UM10441

LPC1224/25/26/27 User manual

Rev. 2.2 — 16 May 2017 **User manual**

Document information

Info	Content
Keywords	LPC122x, LPC1227, LPC1226, LPC1225, LPC1224, LPC12D27, ARM Cortex-M0, microcontroller
Abstract	LPC122x User manual

Revision history

Rev	Date	Description
2.2	20170516	LPC122x User manual
Modifications:		• Fixed typos in the "Value" column for the FUNC field of the PIO2_0 register. The correct functions are: 0x0: PIO2_0, 0x1: Reserved, 0x2: CT16B0_CAP0, 0x3: CT16B0_MAT0, 0x4: RTS0. See Table 88 “PIO2_0 register (PIO2_0, address 0x4004 4070) bit description”. • Updated the FLASH_OVERRIDE description to add flash wait states for frequencies above 30 MHz and below 30 MHZ. See Table 9 “Peripheral reset control register (PRESETCTRL, address 0x4004 8004) bit description” and Table 52 “Flash configuration register (FLASHCFG, address 0x5006 0028) bit description” • Updated Table 71 “PIO0_27 register (PIO0_27, address 0x4004 4028) bit description”, PIO0_27 register. Bit 2:0, FUNC, value 0x0 selects function PIO0_27; was PIO0_26.
2.1	20140527	LPC122x User manual
Modifications:		• Remove instruction breakpoints from feature list for SWD. See Section 24.2. • The BOD must be powered to operate the ADC. See Section 19.2. • Use of UART0 for ISP mode clarified. See Table 118 “LPC122x pin description” and Section 20.2. • Remark added: Do not write 1s to reserved bits in GPIO ports GPIO1 and GPIO2. See Section 8.3.5 and Section 8.3.7. • Added this step to deep-sleep mode configuration instructions: Configure all pins hosting comparator inputs as GPIO outputs and drive the outputs to LOW. Disable the pin’s internal pull-up/pull-down resistors. Affects pins PIO0_19, PIO0_20, PIO0_21, PIO0_22, PIO0_23, PIO0_24, PIO0_25, and PIO0_26.See Section 4.7.3.2 “Programming Deep-sleep mode”.
2	20110919	LPC122x User manual
Modifications:		• Pinout updated for pins RTCXOUT and RTCXIN (LQFP64 package) in Table 118. • DRV bit updated for IOCON registers Table 63 to Table 70. 0 = High mode current selected, 1 = Low mode current selected. • Corrected alignment of the DMA control structure (Section 21.7.5 “DMA control”). • RTC functional description expanded (Section 16.6). • Part LPC12D27 added. • Start logic 1 registers updated (Table 42 to Table 45). • LPC12D27 pin configuration added (Table 119).
1	20110215	LPC122x 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 LPC122x extend NXP's 32-bit ARM microcontroller continuum and target a wide range of industrial applications in the areas of factory and home automation. Benefitting from the ARM Cortex-M0 Thumb instruction set, the LPC122x have up to 50 % higher code density compared to common 8/16-bit microcontroller performing typical tasks. The LPC122x also feature an optimized ROM-based divide library for Cortex-M0, which offers several times the arithmetic performance of software based libraries, as well as highly deterministic cycle time combined with reduced flash code size. The ARM Cortex-M0 efficiency also helps the LPC122x achieve lower average power for similar applications.

The LPC122x operate at CPU frequencies of up to 45 MHz.They offer a wide range of flash memory options, from 32 kB to 128 kB. The small 512-byte page erase of the flash memory brings multiple design benefits, such as finer EEPROM emulation, boot-load support from any serial interface and ease of in-field programming with reduced on-chip RAM buffer requirements.

The peripheral complement of the LPC122x includes a 10-bit ADC, two comparators with output feedback loop, two UARTs, one SSP/SPI interface, one I^2C-bus interface with Fast-mode Plus features, a Windowed Watchdog Timer, a DMA controller, a CRC engine, four general purpose timers, a 32-bit RTC, a 1% internal oscillator for baud rate generation, and up to 55 General Purpose I/O (GPIO) pins.

Part LPC1227 is available as a dual-chip module integrating the LPC1227 with a PCF8576D LCD driver.

For additional documentation related to the LPC122x parts, see Section 26.2 “References”.

1.2 Features

- Processor core
  - ARM Cortex-M0 processor, running at frequencies of up to 45 MHz (one wait state from flash) or 30 MHz (zero wait states from flash). The LPC122x have a high score of over 45 in CoreMark CPU performance benchmark testing, equivalent to 1.51/MHz.
  - ARM Cortex-M0 built-in Nested Vectored Interrupt Controller (NVIC).
  - Serial Wire Debug (SWD).
  - System tick timer.
- Memory
  - Up to 8 kB SRAM.
  - Up to 128 kB on-chip flash programming memory.
  - In-System Programming (ISP) and In-Application Programming (IAP) via on-chip bootloader software.
  - Includes ROM-based 32-bit integer division routines.

- Clock generation unit
  - Crystal oscillator with an operating range of 1 MHz to 25 MHz.
  - 12 MHz Internal RC (IRC) oscillator trimmed to 1 % accuracy that can optionally be used as a system clock.
  - 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 or the internal RC oscillator.
  - Clock output function with divider that can reflect the system oscillator clock, IRC clock, main clock, and Watchdog clock.
  - Real-Time Clock (RTC).
- Digital peripherals
  - Micro DMA controller with 21 channels.
  - CRC engine.
  - Two UARTs with fractional baud rate generation and internal FIFO. One UART with RS-485 and modem support and one standard UART with IrDA.
  - SSP/SPI controller with FIFO and multi-protocol capabilities.
  - I^2C-bus interface supporting full I^2C-bus specification and Fast-mode Plus with a data rate of 1 Mbit/s with multiple address recognition and monitor mode. I^2C-bus pins have programmable glitch filter.
  - Up to 55 General Purpose I/O (GPIO) pins with programmable pull-up resistor, open-drain mode, programmable digital input glitch filter, and programmable input inverter.
  - Programmable output drive on all GPIO pins. Four pins support high-current output drivers.
  - All GPIO pins can be used as edge and level sensitive interrupt sources.
  - Four general purpose counter/timers with four capture inputs and four match outputs (32-bit timers) or two capture inputs and two match outputs (16-bit timers).
  - Windowed WatchDog Timer (WWDT); IEC-60335 Class B certified.
- Analog peripherals
  - One 8-channel, 10-bit ADC.
  - Two highly flexible analog comparators. Comparator outputs can be programmed to trigger a timer match signal or can be used to emulate 555 timer behavior.
- Power
  - Three reduced power modes: Sleep, Deep-sleep, and Deep power-down.
  - Processor wake-up from Deep-sleep mode via start logic using 12 port pins.
  - Processor wake-up from Deep-power down and Deep-sleep modes via the RTC.
  - Brownout detect with three separate thresholds each for interrupt and forced reset.
  - Power-On Reset (POR).
  - Integrated PMU (Power Management Unit).
- Unique device serial number for identification.
- 3.3 V power supply.
- Available as 64-pin and 48-pin LQFP package.

- Available as dual-chip module consisting of a LPC1227 single-chip microcontroller combined with a PCF8576D Universal LCD driver in a 100-pin package (part LPC12D27FBD100/301).[1]

1.3 Ordering information

Table 1. Ordering information

Type number	Package		
	Name	Description	Version
LPC12D27FBD100/301	LQFP100	plastic low profile quad flat package; 100 leads; body 14 x 14 x 1.4 mm	sot407-1
LPC1227FBD64/301	LQFP64	LQFP64: plastic low profile quad flat package; 64 leads; body 10 x 10 x 1.4 mm	SOT314-2
LPC1226FBD64/301	LQFP64	LQFP64: plastic low profile quad flat package; 64 leads; body 10 x 10 x 1.4 mm	SOT314-2
LPC1225FBD64/321	LQFP64	LQFP64: plastic low profile quad flat package; 64 leads; body 10 x 10 x 1.4 mm	SOT314-2
LPC1225FBD64/301	LQFP64	LQFP64: plastic low profile quad flat package; 64 leads; body 10 x 10 x 1.4 mm	SOT314-2
LPC1224FBD64/121	LQFP64	LQFP64: plastic low profile quad flat package; 64 leads; body 10 x 10 x 1.4 mm	SOT314-2
LPC1224FBD64/101	LQFP64	LQFP64: plastic low profile quad flat package; 64 leads; body 10 x 10 x 1.4 mm	SOT314-2
LPC1227FBD48/301	LQFP48	LQFP48: plastic low profile quad flat package; 48 leads; body 7 × 7 × 1.4 mm	SOT313-2
LPC1226FBD48/301	LQFP48	LQFP48: plastic low profile quad flat package; 48 leads; body 7 × 7 × 1.4 mm	SOT313-2
LPC1225FBD48/321	LQFP48	LQFP48: plastic low profile quad flat package; 48 leads; body 7 × 7 × 1.4 mm	SOT313-2
LPC1225FBD48/301	LQFP48	LQFP48: plastic low profile quad flat package; 48 leads; body 7 × 7 × 1.4 mm	SOT313-2
LPC1224FBD48/121	LQFP48	LQFP48: plastic low profile quad flat package; 48 leads; body 7 × 7 × 1.4 mm	SOT313-2
LPC1224FBD48/101	LQFP48	LQFP48: plastic low profile quad flat package; 48 leads; body 7 × 7 × 1.4 mm	SOT313-2

1. For details on the PCF8576D operation, see Ref. 1.

1.3.1 Part option summary

Table 2. Ordering options for LPC122x

Type number	Flash	Total SRAM	UART	I^2C/ FM+	SSP	ADC channels	GPIO	Package
LPC12D27								
LPC12D27FBD100/301	128 kB	8 kB	1	1	1	8	55	LQFP100
LPC1227								
LPC1227FBD64/301	128 kB	8 kB	2	1	1	8	55	LQFP64
LPC1227FBD48/301	128 kB	8 kB	2	1	1	8	39	LQFP48
LPC1226								
LPC1226FBD64/301	96 kB	8 kB	2	1	1	8	55	LQFP64
LPC1226FBD48/301	96 kB	8 kB	2	1	1	8	39	LQFP48
LPC1225								
LPC1225FBD64/321	80 kB	8 kB	2	1	1	8	55	LQFP64
LPC1225FBD64/301	64 kB	8 kB	2	1	1	8	55	LQFP64
LPC1225FBD48/321	80 kB	8 kB	2	1	1	8	39	LQFP48
LPC1225FBD48/301	64 kB	8 kB	2	1	1	8	39	LQFP48
LPC1224								
LPC1224FBD64/121	48 kB	4 kB	2	1	1	8	55	LQFP64
LPC1224FBD64/101	32 kB	4 kB	2	1	1	8	55	LQFP64
LPC1224FBD48/121	48 kB	4 kB	2	1	1	8	39	LQFP48
LPC1224FBD48/101	32 kB	4 kB	2	1	1	8	39	LQFP48

