**ACMP_I3 —** Analog comparator common input 3.
					A	**ADC_2 —** ADC input 2.
PIO0_15	11	15	[5]	I; PU	IO	**PIO0_15 —** General purpose port 0 input/output 15.
PIO0_16	-	10	[4]	I; PU	IO	**PIO0_16 —** General purpose port 0 input/output 16.
PIO0_17/ADC_9	2	32	[2]	I; PU	IO	**PIO0_17 —** General purpose port 0 input/output 17.
					A	**ADC_9 —** ADC input 9.
PIO0_18/ADC_8	-	31	[2]	I; PU	IO	**PIO0_18 —** General purpose port 0 input/output 18.
					A	**ADC_8 —** ADC input 8.
PIO0_19/ADC_7	-	30	[2]	I; PU	IO	**PIO0_19 —** General purpose port 0 input/output 19.
					A	**ADC_7 —** ADC input 7.
PIO0_20/ADC_6	-	29	[2]	I; PU	IO	**PIO0_20 —** General purpose port 0 input/output 20.
					A	**ADC_6 —** ADC input 6.
PIO0_21/ADC_5	-	28	[2]	I; PU	IO	**PIO0_21 —** General purpose port 0 input/output 21.
					A	**ADC_5 —** ADC input 5.
PIO0_22/ADC_4	-	27	[2]	I; PU	IO	**PIO0_22 —** General purpose port 0 input/output 22.
					A	**ADC_4 —** ADC input 4.
PIO0_23/ADC_3/ ACMP_I4	1	26	[2]	I; PU	IO	**PIO0_23 —** General purpose port 0 input/output 23.
					A	**ADC_3 —** ADC input 3.
					A	**ACMP_I4 —** Analog comparator common input 4.
PIO0_24	-	14	[5]	I; PU	IO	**PIO0_24 —** General purpose port 0 input/output 24.
PIO0_25	-	13	[5]	I; PU	IO	**PIO0_25 —** General purpose port 0 input/output 25.
PIO0_26	-	12	[5]	I; PU	IO	**PIO0_26 —** General purpose port 0 input/output 26.
PIO0_27	-	11	[5]	I; PU	IO	**PIO0_27 —** General purpose port 0 input/output 27.
PIO0_28/ WKTCLKIN	-	5	[3]	I; PU	IO	**PIO0_28 —** General purpose port 0 input/output 28. This pin can host an external clock for the self-wake-up timer. To use the pin as a self-wake-up timer clock input, select the external clock in the wake-up timer CTRL register. The external clock input is active in all power modes, including deep power-down.
V_{DD}	15	19		-	-	Supply voltage for the I/O pad ring, the core voltage regulator, and the analog peripherals.
VSS	16	33		-	-	Ground.
VREFN	17	20		-	-	ADC negative reference voltage.
VREFP	18	21		-	-	ADC positive reference voltage. Must be equal or lower than V_{DD}.

[1] Pin state at reset for default function: I = Input; AI = Analog Input; O = Output; PU = internal pull-up enabled (pins pulled up to full V_{DD} level); IA = inactive, no pull-up/down enabled; F = floating.

[2] 5 V tolerant pin providing standard digital I/O functions with configurable modes, configurable hysteresis, and analog input. When configured as an analog input, the digital section of the pin is disabled, and the pin is not 5 V tolerant.

[3] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors and configurable hysteresis. This pin is active in Deep power-down mode and includes a 20 ns glitch filter (active in all power modes). In Deep power-down mode, pulling the WAKEUP pin LOW wakes up the chip. The wake-up pin function can be disabled and the pin can be used for other purposes, if the WKT low power oscillator is enabled for waking up the part from Deep power-down mode.

[4] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors and configurable hysteresis; includes high-current output driver.

[5] 5 V tolerant pad providing digital I/O functions with configurable pull-up/pull-down resistors and configurable hysteresis.

[6] True open-drain pin. I^2C-bus pins compliant with the I^2C-bus specification for I^2C standard mode, I^2C Fast-mode, and I^2C Fast-mode Plus. Do not use this pad for high-speed applications such as SPI or USART. The pin requires an external pull-up to provide output functionality. When power is switched off, this pin is floating and does not disturb the I2C lines. Open-drain configuration applies to all functions on this pin.