1.4 Block diagram

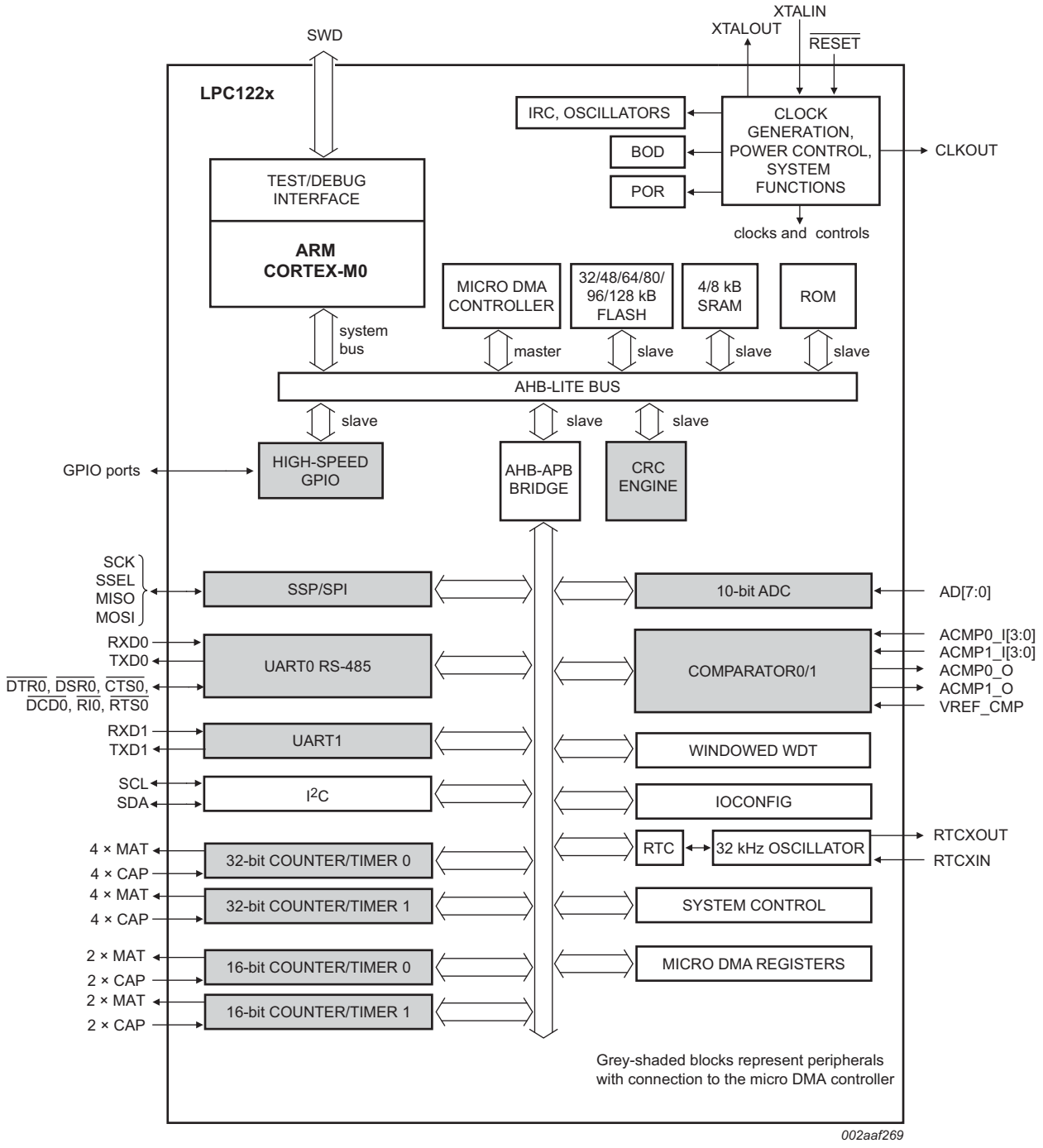


Fig 1. LPC122x block diagram

2.1 How to read this chapter

Table 3 shows the memory configuration for the LPC122x parts.

Table 3. LPC122x memory configuration

Type number	Flash	Total SRAM
LPC12D27FBD100/301	128 kB	8 kB
LPC1227FBD64/301	128 kB	8 kB
LPC1227FBD48/301	128 kB	8 kB
LPC1226FBD64/301	96 kB	8 kB
LPC1226FBD48/301	96 kB	8 kB
LPC1225FBD64/321	80 kB	8 kB
LPC1225FBD64/301	64 kB	8 kB
LPC1225FBD48/321	80 kB	8 kB
LPC1225FBD48/301	64 kB	8 kB
LPC1224FBD64/121	48 kB	4 kB
LPC1224FBD64/101	32 kB	4 kB
LPC1224FBD48/121	48 kB	4 kB
LPC1224FBD48/101	32 kB	4 kB

2.2 Introduction

Figure 2 shows the memory and peripheral address space of the LPC122x. The AHB peripheral area is 2 MB in size and is divided to allow for up to 128 peripherals.

On the LPC122x, the GPIO ports, the CRC engine, and the micro DMA controller are AHB peripherals. The APB peripheral area is 512 kB in size and is divided to allow for up to 32 peripherals. Each peripheral of either type is allocated 16 kB of space. This allows simplifying the address decoding for each peripheral.

All peripheral register addresses are 32-bit word aligned regardless of their size. An implication of this is that word and half-word registers must be accessed all at once. For example, it is not possible to read or write the upper byte of a word register separately.

2.3 Memory allocation

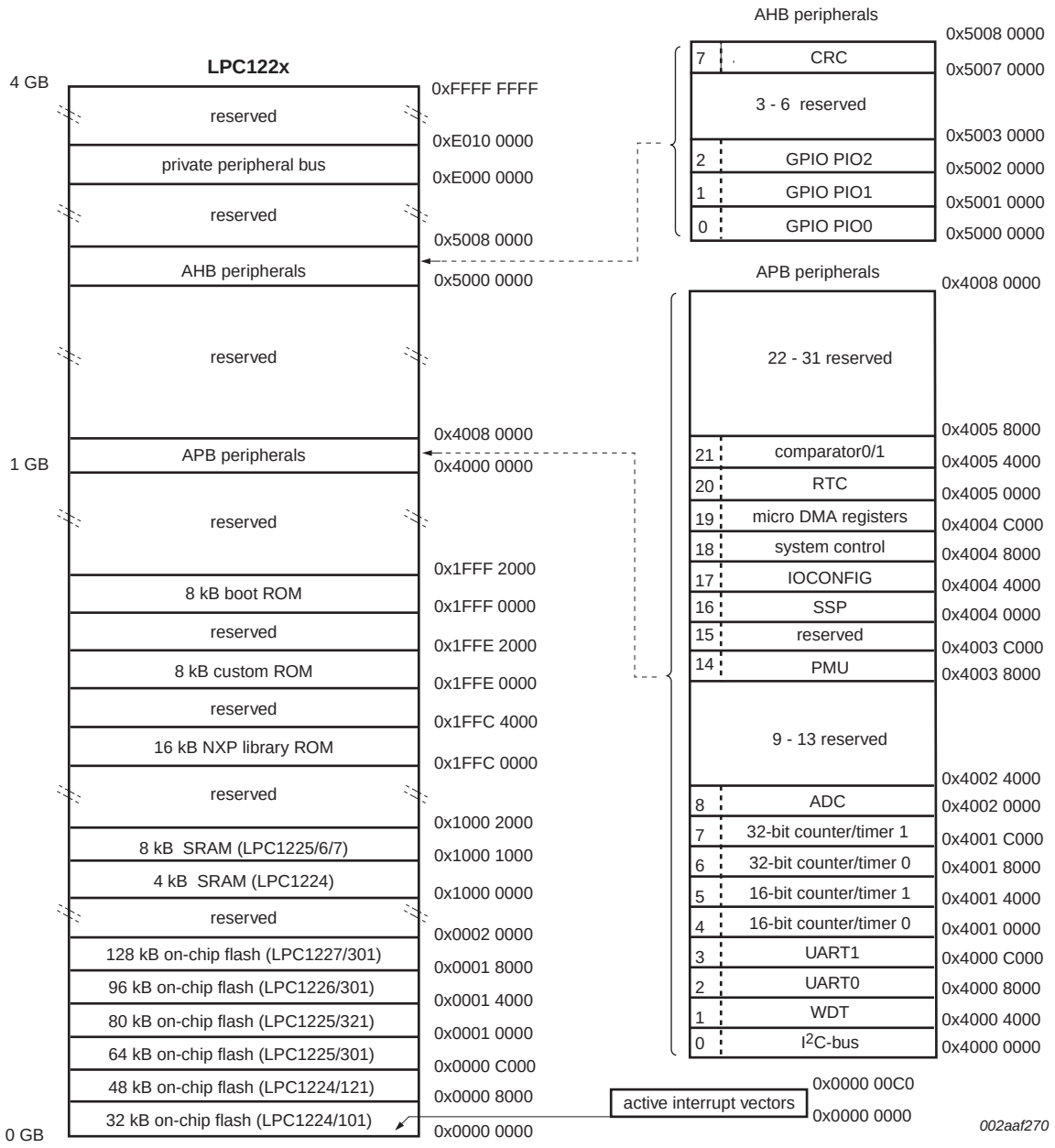


Fig 2. LPC122x memory map

3.1 How to read this chapter

The NVIC is part of the ARM Cortex-M0 system block is identical on all LPC122x parts.

3.2 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.
- Supports 32 vectored interrupts.
- Four programmable interrupt priority levels with hardware priority level masking.
- Software interrupt generation.
- Non-Maskable Interrupt (NMI) with configurable source.

3.3 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.

The source of the NMI is configurable (see Section 4.5.29). Multiple peripheral interrupts can be selected for the NMI functionality.

Refer to the ARM Cortex-M0 Technical Reference Manual and to the ARM Cortex-M0 appendix (Section 25.5.2) for details of NVIC operation and the register description.

3.4 Interrupt sources

Table 4 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. There is no significance or priority about what line is connected where, except for certain standards from ARM.

Table 4. Connection of interrupt sources to the Vectored Interrupt Controller

Exception Number	Function	Flag(s)
11 to 0	start logic wake-up interrupts	Each interrupt is connected to a PIO input pin serving as wake-up pin when the part is in Deep-sleep mode; Interrupt 0 to 11 correspond to PIO0_0 to PIO0_11(see Table 37).
12	I^2C	SI (state change)
13	CT16B0	Match 3 to 0 Capture 3 to 0

Table 4. Connection of interrupt sources to the Vectored Interrupt Controller

Exception Number	Function	Flag(s)
14	CT16B1	Match 3 to 0 Capture 3 to 0
15	CT32B0	Match 3 to 0 Capture 3 to 0
16	CT32B1	Match 3 to 0 Capture 3 to 0
17	SSP	Tx FIFO half empty Rx FIFO half full Rx Timeout Rx Overrun
18	UART0	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) End of Auto-Baud (ABEO) Auto-Baud Time-Out (ABTO) Modem interrupt
19	UART1	Rx Line Status (RLS) Transmit Holding Register Empty (THRE) Rx Data Available (RDA) Character Time-out Indicator (CTI) End of Auto-Baud (ABEO) Auto-Baud Time-Out (ABTO)
20	Comparator	Comparator 0/1 interrupt
21	ADC	A/D Converter end of conversion
22	WDT	Watchdog interrupt (WDINT)
23	BOD	Brown-out detect
24	-	Reserved
25	PIO0	GPIO interrupt status of port 0
26	PIO1	GPIO interrupt status of port 1
27	PIO2	GPIO interrupt status of port 2
28	-	Reserved
29	DMA	DMA request interrupt
30	RTC	RTC interrupt
31	-	Reserved

4.1 How to read this chapter

The system control block is identical on all LPC122x parts.

4.2 Introduction

The system configuration block controls oscillators, start logic, and clock generation of the LPC122x. Also included in this block are registers for setting the priority for AHB access and a register for remapping flash, SRAM, and ROM memory areas.

4.3 Pin description

Table 5 shows pins that are associated with system control block functions.

Table 5. Pin summary

Pin name	Pin direction	Pin description
CLKOUT	O	Clockout pin
PIO0_0 to PIO0_11	I	Start logic wake-up pins port 0

4.4 General description

See Figure 3 for an overview of the LPC122x Clock Generation Unit (CGU).

Following reset, the LPC122x will operate from the Internal RC oscillator until switched by software. This allows systems to operate without any external crystal and the boot loader code to operate at a known frequency.

The SYSAHBCLKCTRL register gates the system clock to the various peripherals and memories. UART0/1, SSP, the RTC, and the SysTick timer have individual clock dividers to derive peripheral clocks from the main clock.

The watchdog clock can be derived from the oscillator output or the main clock.

The main clock, and the clock outputs from the IRC, the system oscillator, and the watchdog oscillator can be observed directly on the CLKOUT pin.

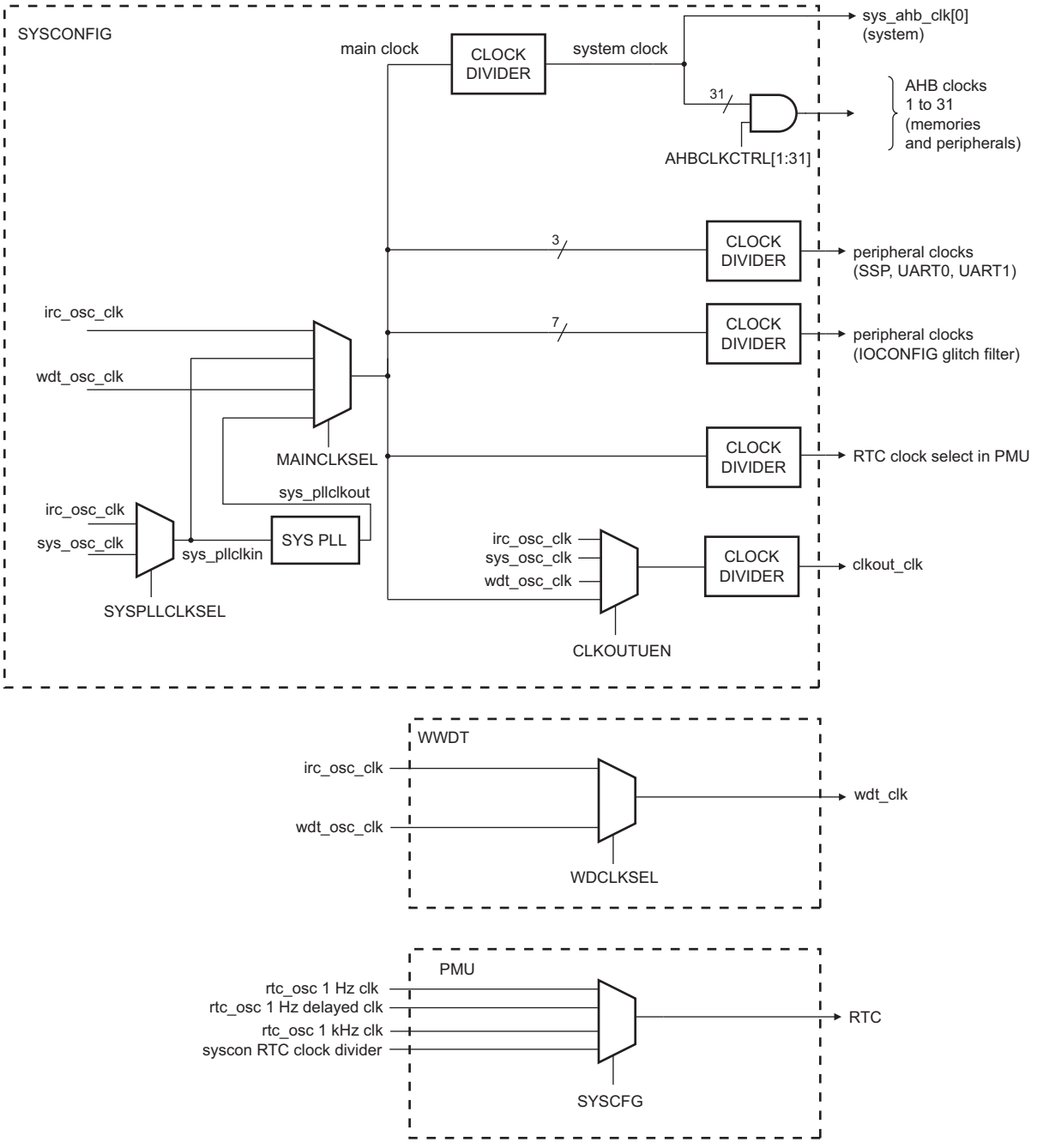


Fig 3. LPC122x CGU block diagram

4.5 Register description

All registers, regardless of size, are on word address boundaries. Details of the registers appear in the description of each function.

Table 6. Register overview: system control block (base address 0x4004 8000)

Name	Access	Address offset	Description	Reset value
SYSMEMREMAP	R/W	0x000	System memory remap	0x0000 0000
PRESETCTRL	R/W	0x004	Peripheral reset control and flash timing overwrite	0x0000 FFFF
SYSPLLCTRL	R/W	0x008	System PLL control	0x0000 0000
SYSPLLSTAT	R	0x00C	System PLL status	0x0000 0000
-	-	0x010 - 0x01C	Reserved	-
SYSOSCCTRL	R/W	0x020	System oscillator control	0x0000 0000
WDTOSCCTRL	R/W	0x024	Watchdog oscillator control	0x0000 0000
IRCCTRL	R/W	0x028	IRC control	0x0000 0080
-	-	0x02C	Reserved	-
SYSRESSTAT	R/W	0x030	System reset status register	0x0000 0000
-	-	0x034 - 0x03C	Reserved	-
SYSPLLCLKSEL	R/W	0x040	System PLL clock source select	0x0000 0000
SYSPLLCLKUEN	R/W	0x044	System PLL clock source update enable	0x0000 0000
-	-	0x048 - 0x06C	Reserved	-
MAINCLKSEL	R/W	0x070	Main clock source select	0x0000 0000
MAINCLKUEN	R/W	0x074	Main clock source update enable	0x0000 0000
SYSAHBCLKDIV	R/W	0x078	System AHB clock divider	0x0000 0001
-	-	0x07C	Reserved	-
SYSAHBCLKCTRL	R/W	0x080	System AHB clock control	0xF01F FFFF
-	-	0x084 - 0x08C	Reserved	-
-	-	0x090	Reserved	-
SSPCLKDIV	R/W	0x094	SSP clock divder	0x0000 0000
UART0CLKDIV	R/W	0x098	UART0 clock divider	0x0000 0000
UART1CLKDIV	R/W	0x09C	UART1 clock divider	0x0000 0000
RTCCLKDIV	R/W	0x0A0	RTC clock divider	0x0000 0000
-	-	0x0A4- 0x0DC	Reserved	-
CLKOUTCLKSEL	R/W	0x0E0	CLKOUT clock source select	0x0000 0000
CLKOUTUEN	R/W	0x0E4	CLKOUT clock source update enable	0x0000 0000
CLKOUTDIV	R/W	0x0E8	CLKOUT clock divider	0x0000 0000
-	-	0x0EC - 0x0FC	Reserved	-
PIOPORCAP0	R	0x100	POR captured PIO status 0	user dependent
PIOPORCAP1	R	0x104	POR captured PIO status 1	user dependent

Table 6. Register overview: system control block (base address 0x4004 8000) *…continued*

Name	Access	Address offset	Description	Reset value
-	-	0x108 - 0x130	Reserved	-
IOCONFIGCLKDIV6	R/W	0x134	Peripheral clock 6 to the IOCONFIG block for programmable glitch filter	0x0000 0000
IOCONFIGCLKDIV5	R/W	0x138	Peripheral clock 5 to the IOCONFIG block for programmable glitch filter	0x0000 0000
IOCONFIGCLKDIV4	R/W	0x13C	Peripheral clock 4 to the IOCONFIG block for programmable glitch filter	0x0000 0000
IOCONFIGCLKDIV3	R/W	0x140	Peripheral clock 3 to the IOCONFIG block for programmable glitch filter	0x0000 0000
IOCONFIGCLKDIV2	R/W	0x144	Peripheral clock 2 to the IOCONFIG block for programmable glitch filter	0x0000 0000
IOCONFIGCLKDIV1	R/W	0x148	Peripheral clock 1 to the IOCONFIG block for programmable glitch filter	0x0000 0000
IOCONFIGCLKDIV0	R/W	0x14C	Peripheral clock 0 to the IOCONFIG block for programmable glitch filter	0x0000 0000
BODCTRL	R/W	0x150	BOD control	not affected by system reset
SYSTCKCAL	R/W	0x154	System tick counter calibration	0x0000 001F
AHBPRIO	-	0x158	AHB priority setting	0x0000 0004
-	-	0x15C - 0x16C	Reserved	-
IRQLATENCY	R/W	0x170	IQR delay. Allows trade-off between interrupt latency and determinism.	0x0000 0010
INTNMI	R/W	0x174	NMI interrupt source configuration control	0x0000 003F
-	-	0x178 - 0x1FC	Reserved	-
STARTAPRP0	R/W	0x200	Start logic edge control register 0	0x0000 0000
STARTERP0	R/W	0x204	Start logic signal enable register 0	0x0000 0000
STARTRSRP0CLR	W	0x208	Start logic reset register 0	0x0000 0000
STARTSRP0	R	0x20C	Start logic status register 0	0x0000 0000
STARTAPRP1	R/W	0x210	Start logic edge control register 1; peripheral interrupts	0x0000 0000
STARTERP1	R/W	0x214	Start logic signal enable register 1; peripheral interrupts	0x0000 0000
STARTRSRP1CLR	W	0x218	Start logic reset register 1; peripheral interrupts	0x0000 0000
STARTSRP1	R	0x21C	Start logic status register 1; peripheral interrupts	0x0000 0000
-	-	0x220 - 0x22C	Reserved	-
PDSLEEPCFG	R/W	0x230	Power-down states in Deep-sleep mode	0x0000 FFFF

Table 6. Register overview: system control block (base address 0x4004 8000) *…continued*

Name	Access	Address offset	Description	Reset value
PDAWAKECFG	R/W	0x234	Power-down states after wake-up from Deep-sleep mode	0x0000 FDF0
PDRUNCFG	R/W	0x238	Power-down configuration register	0x0000 FDF0
-	-	0x23C - 0x3F0	Reserved	-
DEVICE_ID	R	0x3F4	Device ID	part dependent

Remark: The flash configuration block resides in its own memory area but is listed together with the system control registers.

Table 7. Register overview: flash configuration (base address 0x5006 0000)

Name	Access	Address offset	Description	Reset value
FLASHCFG	R/W	0x028	Flash configuration register	0x0000 0000

4.5.1 System memory remap register

The system memory remap register selects whether the ARM interrupt vectors are read from the boot ROM, the flash, or the SRAM.

Table 8. 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.	00
		0x0	Boot Loader 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	0x00

4.5.2 Peripheral reset control register

This register allows software to reset individual peripherals. If a bit in this register is set to 0, the corresponding peripheral is reset. Writing a 1 de-asserts the reset.

Bit 15 of this register overwrites the flash timing for read access.

Table 9. Peripheral reset control register (PRESETCTRL, address 0x4004 8004) bit description

Bit	Symbol	Value	Description	Reset value
0	SSP_RST_N		SSP reset control	1
		0	SSP reset enabled	
		1	SSP reset de-asserted	
1	I2C_RST_N		I2C reset control	1
		0	I2C reset enabled	
		1	I2C reset de-asserted	
2	UART0_RST_N		UART0 reset control	1
		0	UART0 reset enabled	
		1	UART0 reset de-asserted	
3	UART1_RST_N		UART1 reset control	1
		0	UART1 reset enabled	
		1	UART1 reset de-asserted	
4	CT16B0_RST_N		16-bit counter/timer 0 (CT16B0) reset control	1
		0	CT16B0 reset enabled	
		1	CT16B0 reset de-asserted	
5	CT16B1_RST_N		16-bit counter/timer 1 (CT16B1) reset control	1
		0	CT16B1 reset enabled	
		1	CT16B1 reset de-asserted	
6	CT32B0_RST_N		32-bit counter/timer 0 (CT32B0) reset control	1
		0	CT32B0 reset enabled	
		1	CT32B0 reset de-asserted	
7	CT32B1_RST_N		32-bit counter/timer 1 (CT32B1) reset control	1
		0	CT32B1 reset enabled	
		1	CT32B1 reset de-asserted	
8	CMP_RST_N		Comparator reset control	1
		0	Comparator reset enabled	
		1	Comparator reset de-asserted	
9	CRC_RST_N		CRC reset control	1
		0	CRC reset enabled	
		1	CRC reset de-asserted	
10	DMA_RST_N		Micro DMA reset control	1
		0	DMA reset enabled	
		1	DMA reset de-asserted	
14:11	-	-	Reserved	-

Table 9. Peripheral reset control register (PRESETCTRL, address 0x4004 8004) bit description

Bit	Symbol	Value	Description	Reset value
15	FLASH_OVERRIDE		Flash read mode. The default is 1-cycle flash read access. At higher operating frequencies (above 30 MHz) the multi-cycle read mode must be used to allow 2, 3, 4, or 5-cycle read configuration. If multi-cycle read mode is selected, the number of cycles can be configured in the FLASHCFG register (see Table 52).	1
		0	Flash multi-cycle read mode (for frequencies greater than 30 MHz).	
		1	Flash 1-cycle read mode (for frequencies 30 MHz or below).	
31:16	-	-	Reserved	-

4.5.3 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 up to a high frequency, 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.