[7] This pin includes a 20 ns glitch filter (active in all power modes). $\overline{\text{RESET}}$ functionality is not available in Deep power-down mode. Use the WAKEUP pin to reset the chip and wake up from Deep power-down mode. An external pull-up resistor is required on this pin for the Deep power-down mode.

[8] 5 V tolerant pin providing standard digital I/O functions with configurable modes, configurable hysteresis, and analog I/O for the system oscillator. When configured for XTALIN and XTALOUT, the digital section of the pin is disabled, and the pin is not 5 V tolerant.

[9] The WKTCLKIN function is enabled in the DPDCTRL register in the PMU. See the LPC82x user manual.

[10] The digital part of this pin is 3 V tolerant pin due to special analog functionality. Pin provides standard digital I/O functions with configurable modes, configurable hysteresis, and an analog input. When configured as an analog input, the digital section of the pin is disabled.

34.1 How to read this chapter

This chapter contains code examples to help understand how to use the registers of various peripheral blocks when writing software drivers. For a comprehensive description of each peripheral and register interface, see the respective chapter.

Remark: The code listings are for illustrative purposes only and are not intended to be application-ready functions.

34.2 Code examples I2C

34.2.1 Definitions

Table 403. I2C Code example

I2C defines

```
#define I2C_CFG_MSTEN (0x1)
#define I2C_CFG_SLVEN (0x2)
#define I2C_STAT_MSTPENDING (0x1)
#define I2C_STAT_MSTSTATE (0xe)
#define I2C_STAT_MSTST_IDLE (0x0)
#define I2C_STAT_MSTST_RX (0x2)
#define I2C_STAT_MSTST_TX (0x4)
#define I2C_STAT_MSTST_NACK_ADDR (0x6)
#define I2C_STAT_MSTST_NACK_TX (0x8)
#define I2C_STAT_SLVPENDING (0x100)
#define I2C_STAT_SLVSTATE (0x600)
#define I2C_STAT_SLVST_ADDR (0x000)
#define I2C_STAT_SLVST_RX (0x200)
#define I2C_STAT_SLVST_TX (0x400)
#define I2C_MSTCTL_MSTCONTINUE (0x1)
#define I2C_MSTCTL_MSTSTART (0x2)
#define I2C_MSTCTL_MSTSTOP (0x4)
#define I2C_SLVCTL_SLVCONTINUE (0x1)
#define I2C_SLVCTL_SLVNACK (0x2)
```

34.2.2 Interrupt handler

Table 404. I2C Code example

Interrupt handler

```
void I2c_IRQHandler() {
uint32_t intstat = LPC_I2C->INTSTAT;
     uint32_t stat = LPC_I2C->STAT;
     if(intstat & I2C_STAT_MSTPENDING) {
          uint32_t mst_state = stat & I2C_STAT_MSTSTATE;
          if(mst_state == I2C_STAT_MSTST_IDLE) {
               LPC_I2C->MSTDAT = (0x23 << 1) | 1; // address and 1 for RWn bit in
     order
// to read data
               LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
          }
          if(mst_state == I2C_STAT_MSTST_RX) {
               uint8_t data;
               data = LPC_I2C->MSTDAT; // receive data
               if(data != 0xdd) abort();
               LPC_I2C->MSTDAT = (0x23 << 1) | 0; // address and 1 for RWn bit in
     order
// to read data
               LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // repeated start (nack
     implied)
          }
          if(mst_state == I2C_STAT_MSTSTX) {
               LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // stop transaction
               LPC_I2C->INTENCLR = I2C_STAT_MSTPENDING;
          }
     }
}
```

34.2.3 Master write one byte to slave

Table 405. I2C Code example

Master write one byte to slave. Address 0x23, Data 0xdd. Polling mode.

```
LPC_I2C->CFG = I2C_CFG_MSTEN;
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
LPC_I2C->MSTDAT = (0x23 << 1) | 0; // address and 0 for RWn bit
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTSTX) abort();
LPC_I2C->MSTDAT = 0xdd; // send data
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTCONTINUE; // continue transaction
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTSTX) abort();
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
```

34.2.4 Master read one byte from slave

Table 406. I2C Code example

Master read one byte from slave. Address 0x23. Polling mode. No error checking.