Table 10. 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	00000
6:5	PSEL		Post divider ratio P. The division ratio is 2 × P.	00
		0x0	P = 1	
		0x1	P = 2	
		0x2	P = 4	
		0x3	P = 8	
31:7	-	-	Reserved	0x00

4.5.4 System PLL status register

This register is a Read-only register and supplies the PLL lock status (see Section 4.10.1).

Table 11. 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	0x00

4.5.5 System oscillator control register

This register configures the frequency range for the system oscillator.

Table 12. System oscillator control register (SYSOSCCTRL, address 0x4004 8020) bit description

Bit	Symbol	Value	Description	Reset value
0	BYPASS		Bypass system oscillator	0
		0	Oscillator is not bypassed.	
		1	Bypass enabled. PLL input (sys_osc_clk) is fed directly from the XTALIN and XTALOUT pins.	
1	FREQRANGE		Determines frequency range for Low-power oscillator.	0
		0	1 - 20 MHz frequency range.	
		1	15 - 25 MHz frequency range	
31:2	-	-	Reserved	0x00

4.5.6 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 500 kHz and 3.4 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 $\times$ (1 + DIVSEL)) = 7.8 kHz to 1.7 MHz (nominal values).

Remark: Any setting of the FREQSEL bits will yield a Fclkana value within $\pm$ 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 clock.

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 13. 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).	0
		0x1	0.5 MHz	
		0x2	0.8 MHz	
		0x3	1.1 MHz	
		0x4	1.4 MHz	
		0x5	1.6 MHz	
		0x6	1.8 MHz	
		0x7	2.0 MHz	
		0x8	2.2 MHz	
		0x9	2.4 MHz	
		0xA	2.6 MHz	
		0xB	2.7 MHz	
		0xC	2.9 MHz	
		0xD	3.1 MHz	
		0xE	3.2 MHz	
		0xF	3.4 MHz	
31:9	-	-	Reserved	-

4.5.7 Internal resonant crystal control register

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

Table 14. Internal resonant crystal control register (IRCCTRL, address 0x4004 8028) bit description

Bit	Symbol	Description	Reset value
7:0	TRIM	Trim value	0x000 0080, then flash will reprogram
31:9	-	Reserved	0x00

4.5.8 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, but if another reset signal (e.g., EXTRST) remains asserted after the POR signal is negated, then its bit is set to detected.

Table 15. System reset status register (SYSRESSTAT, address 0x4004 8030) bit description

Bit	Symbol	Value	Description	Reset value
0	POR		POR reset status	0
		0	no POR detected	
		1	POR detected	
1	EXTRST		reset status	0
		0	no $\overline{\text{RESET}}$ event detected	
		1	$\overline{\text{RESET}}$ detected	
2	WDT		Status of the Watchdog reset	0
		0	no WDT reset detected	
		1	WDT reset detected	
3	BOD		Status of the Brown-out detect reset	0
		0	no BOD reset detected	
		1	BOD reset detected	
4	SYSRST		Status of the software system reset.The ARM software reset has the same effect as the hardware reset using the $\overline{\text{RESET}}$ pin.	0
		0	no System reset detected	
		1	System reset detected	
31:5	-	-	Reserved	0x00

4.5.9 System PLL clock source select register

This register selects the clock source for the system PLL. The SYSPLLCLKUEN register (see Section 4.5.10) must be toggled from LOW to HIGH for the update to take effect.

Remark: When switching clock sources, both clocks must be running before updating the clock source.

Table 16. 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	0x00
		0x0	IRC oscillator	
		0x1	System oscillator	
		0x2	Reserved	
		0x3	Reserved	
31:2	-	-	Reserved	0x00

4.5.10 System PLL clock source update enable 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 17. 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	0x00

4.5.11 Main clock source select register

This register selects the main system clock which can be either any input to the system PLL, the output from the system PLL (sys_pllclkout), or the watchdog or IRC oscillators directly. The main system clock clocks the core, the peripherals, and the memories.

The MAINCLKUEN register (see Section 4.5.12) must be toggled from LOW to HIGH for the update to take effect.

Remark: When switching clock sources, both clocks must be running before updating the clock source.

Table 18. 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	00
		0x0	IRC oscillator	
		0x1	Input clock to system PLL	
		0x2	WDT oscillator	
		0x3	System PLL clock out	
31:2	-	-	Reserved	0x00

4.5.12 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 the MAINCLKUEN register and then write a one to MAINCLKUEN.

Table 19. 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	0x0
		0	No change	
		1	Update clock source	
31:1	-	-	Reserved	0x00

4.5.13 System AHB clock divider register

This register divides the main clock to provide the system clock to the core, memories, and the peripherals. The system clock can be shut down completely by setting the DIV bits to 0x0.

Table 20. System AHB 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.	1
31:8	-	Reserved	0x00

4.5.14 System AHB clock control register

The AHBCLKCTRL register enables the clocks to individual system and peripheral blocks. The system clock (sys_ahb_clk[0], bit 0 in the AHBCLKCTRL register) provides the clock for the AHB to APB bridge, the AHB matrix, the ARM Cortex-M0, the SYSCON block, and the PMU. This clock cannot be disabled.

Table 21. System AHB clock control register (SYSAHBCLKCTRL, address 0x4004 8080) bit description

Bit	Symbol	Value	Description	Reset value
0	SYS		Enables clock for AHB to APB bridge, to the AHB matrix, to the Cortex-M0 FCLK and HCLK, to the SysCon, and to the PMU. This bit is read only.	1
		0	Reserved	
		1	Enable	
1	ROM		Enables clock for ROM.	1
		0	Disable	
		1	Enable	
2	RAM		Enables clock for RAM.	1
		0	Disable	
		1	Enable	
3	FLASHREG		Enables clock for flash controller registers.	1
		0	Disable	
		1	Enable	
4	FLASHARRAY		Enables clock for flash array.	1
		0	Disable	
		1	Enable	
5	I2C		Enables clock for I2C.	1
		0	Disable	
		1	Enable	
6	CRC		Enables clock for CRC.	1
		0	Disable	
		1	Enable	

Table 21. System AHB clock control register (SYSAHBCLKCTRL, address 0x4004 8080) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
7	CT16B0		Enables clock for 16-bit counter/timer 0.	1
		0	Disable	
		1	Enable	
8	CT16B1		Enables clock for 16-bit counter/timer 1.	1
		0	Disable	
		1	Enable	
9	CT32B0		Enables clock for 32-bit counter/timer 0.	1
		0	Disable	
		1	Enable	
10	CT32B1		Enables clock for 32-bit counter/timer 1.	1
		0	Disable	
		1	Enable	
11	SSP		Enables clock for SSP.	1
		0	Disable	
		1	Enable	
12	UART0		Enables clock for UART0. Note that the UART0 pins must be configured in the IOCON block before the UART0 clock can be enabled.	1
		0	Disable	
		1	Enable	
13	UART1		Enables clock for UART1. Note that the UART1 pins must be configured in the IOCON block before the UART1 clock can be enabled.	1
		0	Disable	
		1	Enable	
14	ADC		Enables clock for ADC.	1
		0	Disable	
		1	Enable	
15	WDT		Enables clock for WDT.	1
		0	Disable	
		1	Enable	
16	IOCON		Enables clock for IO configuration block.	1
		0	Disable	
		1	Enable	
17	DMA		Enables clock for micro DMA.	1
		0	Disable	
		1	Enable	
18	-		Reserved. Write as zero.	1

Table 21. System AHB clock control register (SYSAHBCLKCTRL, address 0x4004 8080) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
19	RTC		Enables clock for RTC. **Remark:** If the RTC clock source is disabled, select the RTC clock source first, before re-enabling the RTC through this bit (see Table 58).	1
		0	Disable	
		1	Enable	
20	CMP		Enables clock for comparator.	1
		0	Disable	
		1	Enable	
27:21	-	-	Reserved	0
28	-	-	Reserved. Write as zero.	1
29	GPIO2		Enables clock for GPIO port 2.	1
		0	Disable	
		1	Enable	
30	GPIO1		Enables clock for GPIO port 1.	1
		0	Disable	
		1	Enable	
31	GPIO0		Enables clock for GPIO port 0.	1
		0	Disable	
		1	Enable	

4.5.15 SSP clock divider register

This register configures the SSP peripheral clock SSP_PCLK. The SSP_PCLK can be shut down by setting the DIV bits to 0x0.

Table 22. SSP clock divider register (SSPCLKDIV, address 0x4004 8094) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	SSP0_PCLK clock divider values 0: Disable SSP0_PCLK. 1: Divide by 1. to 255: Divide by 255.	0
31:8	-	Reserved	0x00

4.5.16 UART0 clock divider register

This register configures the UART0 peripheral clock UART0_PCLK. The UART0_PCLK can be shut down by setting the DIV bits to 0x0.

Remark: Note that the UART0 pins must be configured in the IOCON block before the UART0 clock can be enabled.

Table 23. UART0 clock divider register (UART0CLKDIV, address 0x4004 8098) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	UART0_PCLK clock divider values 0: Disable UART0_PCLK. 1: Divide by 1. to 255: Divide by 255.	0
31:8	-	Reserved	0x00

4.5.17 UART1 clock divider register

This register configures the UART1 peripheral clock UART1_PCLK. The UART1_PCLK can be shut down by setting the DIV bits to 0x0.

Remark: Note that the UART1 pins must be configured in the IOCON block before the UART1 clock can be enabled.

Table 24. UART1 clock divider register (UART1CLKDIV, address 0x4004 809C) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	UART1_PCLK clock divider values 0: Disable UART1_PCLK. 1: Divide by 1. to 255: Divide by 255.	0
31:8	-	Reserved	0x00

4.5.18 RTC clock divider register

This register configures the RTC peripheral clock RTC_PCLK. The RTC_PCLK can be shut down by setting the DIV bits to 0x0.

Table 25. RTC clock divider register (RTCCLKDIV, address 0x4004 80A0) bit description

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

4.5.19 CLKOUT clock source select register

This register configures the clkout_clk signal to be output on the CLKOUT pin. All three oscillators and the main clock can be selected for the clkout_clk clock.

The CLKOUTCLKUEN register (see Section 4.5.20) must be toggled from LOW to HIGH for the update to take effect.

Table 26. CLKOUT clock source select register (CLKOUTCLKSEL, address 0x4004 80E0) bit description

Bit	Symbol	Value	Description	Reset value
1:0	SEL		CLKOUT clock source	00
		0x0	IRC oscillator	
		0x1	System oscillator	
		0x2	Watchdog oscillator	
		0x3	Main clock	
31:2	-	-	Reserved	0x00

4.5.20 CLKOUT clock source update enable register

This register updates the clock source of the CLKOUT pin with the new clock after the CLKOUTCLKSEL 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 the CLKOUTUEN register and then write a one to CLKOUTUEN.

Table 27. 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	0x00

4.5.21 CLKOUT clock divider register

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

Table 28. CLKOUT clock divider registers (CLKOUTDIV, address 0x4004 80E8) bit description

Bit	Symbol	Description	Reset value
7:0	DIV	Clock output divider values 0: Disable CLKOUT. 1: Divide by 1. to 255: Divide by 255.	0
31:8	-	Reserved	0x00

4.5.22 POR captured PIO status register 0

The PIOPORCAP0 register captures the state (HIGH or LOW) of the PIO pins of ports 0,1, and 2 (pins PIO2_0 to PIO2_7) at power-on-reset. Each bit represents the reset state of one GPIO pin. This register is a read-only status register.

Table 29. POR captured PIO status registers 0 (PIOPORCAP0, address 0x4004 8100) bit description

Bit	Symbol	Description	Reset value
31:0	PIOSTAT	Raw reset status input PIO0_31 to PIO2_7	User implementation dependent

4.5.23 POR captured PIO status register 1

The PIOPORCAP1 register captures the state (HIGH or LOW) of the PIO pins of port 2 (PIO2_8 to PIO2_11) at power-on-reset. Each bit represents the reset state of one PIO pin. This register is a read-only status register.