LPC_I2C->CFG = I2C_CFG_MSTEN;
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
LPC_I2C->MSTDAT = (0x23 << 1) | 1; // address and 1 for RWn bit
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_RX) abort();
data = LPC_I2C->MSTDAT; // read data
if(data != 0xdd) abort();
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();

34.2.5 Master write one byte to subaddress on slave

Table 407. I2C Code example

Master write one byte to subaddress on slave. Address 0x23, subaddress 0xaa, Data 0xdd. Polling mode. No error checking.

LPC_I2C->CFG = I2C_CFG_MSTEN;
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
LPC_I2C->MSTDAT = (0x23 << 1) | 0; // address and 0 for RWn bit in order to write
// subaddress
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTSTX) abort();
LPC_I2C->MSTDAT = 0xaa; // send subaddress
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTCONTINUE; // continue transaction
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTSTX) abort();
LPC_I2C->MSTDAT = 0xdd; // send data
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTCONTINUE; // continue transaction
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTSTX) abort();
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();

34.2.6 Master read one byte from subaddress on slave

Table 408. I2C Code example

Master read one byte from subaddress on slave. Address 0x23, subaddress 0xaa. Polling mode. No error checking.

LPC_I2C->CFG = I2C_CFG_MSTEN;
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
LPC_I2C->MSTDAT = (0x23 << 1) | 0; // address and 0 for RWn bit in order to write
// subaddress
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTSTX) abort();
LPC_I2C->MSTDAT = 0xaa; // send subaddress
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTCONTINUE; // continue transaction
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTSTX) abort();
LPC_I2C->MSTDAT = (0x23 << 1) | 1; // address and 1 for RWn bit in order to write
// subaddress
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send repeated start
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_RX) abort();
data = LPC_I2C->MSTDAT; // read data
if(data != 0xdd) abort();
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // send stop
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();

34.2.7 Master receiving nack on address

Table 409. I2C Code example

Master receive nack on address. Address 0x23. Polling mode. No error checking.

LPC_I2C->CFG = I2C_CFG_MSTEN;
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
LPC_I2C->MSTDAT = (0x23 << 1) | 0; // address and 0 for RWn bit
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_NACK_ADDR) abort();
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // stop transaction
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();

34.2.8 Master receiving nack on data

Table 410. I2C Code example

Master receive nack on data. Address 0x23, data 0xdd. Polling mode. No error checking.

LPC_I2C->CFG = I2C_CFG_MSTEN;
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
LPC_I2C->MSTDAT = (0x23 << 1) | 0; // address and 0 for RWn bit in order to write data
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTSTX) abort();
LPC_I2C->MSTDAT = 0xdd; // send data
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTCONTINUE; // continue transaction
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_NACKX) abort();
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // stop transaction
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();

34.2.9 Master sending nack and stop on data

Table 411. I2C Code example

Master sending nack and stop on data. Address 0x23. Polling mode. No error checking.

LPC_I2C->CFG = I2C_CFG_MSTEN;
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
LPC_I2C->MSTDAT = (0x23 << 1) | 1; // address and 1 for RWn bit in order to read data
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_RX) abort();
data = LPC_I2C->MSTDAT; // receive data
if(data != 0xdd) abort();
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // stop transaction (nack implied)
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();

34.2.10 Master sending nack and repeated start on data

Table 412. I2C Code example

Master sending nack and repeated start on data. Address 0x23. Polling mode. No error checking.

LPC_I2C->CFG = I2C_CFG_MSTEN;
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
LPC_I2C->MSTDAT = (0x23 << 1) | 1; // address and 1 for RWn bit in order to read data
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // send start
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_RX) abort();
data = LPC_I2C->MSTDAT; // receive data
if(data != 0xdd) abort();
LPC_I2C->MSTDAT = (0x23 << 1) | 0; // address and 1 for RWn bit in order to read data
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTART; // repeated start (nack implied)
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
LPC_I2C->MSTCTL = I2C_MSTCTL_MSTSTOP; // stop transaction
while(!(LPC_I2C->STAT & I2C_STAT_MSTPENDING));
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();