Table 30. POR captured PIO status registers 1 (PIOPORCAP1, address 0x4004 8104) bit description

Bit	Symbol	Description	Reset value
31:0	PIOSTAT	Raw reset status input PIO2_8 to PIO2_11	User implementation dependent

4.5.24 IOCONFIG clock divider registers 0 to 6

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

Table 31. IOCONFIG filter clock divider registers 0 to 6 (IOCONFIGCLKDIV0 to IOCONFIGCLKDIV6, address 0x4004 814C to 0x4004 8134) bit description

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

4.5.25 BOD control register

The BOD control register selects three separate threshold values for sending a BOD interrupt to the NVIC and for forced reset. Reset and interrupt threshold values listed in Table 32 are typical values.

The reset value of the BODCTRL register is not affected by a system reset. Instead, the BODCTRL register retains its last programmed value.

The BOD control circuit is only reset by POR.

Table 32. BOD control register (BODCTRL, address 0x4004 8150) bit description

Bit	Symbol	Value	Description	Reset value
1:0	BODRSTLEV		BOD reset level	-
		0x0	Reserved. Writing 00 to these bits will not change the BOD reset level.	
		0x1	Level 1: The reset assertion threshold voltage is 2.038 V; the reset de-assertion threshold voltage is 2.180 V.	
		0x2	Level 2: The reset assertion threshold voltage is 2.336 V; the reset de-assertion threshold voltage is 2.471 V.	
		0x3	Level 3: The reset assertion threshold voltage is 2.624 V; the reset de-assertion threshold voltage is 2.761 V.	
3:2	BODINTVAL		BOD interrupt level	-
		0x0	Reserved. Writing 00 to these bits will not change the interrupt level.	
		0x1	Level 1:The interrupt assertion threshold voltage is 2.248 V; the interrupt de-assertion threshold voltage is 2.385 V.	
		0x2	Level 2: The interrupt assertion threshold voltage is 2.541 V; the interrupt de-assertion threshold voltage is 2.669 V.	
		0x3	Level 3: The interrupt assertion threshold voltage is 2.828 V; the interrupt de-assertion threshold voltage is 2.929 V.	
4	BODRSTENA		BOD reset enable	-
		0	Disable reset function.	
		1	Enable reset function.	
31:5	-	-	Reserved	-

4.5.26 System tick counter calibration register

This register determines the value of the SYST_CALIB register (see Table 254).

Table 33. System tick timer calibration register (SYSTCKCAL, address 0x4004 8154) bit description

Bit	Symbol	Description	Reset value
25:0	CAL	System tick timer calibration value	0x00
31:26	-	Reserved	0x00

4.5.27 AHB matrix master priority register

The AHB matrix master priority register determines the priority with which the AHB masters can access the bus.

Table 34. AHB matrix master priority register (AHBPRIO, address 0x4004 8158) bit description

Bit	Symbol	Description	Reset value
1:0	M0PRIO	Priority of the ARM Cortex-M0 core	00
3:2	DMAPRIO	Priority of the micro DMA controller	01
5:4	-	Reserved	-
31:6	-	Reserved	-

4.5.28 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. System factors external to the cpu, however, bus latencies, peripheral response times, etc. 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 35. 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	-

4.5.29 NMI interrupt source configuration register

This register configures a source for the ARM Cortex-M0 Non-Maskable Interrupt (NMI). See Section 25.5.2.

Table 36. NMI interrupt source configuration register (INTNMI, address 0x4004 8174) bit description

Bit	Symbol	Description	Reset value
5:0	NMISRC	NMI interrupt source select. Values 0x13 to 0x3E are reserved. 0x0 = I2C interrupt. 0x1 = CT16B0 interrupt. 0x2 = CT16B1 interrupt. 0x3 = CT32B0 interrupt. 0x4 = CT32B1 interrupt. 0x5 = SSP interrupt. 0x6 = UART0 interrupt. 0x7 = UART1 interrupt. 0x8 = Comparator interrupt. 0x9 = ADC interrupt. 0xA = Watchdog timer interrupt. 0xB = I2C interrupt. 0xC = Reserved. 0xD = PIO0 interrupt. 0xE = PIO1 interrupt. 0xF = PIO2 interrupt. 0x10 = PMU wake-up interrupt. 0x11 = DMA interrupt. 0x12 = RTC interrupt. 0x3F = NMI disabled.	-
31:6	-	Reserved	0x00

4.5.30 Start logic edge control register 0

The STARTAPRP0 register controls the start logic inputs of ports 0 (PIO0_0 to PIO0_11). This register selects a falling or rising edge on the corresponding PIO input to produce a falling or rising clock edge, respectively, for the start logic (see Section 4.8.2).

Every bit in the STARTAPRP0 register controls one port input and is connected to one wake-up interrupt in the NVIC. Bit 0 in the STARTAPRP0 register corresponds to interrupt 0, bit 1 to interrupt 1, ... , up to a total of 12 interrupts. Through this start logic register set, external pins can be used to wake up the microcontroller from Deep-sleep mode.

Remark: Each interrupt connected to a start logic input must be enabled in the NVIC if the corresponding PIO pin is used to wake up the microcontroller from Deep-sleep mode.

Table 37. Start logic edge control register 0 (STARTAPRP0, address 0x4004 8200) bit description

Bit	Symbol	Description	Reset value
0	APRPIO0_0	Edge select for start logic input n for pin PIO0_0. 0 = Falling edge 1 = Rising edge	0
1	APRPIO0_1	Edge select for start logic input n for pin PIO0_1. 0 = Falling edge 1 = Rising edge	
2	APRPIO0_2	Edge select for start logic input n for pin PIO0_2. 0 = Falling edge 1 = Rising edge	

Table 37. Start logic edge control register 0 (STARTAPRP0, address 0x4004 8200) bit description *…continued*

Bit	Symbol	Description	Reset value
3	APRPIO0_3	Edge select for start logic input n for pin PIO0_3. 0 = Falling edge 1 = Rising edge	
4	APRPIO0_4	Edge select for start logic input n for pin PIO0_4. 0 = Falling edge 1 = Rising edge	
5	APRPIO0_5	Edge select for start logic input n for pin PIO0_5. 0 = Falling edge 1 = Rising edge	
6	APRPIO0_6	Edge select for start logic input n for pin PIO0_6. 0 = Falling edge 1 = Rising edge	
7	APRPIO0_7	Edge select for start logic input n for pin PIO0_7. 0 = Falling edge 1 = Rising edge	
8	APRPIO0_8	Edge select for start logic input n for pin PIO0_8. 0 = Falling edge 1 = Rising edge	
9	APRPIO0_9	Edge select for start logic input n for pin PIO0_9. 0 = Falling edge 1 = Rising edge	
10	APRPIO0_10	Edge select for start logic input n for pin PIO0_10. 0 = Falling edge 1 = Rising edge	
11	APRPIO0_11	Edge select for start logic input n for pin PIO0_11. 0 = Falling edge 1 = Rising edge	
31:12	-	Reserved. Do not write a 1 to reserved register bits.	-

4.5.31 Start logic signal enable register 0

This STARTERP0 register enables or disables the start signal bits in the start logic. The bit assignment is identical to Table 37.

Table 38. Start logic signal enable register 0 (STARTERP0, address 0x4004 8204) bit description

Bit	Symbol	Description	Reset value
0	ERPIO0_0	Enable start signal for start logic input n for pin PIO0_0. 0 = Disabled. 1 = Enabled.	0
1	ERPIO0_1	Enable start signal for start logic input n for pin PIO0_1. 0 = Disabled. 1 = Enabled.	0
2	ERPIO0_2	Enable start signal for start logic input n for pin PIO0_2. 0 = Disabled. 1 = Enabled.	0

Table 38. Start logic signal enable register 0 (STARTERP0, address 0x4004 8204) bit description *…continued*

Bit	Symbol	Description	Reset value
3	ERPIO0_3	Enable start signal for start logic input n for pin PIO0_3. 0 = Disabled. 1 = Enabled.	0
4	ERPIO0_4	Enable start signal for start logic input n for pin PIO0_4. 0 = Disabled. 1 = Enabled.	0
5	ERPIO0_5	Enable start signal for start logic input n for pin PIO0_5. 0 = Disabled. 1 = Enabled.	0
6	ERPIO0_6	Enable start signal for start logic input n for pin PIO0_6. 0 = Disabled. 1 = Enabled.	0
7	ERPIO0_7	Enable start signal for start logic input n for pin PIO0_7. 0 = Disabled. 1 = Enabled.	0
8	ERPIO0_8	Enable start signal for start logic input n for pin PIO0_8. 0 = Disabled. 1 = Enabled.	0
9	ERPIO0_9	Enable start signal for start logic input n for pin PIO0_9. 0 = Disabled. 1 = Enabled.	0
10	ERPIO0_10	Enable start signal for start logic input n for pin PIO0_10. 0 = Disabled. 1 = Enabled.	0
11	ERPIO0_11	Enable start signal for start logic input n for pin PIO0_11. 0 = Disabled. 1 = Enabled.	0
31:12	-	Reserved. Do not write a 1 to reserved register bits.	-

4.5.32 Start logic reset register 0

Writing a one to a bit in the STARTRSRP0CLR register resets the start logic state. The bit assignment is identical to Table 37. The start-up logic uses the input signals to generate a clock edge for registering a start signal. This clock edge (falling or rising) sets the interrupt for waking up from Deep-sleep mode. Therefore, the start-up logic states must be cleared before being used.

Table 39. Start logic reset register 0 (STARTRSRP0CLR, address 0x4004 8208) bit description

Bit	Symbol	Description	Reset value
0	RSRPIO0_0	Start signal reset for start logic input n for pin PIO0_0. 0 = No effect. 1 = Write: reset start signal.	0
1	RSRPIO0_1	Start signal reset for start logic input n for pin PIO0_1. 0 = No effect. 1 = Write: reset start signal.	0
2	RSRPIO0_2	Start signal reset for start logic input n for pin PIO0_2. 0 = No effect. 1 = Write: reset start signal.	0
3	RSRPIO0_3	Start signal reset for start logic input n for pin PIO0_3. 0 = No effect. 1 = Write: reset start signal.	0
4	RSRPIO0_4	Start signal reset for start logic input n for pin PIO0_4. 0 = No effect. 1 = Write: reset start signal.	0
5	RSRPIO0_5	Start signal reset for start logic input n for pin PIO0_5. 0 = No effect. 1 = Write: reset start signal.	0
6	RSRPIO0_6	Start signal reset for start logic input n for pin PIO0_6. 0 = No effect. 1 = Write: reset start signal.	0
7	RSRPIO0_7	Start signal reset for start logic input n for pin PIO0_7. 0 = No effect. 1 = Write: reset start signal.	0
8	RSRPIO0_8	Start signal reset for start logic input n for pin PIO0_8. 0 = No effect. 1 = Write: reset start signal.	0
9	RSRPIO0_9	Start signal reset for start logic input n for pin PIO0_9. 0 = No effect. 1 = Write: reset start signal.	0
10	RSRPIO0_10	Start signal reset for start logic input n for pin PIO0_10. 0 = No effect. 1 = Write: reset start signal.	0
11	RSRPIO0_11	Start signal reset for start logic input n for pin PIO0_11. 0 = No effect. 1 = Write: reset start signal.	0
31:12	-	Reserved. Do not write a 1 to reserved register bits.	-

4.5.33 Start logic status register 0

This register reflects the status of the enabled start signal bits. The bit assignment is identical to Table 37. Each bit (if enabled) reflects the state of the start logic, i.e. whether or not a wake-up signal has been received for a given pin.