34.2.11 Master sending nack and repeated start on data. Interrupt mode

Table 413. I2C Code example

Master sending nack and repeated start on data. Address 0x23. No error checking. Interrupt mode

LPC_I2C->CFG = I2C_CFG_MSTEN;
LPC_I2C->INTENSET = I2C_STAT_MSTPENDING;
NVIC_EnableIRQ(I2c_IRQn);
while(LPC_I2C->INTENSET & I2C_STAT_MSTPENDING);
if((LPC_I2C->STAT & I2C_STAT_MSTSTATE) != I2C_STAT_MSTST_IDLE) abort();
NVIC_DisableIRQ(I2c_IRQn);

34.2.12 Slave read one byte from master

Table 414. I2C Code example

Slave read one byte from master. Address 0x23. Polling mode.

LPC_I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register
LPC_I2C->CFG = I2C_CFG_SLVEN;
while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING));
if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address
while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING));
if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_RX) abort();
data = LPC_I2C->SLVDAT; // read data
if(data != 0xdd) abort();
LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack data

34.2.13 Slave write one byte to master

Table 415. I2C Code example

Slave write one byte to master. Address 0x23, Data 0xdd. Polling mode.

LPC_I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register
LPC_I2C->CFG = I2C_CFG_SLVEN;
while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING));
if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address
while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING));
if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVSTX) abort();
LPC_I2C->SLVDAT = 0xdd; // write data
LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // continue transaction

34.2.14 Slave read one byte from master into subaddress

Table 416. I2C Code example

Slave read one byte from master into subaddress. Address 0x23. Polling mode.

LPC_I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register
LPC_I2C->CFG = I2C_CFG_SLVEN;
while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING));
if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address
while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING));
if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_RX) abort();
subaddress = LPC_I2C->SLVDAT; // read subaddress
if(subaddress != 0xaa) abort();
LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack data
while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING));
if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_RX) abort();
data[subaddress] = LPC_I2C->SLVDAT; // read data into subaddress
if(data[subaddress] != 0xdd) abort();
LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack data

34.2.15 Slave write one byte to master from subaddress

Table 417. I2C Code example

Slave write one byte to master from subaddress. Address 0x23. Polling mode.

LPC_I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register
LPC_I2C->CFG = I2C_CFG_SLVEN;
while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING));
if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address
while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING));
if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_RX) abort();
subaddress = LPC_I2C->SLVDAT; // read subaddress
if(subaddress != 0xaa) abort();
LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // continue transaction
while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING));
if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address
while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING));
if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVSTX) abort();
LPC_I2C->SLVDAT = data[subaddress]; // write data from subaddress
LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // continue transaction

34.2.16 Slave nack matched address from master

Table 418. I2C Code example

Slave nack matched address from master. Address 0x23. Polling mode.

LPC_I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register
LPC_I2C->CFG = I2C_CFG_SLVEN;
while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING));
if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
LPC_I2C->SLVCTL = I2C_SLVCTL_SLVNACK; // nack address

34.2.17 Slave nack data from master

Table 419. I2C Code example

Slave nack data from master. Address 0x23. Polling mode.

LPC_I2C->SLVADR0 = 0x23 << 1; // put address in address 0 register
LPC_I2C->CFG = I2C_CFG_SLVEN;
while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING));
if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_ADDR) abort();
LPC_I2C->SLVCTL = I2C_SLVCTL_SLVCONTINUE; // ack address
while(!(LPC_I2C->STAT & I2C_STAT_SLVPENDING));
if((LPC_I2C->STAT & I2C_STAT_SLVSTATE) != I2C_STAT_SLVST_RX) abort();
data = LPC_I2C->SLVDAT; // read data
if(data != 0xdd) abort();
LPC_I2C->SLVCTL = I2C_SLVCTL_SLVNACK; // nack data

34.3 Code examples SPI

34.3.1 Definitions

Table 420. SPI Code example

SPI defines

```
#define SPI_CFG_ENABLE (0x1)
#define SPI_CFG_MASTER (0x4)
#define SPI_STAT_RXRDY (0x1)
#define SPI_STAT_TXRDY (0x2)
#define SPI_STAT_SSD (0x20)
#define SPI_STAT_MSTIDLE (0x100)
#define SPI_TXDATCTL_SSEL_N(s) ((s) << 16)
#define SPI_TXDATCTL_EOT (1 << 20)
#define SPI_TXDATCTL_EOF (1 << 21)
#define SPI_TXDATCTL_RXIGNORE (1 << 22)
#define SPI_TXDATCTL_FLEN(l) ((l) << 24)
```

34.3.2 Interrupt handler

Table 421. SPI Code example

Interrupt handler

```
void Spi_IRQHandler() {
     uint16_t data;
     uint32_t intstat = LPC_SPI->INTSTAT;
     if(intstat & SPI_STAT_TXRDY) {
          if(tx_state == 0) {
               LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(15) |
     SPI_TXDATCTL_SSEL_N(0xe) | 0xdddd;
               tx_state++;
          }
          if(tx_state == 1) {
               LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(7) | SPI_TXDATCTL_SSEL_N(0xe)
     | 0xdd;
               LPC_SPI->INTENCLR = SPI_STAT_TXRDY;
          }
     }
     if(intstat & SPI_STAT_RXRDY) {
          if(rx_state == 0) {
               data = LPC_SPI->RXDAT;
               if(data != 0xdddd) abort();
               rx_state++;
          }
          if(rx_state == 1) {
               data = LPC_SPI->RXDAT;
               if(data != 0xdd) abort();
               LPC_SPI->INTENCLR = SPI_STAT_RXRDY;
          }
     }
}
```

34.3.3 Transmit one byte to slave 0

Table 422. SPI Code example

Transmit one byte to slave 0.

LPC_SPI->CFG = SPI_CFG_MASTER | SPI_CFG_ENABLE;
while(~LPC_SPI->STAT & SPI_STAT_TXRDY);
LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(7) | SPI_TXDATCTL_RXIGNORE | SPI_TXDATCTL_EOT
| SPI_TXDATCTL_SSEL_N(0xe) | 0xdd;
while(~LPC_SPI->STAT & SPI_STAT_MSTIDLE);

34.3.4 Receive one byte from slave 0

Table 423. SPI Code example

Receive one byte from slave 0.

LPC_SPI->CFG = SPI_CFG_MASTER | SPI_CFG_ENABLE;
while(~LPC_SPI->STAT & SPI_STAT_TXRDY);
LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(7) | SPI_TXDATCTL_EOT |
SPI_TXDATCTL_SSEL_N(0xe);
while(~LPC_SPI->STAT & SPI_STAT_RXRDY);
data = LPC_SPI->RXDAT;
if(data != 0xdd) abort();
while(~LPC_SPI->STAT & SPI_STAT_MSTIDLE);

34.3.5 Transmit and receive a byte to/from slave 0

Table 424. SPI Code example

Transmit and receive a byte to/from slave 0.

LPC_SPI->CFG = SPI_CFG_MASTER | SPI_CFG_ENABLE;
while(~LPC_SPI->STAT & SPI_STAT_TXRDY);
LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(7) | SPI_TXDATCTL_EOT |
SPI_TXDATCTL_SSEL_N(0xe) | 0xdd;
while(~LPC_SPI->STAT & SPI_STAT_RXRDY);
data = LPC_SPI->RXDAT;
if(data != 0xdd) abort();
while(~LPC_SPI->STAT & SPI_STAT_MSTIDLE);

34.3.6 Transmit and receive 24 bits to/from slave 0

Table 425. SPI Code example

Transmit and receive 24 bits to/from slave 0.

LPC_SPI->CFG = SPI_CFG_MASTER | SPI_CFG_ENABLE;
while(~LPC_SPI->STAT & SPI_STAT_TXRDY);
LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(15) | SPI_TXDATCTL_SSEL_N(0xe) | 0xdddd;
while(~LPC_SPI->STAT & SPI_STAT_RXRDY);
data = LPC_SPI->RXDAT;
if(data != 0xdddd) abort();
while(~LPC_SPI->STAT & SPI_STAT_TXRDY);
LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(7) | SPI_TXDATCTL_EOT |
SPI_TXDATCTL_SSEL_N(0xe) | 0xdd;
while(~LPC_SPI->STAT & SPI_STAT_RXRDY);
data = LPC_SPI->RXDAT;
if(data != 0xdd) abort();
while(~LPC_SPI->STAT & SPI_STAT_MSTIDLE);