Table 40. Start logic status register 0 (STARTSRP0, address 0x4004 820C) bit description

Bit	Symbol	Description	Reset value
0	SRPIO0_0	Start signal status for start logic input n for pin PIO0_0. 0 = No start signal received. 1 = Start signal pending.	0
1	SRPIO0_1	Start signal status for start logic input n for pin PIO0_1. 0 = No start signal received. 1 = Start signal pending.	0
2	SRPIO0_2	Start signal status for start logic input n for pin PIO0_2. 0 = No start signal received. 1 = Start signal pending.	0
3	SRPIO0_3	Start signal status for start logic input n for pin PIO0_3. 0 = No start signal received. 1 = Start signal pending.	0
4	SRPIO0_4	Start signal status for start logic input n for pin PIO0_4. 0 = No start signal received. 1 = Start signal pending.	0
5	SRPIO0_5	Start signal status for start logic input n for pin PIO0_5. 0 = No start signal received. 1 = Start signal pending.	0
6	SRPIO0_6	Start signal status for start logic input n for pin PIO0_6. 0 = No start signal received. 1 = Start signal pending.	0
7	SRPIO0_7	Start signal status for start logic input n for pin PIO0_7. 0 = No start signal received. 1 = Start signal pending.	0
8	SRPIO0_8	Start signal status for start logic input n for pin PIO0_8. 0 = No start signal received. 1 = Start signal pending.	0
9	SRPIO0_9	Start signal status for start logic input n for pin PIO0_9. 0 = No start signal received. 1 = Start signal pending.	0
10	SRPIO0_10	Start signal status for start logic input n for pin PIO0_10. 0 = No start signal received. 1 = Start signal pending.	0
11	SRPIO0_11	Start signal status for start logic input n for pin PIO0_11. 0 = No start signal received. 1 = Start signal pending.	0
31:12	-	Reserved. Do not write a 1 to reserved register bits.	-

4.5.34 Start logic edge control register 1

The STARTAPRP1 register controls the start logic inputs which are connected to the peripheral interrupts. This register selects a falling or rising edge on the corresponding interrupt to produce a falling or rising clock edge, respectively, for the start logic. Through this start logic register set, peripheral interrupts can be used to wake up the microcontroller from Deep-sleep mode.

Every bit in the STARTAPRP1 register controls one peripheral interrupt (see Table 41) and corresponds to an interrupt in the NVIC, see Table 4.

Table 41. Start logic bit connection to peripheral interrupts

Start logic bit	Interrupt	Description
9:0	-	Reserved
10	WDT	Watchdog interrupt (WDINT)
11	BOD	Brown-out detect
17:12	-	Reserved
18	RTC	RTC interrupt
31:19	-	Reserved

Table 42. Start logic edge control register 1 (STARTAPRP1, address 0x4004 8210) bit description

Bit	Symbol	Description	Reset value
9:0	-	Reserved. Do not write a 1 to reserved register bits.	-
10	APRWDT	Edge select for start logic interrupt WDT. 0 = Falling edge. 1 = Rising edge.	0
11	APRBOD	Edge select for start logic interrupt BOD. 0 = Falling edge. 1 = Rising edge.	0
17:12	-	Reserved. Do not write a 1 to reserved register bits.	0
18	APRRTC	Edge select for start logic interrupt RTC. 0 = Falling edge. 1 = Rising edge.	0
31:19	-	Reserved. Do not write a 1 to reserved register bits.	-

4.5.35 Start logic signal enable register 1

This STARTERP1 register enables or disables the start signal bits in the start logic. The bit assignment is identical to Table 42.

Table 43. Start logic signal enable register 1 (STARTERP1, address 0x4004 8214) bit description

Bit	Symbol	Description	Reset value
9:0	-	Reserved. Do not write a 1 to reserved register bits.	-
10	ERWDT	Enable start signal for start logic interrupt WDT. 0 = Disabled. 1 = Enabled.	0
11	ERBOD	Enable start signal for start logic interrupt BOD. 0 = Disabled. 1 = Enabled.	0
17:12	-	Reserved. Do not write a 1 to reserved register bits.	0
18	ERRTC	Enable start signal for start logic interrupt RTC. 0 = Disabled. 1 = Enabled.	0
31:19	-	Reserved. Do not write a 1 to reserved register bits.	-

4.5.36 Start logic reset register 1

Writing a one to a bit in the STARTRSRP1CLR register resets the start logic state. The bit assignment is identical to Table 42. The start-up logic uses the input signals to generate a clock edge for registering a start signal. This clock edge (falling or rising) sets the interrupt for waking up from Deep-sleep mode. Therefore, the start-up logic states must be cleared before being used.

Table 44. Start logic reset register 1 (STARTRSRP1CLR, address 0x4004 8218) bit description

Bit	Symbol	Description	Reset value
9:0	-	Reserved. Do not write a 1 to reserved register bits.	-
10	RSRWDT	Start signal reset for start logic interrupt WDT. 0 = No effect. 1 = Write: reset start signal.	0
11	RSRBOD	Start signal reset for start logic interrupt BOD. 0 = No effect. 1 = Write: reset start signal.	0
17:12	-	Reserved. Do not write a 1 to reserved register bits.	0
18	RSRRTC	Start signal reset for start logic interrupt RTC. 0 = No effect. 1 = Write: reset start signal.	0
31:19	-	Reserved. Do not write a 1 to reserved register bits.	-

4.5.37 Start logic status register 1

This register reflects the status of the enabled start signals. The bit assignment is identical to Table 42.

Table 45. Start logic signal status register 1 (STARTSRP1, address 0x4004 821C) bit description

Bit	Symbol	Description	Reset value
9:0	-	Reserved. Do not write a 1 to reserved register bits.	-
10	SRWDT	Start signal status for start logic interrupt WDT. 0 = No start signal received. 1 = Start signal pending.	0
11	SRBOD	Start signal status for start logic interrupt BOD. 0 = No start signal received. 1 = Start signal pending.	0
17:12	-	Reserved. Do not write a 1 to reserved register bits.	0
18	SRRTC	Start signal status for start logic interrupt RTC. 0 = No start signal received. 1 = Start signal pending.	0
31:19	-	Reserved. Do not write a 1 to reserved register bits.	-

4.5.38 Deep-sleep mode configuration register

This register controls the behavior of the WatchDog (WD) oscillator and the BOD circuit when the device enters Deep-sleep mode. In addition, the WD oscillator behavior is influenced by the WDLOCKCLK bit in the WDMODE register (Table 266). All other analog blocks are shut down in Deep-sleep mode.

This register must be written before entering Deep-sleep mode and before setting the WDT lock bit WDLOCKCLK with one of the four values shown in Table 46:

Table 46. Allowed values for PDSLEEPCFG register if WDLOCKCLK = 0 (WD oscillator not locked)

Configuration	**WD oscillator on**	**WD oscillator off**
BOD on	PDSLEEPCFG = 0x0000 FFB7	PDSLEEPCFG = 0x0000 FFF7
BOD off	PDSLEEPCFG = 0x0000 FFBF	PDSLEEPCFG = 0x0000 FFFF

Table 47. Allowed values for PDSLEEPCFG register if WDLOCKCLK = 1 (WD oscillator locked)

Configuration	**WD oscillator on**	**WD oscillator off**
BOD on	PDSLEEPCFG = 0x0000 FFB7	not allowed
BOD off	PDSLEEPCFG = 0x0000 FFBF	not allowed

Remark: Failure to initialize and program this register correctly may result in undefined behavior of the microcontroller. The values listed in Table 46 and Table 47 are the only values allowed for PDSLEEPCFG register.

To select the appropriate power configuration for Deep-sleep mode, consider the following:

- BOD: Leaving the BOD circuit enabled will protect the part from a low voltage event occurring while the part is in Deep-sleep mode. However, the BOD circuit causes an additional current drain in Deep-sleep mode.
- WD oscillator: The watchdog oscillator can be left running in Deep-sleep mode to provide a clock for the watchdog timer if needed for timing a wake-up event (see Section 4.8.3 for details). In this case, the watchdog oscillator analog output frequency must be set to its lowest value (bits FREQSEL in the WDTOSCCTRL = 0001, see Table 13) and all peripheral clocks other than the timer clock must be disabled in the SYSAHBCLKCTRL register (see Table 21) before entering Deep-sleep mode.

  Note that the WD oscillator must be running before setting the WDLOCKCLK bit in the WDMODE register.

  The watchdog oscillator, if running, contributes an additional current drain in Deep-sleep mode.

Remark: Reserved bits in this register must always be written as indicated. This register must be initialized correctly before entering Deep-sleep mode.

Remark: Settings in this register are affected by the WDT lock status: If the watchdog oscillator is selected as the clock source for the WDT, writes to bit 6 in the PDSLEEPCFG register are ignored if at the same time bit 5 is set in the WDMOD register (see Table 266).

Table 48. Deep-sleep configuration register (PDSLEEPCFG, address 0x4004 8230) bit description

Bit	Symbol	Value	Description	Reset value
2:0	-		Reserved. Always write these bits as ones.	111
3	BOD_PD		BOD power-down control in Deep-sleep mode	1
		0	Powered	
		1	Powered down	
5:4	-		Reserved. Always write these bits as ones.	11
6	WDTOSC_PD		Watchdog oscillator power-down control in Deep-sleep mode	1
		0	Powered	
		1	Powered down. **Must be changed to 0 before the WDLOCKCLK bit is set in the WDMODE register.** See Table 266.	
15:7	-		Reserved. Always write these bits as ones.	1 1111 1111
31:16	-	-	Reserved	0

4.5.39 Wake-up configuration register

The bits in this register can be programmed to indicate the state the microcontroller must enter when it is waking up from Deep-sleep mode.

Remark: Settings in this register are affected by the WDT lock status:

- If the watchdog oscillator is selected as the clock source for the WDT, writes to bit 6 in the PDAWAKECFG register are ignored if at the same time bit 5 is set in the WDMOD register (see Table 266).
- If the IRC is selected as the clock source for the WDT, writes to bits 0 and bit 1 in the PDAWAKECFG register are ignored if at the same time bit 5 is set in the WDMOD register (see Table 266).

Table 49. 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	

Table 49. Wake-up configuration register (PDAWAKECFG, address 0x4004 8234) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
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		System oscillator wake-up configuration	1
		0	Powered	
		1	Powered down	
6	WDTOSC_PD		Watchdog oscillator wake-up configuration	1
		0	Powered	
		1	Powered down	
7	SYSPLL_PD		System PLL wake-up configuration	1
		0	Powered	
		1	Powered down	
8	-	-	Reserved	-
9	-	-	Reserved.	0
10	-		Reserved	-
11	-	-	Reserved.	1
14:12	-	-	Reserved	-
15	COMP_PD		Comparator power-down wake-up configuration	1
		0	Powered	
		1	Powered down	
31:16	-	-	Reserved	0

4.5.40 Power-down configuration register

The bits in the PDRUNCFG register control the power to the various analog blocks. This register can be written to at any time while the microcontroller 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.

Remark: Settings in this register are affected by the WDT lock status:

- If the watchdog oscillator is selected as the clock source for the WDT, writes to bit 6 in the PDRUNCFG register are ignored if at the same time bit 5 is set in the MOD register (see Table 266).
- If the IRC is selected as the clock source for the WDT, writes to bits 0 and bit 1 in the PDRUNCFG register are ignored if at the same time bit 5 is set in the MOD register (see Table 266).

Table 50. Power-down configuration register (PDRUNCFG, address 0x4004 8238) bit description

Bit	Symbol	Value	Description	Reset value
0	IRCOUT_PD		IRC oscillator output power-down	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	
3	BOD_PD		BOD power-down	0
		0	Powered	
		1	Powered down	
4	ADC_PD		ADC power-down	1
		0	Powered	
		1	Powered down	
5	SYSOSC_PD		System oscillator power-down	1
		0	Powered	
		1	Powered down	
6	WDTOSC_PD		Watchdog oscillator power-down	1
		0	Powered	
		1	Powered down	
7	SYSPLL_PD		System PLL power-down	1
		0	Powered	
		1	Powered down	
8	-	-	Reserved	-
9	-	-	Reserved.	0
10	-	-	Reserved	-
11	-	-	Reserved.	1
14:12	-	-	Reserved	-
15	COMP_PD		Comparator power-down	1
		0	Powered	
		1	Powered down	
31:16	-	-	Reserved	0

4.5.41 Device ID register

This device ID register is a read-only register and contains the device ID for each LPC122x part. This register is also read by the ISP/IAP commands (see Section 20.7.11 and Section 20.8.5).