34.3.7 Transmit and receive 24 bits to/from slave 0, interrupt mode

Table 426. SPI Code example

Transmit and receive 24 bits to/from slave 0, interrupt mode.
LPC_SPI->CFG = SPI_CFG_MASTER | SPI_CFG_ENABLE; LPC_SPI->INTENSET = SPI_STAT_TXRDY | SPI_STAT_RXRDY; while(LPC_SPI->INTENSET & (SPI_STAT_TXRDY | SPI_STAT_RXRDY)); NVIC_DisableIRQ(Spi_IRQn);

34.3.8 Transmit 8 bits to master

Table 427. SPI Code example

Transmit 8 bits to master.

LPC_SPI->CFG = SPI_CFG_ENABLE;
while(~LPC_SPI->STAT & SPI_STAT_TXRDY);
LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(7) | SPI_TXDATCTL_RXIGNORE | 0xdd;
while(~LPC_SPI->STAT & SPI_STAT_TXRDY);
LPC_SPI->STAT = SPI_STAT_SSD;
while(~LPC_SPI->STAT & SPI_STAT_SSD);
LPC_SPI->STAT = SPI_STAT_SSD;

34.3.9 Receive 8 bits to master

Table 428. SPI Code example

Receive 8 bits to master.

LPC_SPI->CFG = SPI_CFG_ENABLE;
while(~LPC_SPI->STAT & SPI_STAT_TXRDY);
LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(7);
while(~LPC_SPI->STAT & SPI_STAT_RXRDY);
data = LPC_SPI->RXDAT;
if(data != 0xdd) abort();

34.3.10 Transmit and receive 24 bits to master

Table 429. SPI Code example

Transmit and receive 24 bits to master.

```
LPC_SPI->CFG = SPI_CFG_ENABLE;
while(~LPC_SPI->STAT & SPI_STAT_TXRDY);
LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(15) | 0xdddd;
while(~LPC_SPI->STAT & SPI_STAT_TXRDY);
LPC_SPI->TXDATCTL = SPI_TXDATCTL_FLEN(7) | 0xdd;
while(~LPC_SPI->STAT & SPI_STAT_RXRDY);
data = LPC_SPI->RXDAT;
if(data != 0xdddd) abort();
while(~LPC_SPI->STAT & SPI_STAT_RXRDY);
data = LPC_SPI->RXDAT;
if(data != 0xdd) abort();
```

34.4 Code examples UART

34.4.1 Definitions

Table 430. UART Code example

UART defines

```
#define UART_CFG_ENABLE (0x1 << 0)
#define UART_CFG_DATALEN(d) (((d) - 7) << 2)
#define UART_STAT_RXRDY (0x1 << 0)
#define UART_STAT_TXRDY (0x1 << 2)
#define UART_STAT_TXIDLE (0x1 << 3)
```

34.4.2 Interrupt handler

Table 431. UART Code example

Interrupt handler

```
void Usart_IRQHandler() {
     uint32_t intstat = LPC_USART->INTSTAT;
     if(intstat & UART_STAT_RXRDY) {
          if(!tx_rdy_flag) abort();
          tx_rdy_flag = 0;
          LPC_USART->TXDAT = LPC_USART->RXDAT;
          LPC_USART->INTENSET = UART_STAT_TXRDY;
          tx_counter++;
     }
     if(intstat & UART_STAT_TXRDY) {
          if(tx_rdy_flag) abort();
          tx_rdy_flag = 1;
          LPC_USART->INTENCLR = UART_STAT_TXRDY;
     }
}
```

34.4.3 Transmit one byte of data

Table 432. UART Code example

Transmit one byte of data.

`LPC_USART->CFG = UART_CFG_DATALEN(8) | UART_CFG_ENABLE;`
`while(~LPC_USART->STAT & UART_STAT_TXRDY);`
`LPC_USART->TXDAT = 0xdd;`
`while(~LPC_USART->STAT & UART_STAT_TXIDLE);`

34.4.4 Receive one byte of data

Table 433. UART Code example

Receive one byte of data.