Table 51. Device ID register (DEVICE_ID, address 0x4004 83F4) bit description

Bit	Symbol	Description	Reset value
31:0	DEVICEID	Device ID for LPC122x parts: LPC12D27FBD100/301 = 0x3670 002B LPC1227FBD64/301 = 0x3670 002B LPC1227FBD48/301 = 0x3670 002B LPC1226FBD64/301 = 0x3660 002B LPC1226FBD48/301 = 0x3660 002B LPC1225FBD64/321 = 0x3652 002B LPC1225FBD64/301 = 0x3650 002B LPC1225FBD48/321 = 0x3652 002B LPC1225FBD48/301 = 0x3650 002B LPC1224FBD64/121 = 0x3642 C02B LPC1224FBD64/101 = 0x3640 C02B LPC1224FBD48/121 = 0x3642 C02B LPC1224FBD48/101 = 0x3640 C02B	part-dependent

4.5.42 Flash configuration register

Access to the flash memory can be configured with various access times by writing to the FLASHCFG register at address 0x5006 0028.

Remark: Settings in this register only have an effect when the FLASH_OVERRIDE bit in the PRESET_CTRL register (see Table 9) is set to 0 allowing multi-cycle read access. For frequencies greater than 30 MHz, the FLASH_OVERRIDE bit must be set to 0, and the user can choose any desired flash access time mentioned in Table 52.

Table 52. Flash configuration register (FLASHCFG, address 0x5006 0028) bit description

Bit	Symbol	Value	Description	Reset value
1:0	RDCFG		Flash memory read access time:	0
		0x0	2 system clock cycles flash access time	
		0x1	3 system clock cycles flash access time	
		0x2	4 system clock cycles flash access time	
		0x3	5 system clock cycles flash access time	
31:2	-		Reserved.	-

4.6 Reset

Reset has four sources on the LPC122x: the $\overline{\text{RESET}}$ pin, Watchdog Reset, Power-On Reset (POR), and Brown Out Detect (BOD). In addition, there is a software reset.

The $\overline{\text{RESET}}$ pin is a Schmitt trigger input pin. Assertion of 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 a reset source external to the Cortex-M0 CPU (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 boot code in the ROM starts. The boot code performs the boot tasks and may jump to the flash.
3. The flash is powered up. This takes approximately <tbd> μs. Then the flash initialization sequence is started, which takes about <tbd> cycles.

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

4.7 Power management

The LPC122x support a variety of power control features. In Active mode, when the microcontroller is running, power and clocks to selected peripherals can be optimized for power consumption. In addition, there are three special modes of processor power reduction: Sleep mode, Deep-sleep mode, and Deep power-down mode.

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

4.7.1 Active mode

In Active mode, the ARM Cortex-M0 core and memories are clocked by the system clock, and peripherals are clocked by the system clock or a dedicated peripheral clock.

The microcontroller 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.

4.7.1.1 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 21).
- The power to various analog blocks (PLL, oscillators, the ADC, the BOD circuit, and the flash block) can be controlled at any time individually through the PDRUNCFG register (Table 50).
- The clock source for the system clock can be selected from the IRC (default), the system oscillator, or the watchdog oscillator (see Figure 3 and related registers).
- The system clock frequency can be selected by the SYSPLLCTRL (Table 10) and the SYSAHBCLKDIV register (Table 21).
- Selected peripherals (UART, SSP0/1, WDT) use individual peripheral clocks with their own clock dividers. The peripheral clocks can be shut down through the corresponding clock divider registers (Table 22 to Table 28).

4.7.2 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 enabled 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 their 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.

4.7.2.1 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.

4.7.2.2 Programming Sleep mode

The following steps must be performed to enter Sleep mode:

1. The DPDEN bit in the PCON register must be set to zero (Table 56).
2. The SLEEPDEEP bit in the ARM Cortex-M0 SCR register must be set to zero, see (Table 388).
3. Use the ARM Cortex-M0 Wait-For-Interrupt (WFI) instruction.

4.7.2.3 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.

4.7.3 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 must be selected or deselected during Deep-sleep mode in the PDSLEEPCFG register. The RTC and the RTC oscillator are operating in Deep-sleep mode unless the RTC is powered down.

Deep-sleep mode eliminates all power used by the flash, analog peripherals and all dynamic power used by the processor itself, memory systems and their 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.

4.7.3.1 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 48) register:

- Except for the RTC oscillator, the only clock source available in Deep-sleep mode is the watchdog oscillator. The watchdog oscillator can be left running in Deep-sleep mode if required for peripheral-controlled wake-up (see Section 4.8.3). All other clock

sources (the IRC and system oscillator) and the system PLL are shut down. The watchdog oscillator analog output frequency must be set to the lowest value of its analog clock output (bits FREQSEL in the WDTOSCCTRL = 0001, see Table 13).

- The BOD circuit can be left running in Deep-sleep mode if required by the application.
- If the watchdog oscillator is running in Deep-sleep mode, only the watchdog timer should be enabled in SYSAHBCLKCTRL register to minimize power consumption.
- The RTC and the RTC oscillator can be left running in Deep-sleep mode.

4.7.3.2 Programming Deep-sleep mode

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

1. The DPDEN bit in the PCON register must be set to zero (Table 56).
2. Select the power configuration in Deep-sleep mode in the PDSLEEPCFG (Table 48) register.
   a. For peripheral controlled wake-up, ensure that the watchdog oscillator is powered in the PDRUNCFG register and switch the clock source to WD oscillator in the MAINCLKSEL register (Table 18).
   b. Without peripheral controlled wake-up and if the watchdog oscillator is shut down, ensure that the IRC is powered in the PDRUNCFG register and switch the clock source to IRC in the MAINCLKSEL register (Table 18). This ensures that the system clock is shut down glitch-free.
3. Select the power configuration after wake-up in the PDAWAKECFG (Table 49) register.
4. Configure the start logic:
   – If an external pin is used for wake-up, enable and clear the wake-up pin in the start logic 0 registers (Table 37 to Table 40), and enable the start logic interrupt in the NVIC.
   – If the RTC is used, enable and clear bit 18 in the start logic 1 registers (Table 42 to Table 45), and enable the RTC interrupt in the NVIC.
5. In the SYSAHBCLKCTRL register (Table 21), disable all peripherals except RTC or WDT if needed.
6. Configure all pins hosting comparator inputs as GPIO outputs and drive the outputs to LOW. Disable the pin's internal pull-up/pull-down resistors. Affects pins PIO0_19, PIO0_20, PIO0_21, PIO0_22, PIO0_23, PIO0_24, PIO0_25, and PIO0_26.
7. Write one to the SLEEPDEEP bit in the ARM Cortex-M0 SCR register (Table 388).
8. Use the ARM WFI instruction.

4.7.3.3 Wake-up from Deep-sleep mode

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

- Signal on an external pin. For this purpose, pins PIO0_0 to PIO0_11 can be enabled as inputs to the start logic. The start logic does not require any clocks and generates the interrupt if enabled in the NVIC to wake up from Deep-sleep mode.
- RTC match interrupt for self-timed wake-up. The RTC interrupt must be enabled in the start logic1 block.

- 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 32).
- Reset from the watchdog timer. 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.
- External $\overline{\text{RESET}}$ pin.

Remark: If the watchdog oscillator is running in Deep-sleep mode, its frequency determines the wake-up time causing the wake-up time to be longer than waking up with the 12 MHz IRC.

4.7.4 Deep power-down mode

In Deep power-down mode, power and clocks are shut off to the entire microcontroller with the exception of the WAKEUP pin. The microcontroller is blocked from entering Deep power-down mode when the WDLOCKDP bit is set to one in the WDMODE register (Table 266).

If the RTC is enabled before entering Deep power-down mode, the RTC and the RTC oscillator continue to run in Deep power-down mode. If the RTC is not needed in Deep power-down mode, disable the RTC to minimize power consumption.

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 four 32-bit general purpose registers of the PMU block.

All functional pins are tri-stated in Deep power-down mode except for the WAKEUP pin.

4.7.4.1 Power configuration in Deep power-down mode

Deep power-down mode has no configuration options. All clocks, the core, and all peripherals except for the RTC and the RTC oscillator are powered down. Only the WAKEUP pin and the backup registers are powered. The low-power RTC and the RTC oscillator can be left running (this is the default).

4.7.4.2 Programming Deep power-down mode

Remark: The microcontroller can only enter Deep-power down mode if the WDLOCKDP bit is set to 0 in the WDMODE register (Table 266). If WDLOCKDP = 1, the microcontroller must be reset before the Deep power-down mode can be entered.

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

1. Write one to the DPDEN bit in the PCON register (see Table 56).
2. Store data to be retained in the general purpose registers (Table 57).
3. Write one to the SLEEPDEEP bit in the ARM Cortex-M0 SCR register (Table 388).
4. Ensure that the IRC is powered by setting bits IRCOUT_PD and IRC_PD to zero in the PDRUNCFG register before entering Deep power-down mode.
5. Use the ARM WFI instruction.

Remark: The WAKEUP pin must be pulled HIGH externally before entering Deep power-down mode.

4.7.4.3 Wake-up from Deep power-down mode

Waking up from Deep power-down mode causes the microcontroller to reset (see Section 4.6). However, the contents of the RTC registers and the backup registers are conserved. The LPC122x can wake up from Deep power-down mode in two ways (see Section 4.9):

- Pulling the WAKEUP pin LOW wakes up the LPC122x from Deep power-down. The minimum pulse width for the HIGH-to-LOW transition on the WAKEUP pin is 50 ns.
- An RTC interrupt (if it is not masked in the RTCIMSC register, see Table 261) wakes up the LPC122x from Deep power-down mode.

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

4.8 Deep-sleep mode details

4.8.1 IRC oscillator

The IRC is the only oscillator on the LPC122x that can always shut down glitch-free. Therefore it is recommended that the user switches the clock source to IRC before the microcontroller enters Deep-sleep mode.

4.8.2 Using external pins to wake up from Deep-sleep mode (start logic 0)

The Deep-sleep mode is exited when the start logic indicates an interrupt to the ARM core. The port pins PIO0_0 to PIO0_11 are connected to the start logic and serve as wake-up pins. The user must program the start logic 0 registers for each input to set the appropriate edge polarity for the corresponding wake-up event. Furthermore, the interrupts corresponding to each input must be enabled in the NVIC. Interrupts 0 to 11 in the NVIC correspond to 12 PIO pins (see Section 4.5.30).

The start logic does not require a clock to run because it uses the input signals on the enabled pins to generate a clock edge. Therefore, the start logic signals should be cleared (see Table 39) before use.

The start logic can also be used in Active mode to provide a vectored interrupt using the LPC122x's input pins.

4.8.3 Using the RTC to wake up from Deep-sleep mode (start logic 1)

The RTC is clocked by the independent RTC oscillator and continues to run in Deep-sleep mode. The RTC interrupt is internally connected to bit 18 in the start logic 1 block and can serve as wake-up interrupt. The user must program the start logic 1 registers for the RTC interrupt to enable the RTC wake-up and select the rising edge configuration. Furthermore, the RTC interrupt must be enabled in the NVIC.

4.9 Deep power-down mode details

4.9.1 Using the WAKEUP pin to wake up from Deep power-down mode

Pulling the WAKEUP pin LOW wakes up the LPC122x from Deep power-down, and the microcontroller goes through the entire reset process (Section 4.6). The minimum pulse

width for the HIGH-to-LOW transition on the WAKEUP pin is 50 ns.

Follow these steps to wake up the microcontroller from Deep power-down mode using the WAKEUP pin:

1. Generate a wake-up signal by going HIGH-to-LOW externally on the WAKEUP pin with a pulse length of at least 50 ns while the part is in Deep power-down mode.
   - 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 microcontroller re-boots.
   - All registers except the GPREG0 to GPREG3 will be in their reset state.
   - The contents of the RTC registers will be preserved if the RTC is enabled.
2. Once the microcontroller has booted, read the deep power-down flag in the PCON register (Table 57) to verify that the reset was caused by a wake-up event from Deep power-down.
3. Clear the deep power-down flag in the PCON register (Table 55).
4. (Optional) Read the stored data in the general purpose registers (Table 57 and Table 58).
5. Set up the PMU for the next Deep power-down cycle.

4.9.2 Using the RTC to wake up from Deep power-down mode

An RTC interrupt wakes up the microcontroller from Deep power-down mode in the same way the WAKEUP pin does. The wake-up signal is generated when an RTC interrupt is created after a programmed number of clocks.

To use the RTC wake-up interrupt, the RTC must be configured as follows:

- Program the RTC match register with the RTC timer match value (Table 258).
- Clear the interrupt mask in the RTC interrupt mask register (Table 261).
- The clock source for the RTC must be the RTC oscillator (Table 58).