`LPC_USART->CFG = UART_CFG_DATALEN(8) | UART_CFG_ENABLE;`
`while(~LPC_USART->STAT & UART_STAT_RXRDY);`
`data = LPC_USART->RXDAT;`
`if(data != 0xdd) abort();`

34.4.5 Transmit and receive one byte of data

Table 434. UART Code example

Transmit and receive one byte of data.

`LPC_USART->CFG = UART_CFG_DATALEN(8) | UART_CFG_ENABLE;`
`while(~LPC_USART->STAT & UART_STAT_TXRDY);`
`LPC_USART->TXDAT = 0xdd;`
`while(~LPC_USART->STAT & UART_STAT_RXRDY);`
`data = LPC_USART->RXDAT;`
`if(data != 0xdd) abort();`

34.4.6 Loop back 10 bytes of data

Table 435. UART Code example

Loop back 10 bytes of data.

`LPC_USART->CFG = UART_CFG_DATALEN(8) | UART_CFG_ENABLE;`
`for(i = 0; i < 10; i++) {`
`  while(~LPC_USART->STAT & (UART_STAT_TXRDY | UART_STAT_RXRDY));`
`  LPC_USART->TXDAT = LPC_USART->RXDAT;`
`}`
`while(~LPC_USART->STAT & UART_STAT_TXIDLE);`

34.4.7 Loop back 10 bytes of data using interrupts

Table 436. UART Code example

Loop back 10 bytes of data using interrupts.

`LPC_USART->CFG = UART_CFG_DATALEN(8) | UART_CFG_ENABLE;`
`LPC_USART->INTENSET = UART_STAT_TXRDY | UART_STAT_RXRDY;`
`while(tx_counter < 10);`
`LPC_USART->INTENCLR = UART_STAT_TXRDY | UART_STAT_RXRDY;`
`while(~LPC_USART->STAT & UART_STAT_TXIDLE);`
`NVIC_DisableIRQ(Usart_IRQn);`

35.1 Abbreviations

Table 437. 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

35.2 References

[1] **LPC82X —** LPC82X Data sheet

[2] **ES_LPC82X —** LPC82X 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

35.3 Legal information

35.3.1 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.

35.3.2 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.

35.3.3 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.

35.4 Tables

35.5 Figures

35.6 Contents

Chapter 6: LPC82x Reduced power modes and power management

Chapter 7: LPC82x Switch matrix (SWM)

Chapter 8: LPC82x I/O configuration (IOCON)

Chapter 9: LPC82x General Purpose I/O (GPIO)

Chapter 10: LPC82x Pin interrupts/pattern match engine

Chapter 11: LPC82x Input multiplexing and DMA trigger multiplexing (INPUT MUX, DMA TRIGMUX)

Chapter 12: LPC82x DMA controller

Chapter 13: LPC82x USART0/1/2

Chapter 14: LPC82x SPI0/1

Chapter 15: LPC82x I2C0/1/2/3

Chapter 16: LPC82x SCTimer/PWM

Chapter 17: LPC82x Windowed Watchdog Timer (WWDT)

Chapter 18: LPC82x Self-wake-up timer (WKT)

Chapter 19: LPC82x Multi-Rate Timer (MRT)

Chapter 20: LPC82x System tick timer (SysTick)

Chapter 21: 12-bit Analog-to-Digital Converter (ADC)

Chapter 22: LPC82x Analog comparator

Chapter 23: LPC82x CRC engine

Chapter 24: LPC82x Flash controller

Chapter 25: LPC82x Flash In-System and In-Application Programming (ISP and IAP)

Chapter 26: LPC82x ROM API Power profiles

Chapter 27: LPC82x ROM API USART driver routines

Chapter 28: LPC82x SPI API ROM driver routines

Chapter 29: LPC82x ROM API I2C driver routines

Chapter 30: LPC82x ROM API ADC drivers

Chapter 31: LPC82x Serial Wire Debug (SWD)

Chapter 32: LPC82x ROM API integer divide routines

Chapter 33: LPC82x Pin description

Chapter 34: LPC82x Code examples

Chapter 35: Supplementary information

continued >>

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 18 September 2014
Document identifier: UM10800