Follow these steps to wake up from Deep power-down mode using the RTC:

1. Generate a self-timed RTC wake-up interrupt by setting the RTC match register.
   - 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 microcontroller re-boots.
   - All registers except the GPREG0 to GPREG3 will be in their reset state.
   - The contents of the RTC registers will be preserved.
2. Once the microcontroller has booted, read the deep power-down flag in the PCON register (Table 57) to verify that the reset was caused by a wake-up event from Deep power-down.
3. Clear the deep power-down flag in the PCON register (Table 55).
4. Read the content of the General Purpose registers if needed.
5. Read the RTC count.
6. (Optional) Clear the RTC interrupt in the RTC ICR register and the pending interrupt in the NVIC.

7. (Optional) Read the stored data in the general purpose registers (Table 57 and Table 58).
8. Set up the PMU for the next Deep power-down cycle.

4.10 System PLL functional description

The LPC122x uses the system PLL to create the clocks for the core and peripherals.

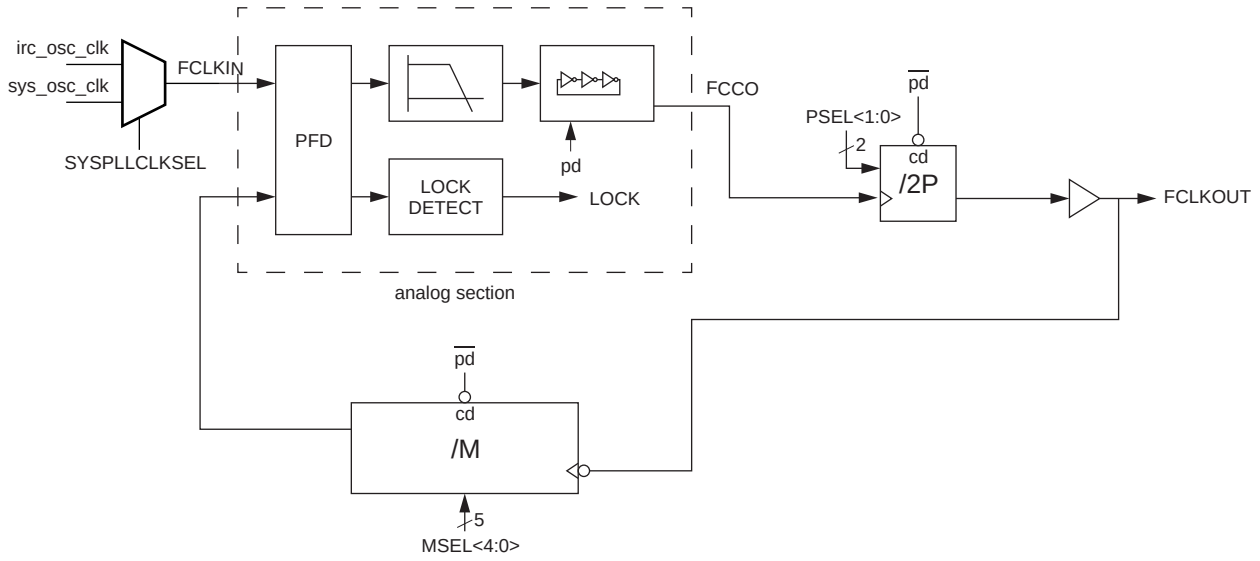


Fig 4. System PLL block diagram

The block diagram of this PLL is shown in Figure 4. 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 2×P by the programmable post divider to create the output clock(s), or are sent directly to the output(s). 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.

4.10.1 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.

4.10.2 Power-down control

To reduce the power consumption when the PLL clock is not needed, a 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 50). 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 Power-down mode, the lock output will be low to indicate that the PLL is not in lock. When the Power-down mode is terminated by setting the SYSPLL_PD bit to zero, the PLL will resume its normal operation and will make the lock signal high once it has regained lock on the input clock.

4.10.3 Divider ratio programming

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 10. This guarantees an output clock with a 50% duty cycle.

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 10.

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.

4.10.4 Frequency selection

The PLL frequency equations use the following parameters (also see Figure 3):

Table 53. PLL frequency parameters

Parameter	System PLL
FCLKIN	Frequency of sys_pllclkin (input clock to the system PLL) from the SYSPLLCLKSEL multiplexer (see Section 4.5.9).
FCCO	Frequency of the Current Controlled Oscillator (CCO); 156 to 320 MHz.
FCLKOUT	Frequency of sys_pllclkout. FCLKOUT must be < 100 MHz.
P	System PLL post divider ratio; PSEL bits in SYSPLLCTRL (see Section 4.5.3).
M	System PLL feedback divider register; MSEL bits in SYSPLLCTRL (see Section 4.5.3).

4.10.4.1 Normal mode

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

(1)

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

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 × P × F_{clkout}.
4. Verify that all frequencies and divider values conform to the limits specified in Table 10.
5. Ensure that FCLKOUT < 100 MHz.

Table 54. PLL configuration examples

PLL input clock sys_pllclkin (Fclkin)	Main clock (Fclkout)	MSEL bits Table 10	M divider value	PSEL bits Table 10	P divider value	FCCO frequency	FLASH_ OVERRIDE bit Table 9	SYSAHB CLKDIV Table 20
12 MHz	24 MHz	00001 (binary)	2	10 (binary)	4	192 MHz	1	1
12 MHz	36 MHz	00010 (binary)	3	10 (binary)	4	288 MHz	0	1

5.1 How to read this chapter

The PMU is identical on all LPC122x parts.

5.2 Introduction

The PMU controls the Deep power-down mode. Four general purpose register in the PMU can be used to retain data during Deep power-down mode. The PMU registers retain their state during Deep-sleep and Deep power-down modes.

5.3 Register description

Table 55. Register overview: PMU (base address 0x4003 8000)

Name	Access	Address offset	Description	Reset value
PCON	R/W	0x000	Power control register	0x0
GPREG0	R/W	0x004	General purpose register 0	0x0
GPREG1	R/W	0x008	General purpose register 1	0x0
GPREG2	R/W	0x00C	General purpose register 2	0x0
GPREG3	R/W	0x010	General purpose register 3	0x0
SYSCFG	R/W	0x014	System configuration register (RTC clock control and hysteresis of the WAKEUP pin).	0x0

5.3.1 Power control register

The power control register selects whether one of the ARM Cortex-M0 controlled power-down modes (Sleep mode or Deep-sleep mode) or the Deep power-down mode is entered and provides the flags for Sleep or Deep-sleep modes and Deep power-down modes respectively. See Section 4.7 for details on how to enter the power-down modes.

Table 56. Power control register (PCON, address 0x4003 8000) bit description

Bit	Symbol	Value	Description	Reset value
0	-	-	Reserved. Do not write 1 to this bit.	0x0
1	DPDEN		Deep power-down mode enable	0
		0	ARM WFI will enter Sleep or Deep-sleep mode (clock to ARM Cortex-M0 core turned off).	
		1	ARM WFI will enter Deep-power down mode (ARM Cortex-M0 core powered-down) if WDLOCKDP = 0 (see Table 266 “Watchdog Mode register (MOD - 0x4000 4000) bit description”).	
7:2	-	-	Reserved. Do not write ones to this bit.	0x0

Table 56. Power control register (PCON, address 0x4003 8000) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
8	SLEEPFLAG		Sleep mode flag	0
		0	Read: No power-down mode entered. LPC122x is in Run mode. Write: No effect.	
		1	Read: Sleep/Deep-sleep or Deep power-down mode entered. Write: Writing a 1 clears the SLEEPFLAG bit to 0.	
10:9	-	-	Reserved. Do not write ones to this bit.	0x0
11	DPDFLAG		Deep power-down flag	0x0
		0	Read: Deep power-down mode **not** entered. Write: No effect.	0x0
		1	Read: Deep power-down mode entered. Write: Clear the Deep power-down flag.	0x0
31:12	-	-	Reserved. Do not write ones to this bit.	0x0

5.3.2 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 $V_{DD(3V3)}$ 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 57. General purpose registers 0 to 3 (GPREG0 - GPREG3, address 0x4003 8004 to 0x4003 8010) bit description

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

5.3.3 System configuration register

This register controls the clock input to the RTC and the hysteresis of the WAKEUP pin.

Three clocks can be selected from the 32 kHz RTC oscillator: the 1 Hz clock (default), the delayed 1 Hz clock, and the 1 kHz clock. In addition, the peripheral RTC clock, which is derived from the main clock by the RTC clock divider, can be selected as RTC clock source.

Remark: The RTC clock source must be selected before the RTC is enabled in the SYSAHBCLKCTRL register (see Table 21). The clock source must not be changed while the RTC is running.

Remark: If the external voltage applied on pin $V_{DD(3V3)}$ drops below <tbd> V, the hysteresis of the WAKEUP input pin has to be disabled in order for the chip to wake up from Deep power-down mode.

Table 58. System configuration register (SYSCFG, address 0x4003 8014) bit description

Bit	Symbol	Value	Description	Reset value
9:0	-	-	Reserved. Do not write ones to this bit.	0x0
10	WAKEUPHYS		WAKEUP pin hysteresis enable	0x0
		0	Hysteresis for WAKUP pin disabled.	
		1	Hysteresis for WAKEUP pin enabled.	
14:11	RTCCLK		RTC clock source select (X = don't care). 0000 = 1 Hz clock 0001 = delayed 1 Hz clock 101X = 1 kHz clock 01XX = RTC PCLK (main clock divided by the RTC clock divider value)	0000
31:15	-		Reserved. Do not write ones to this bit.	0x0

6.1 How to read this chapter

Each pin has one IOCON register assigned. IOCON registers for pins not available are reserved (see Table 59). IOCON registers controlling pins of GPIO port 2 are only available on 64-pin packages.

Table 59. Available GPIO pins/IOCON registers

Package	GPIO0	GPIO1	GPIO2	Total GPIO pins/ IOCON registers
LQFP48	PIO0_0 to PIO0_31	PIO1_0 to PIO1_6	-	39
LQFP64	PIO0_0 to PIO0_31	PIO1_0 to PIO1_6	PIO2_0 to PIO2_15	55
LQFP100[1]	PIO0_0 to PIO0_31	PIO1_0 to PIO1_6	PIO2_0 to PIO2_15	55

[1] Part LPC12D27.

6.2 Features

The I/O configuration registers control the electrical characteristics of the pads. The following features are programmable:

- Pin function
- Pin mode: Internal pull-up resistor enable/disable
- Pin drive
- Analog input or digital mode for pads hosting the ADC inputs
- I^2C-bus mode or GPIO mode for pads hosting the I^2C-bus interface
- True, programmable open-drain mode for I^2C-bus pins

6.3 General description

6.3.1 Pin function

The FUNC bits in the IOCON registers can be set to GPIO (FUNC = 000) or to a peripheral function. If the pins are configured as GPIO pins, the DIR registers determine whether the pin is configured as an input or output (see Table 129). For any peripheral function, the pin direction is controlled automatically depending on the pin's functionality. The GPIOnDIR registers have no effect for peripheral functions.

6.3.2 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.

6.3.3 Pin drive

Two levels of output drive can be selected for each normal-drive pin, named low mode and high mode. Four pins (PIO0_27, PIO0_28, PIO0_29, PIO0_12) are designated high-drive pins with a high mode and low mode output drive. For details see the LPC122x data sheet.

6.3.4 Open-drain mode

An open-drain mode can be enabled for all digital I/O pins. Unless for pins PIO0_10 and PIO0_11, this mode is not a true open-drain mode. The input cannot be pulled up above $V_{DD(IO)}$.

6.3.5 A/D-mode

In A/D-mode, the digital receiver is disconnected to obtain an accurate input voltage for analog-to-digital conversions. This mode is available in those IOCON registers that control pins which can function as ADC inputs. If A/D mode is selected, the pin mode setting has no effect.

6.3.6 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.

6.3.7 Programmable glitch filter

All PIO 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). The filter clock can be selected from one of seven peripheral clocks PCLK0 to 6, which are derived from the main clock (see Figure 3) using the IOCONFIGCLKDIV0 to 6 (see Table 31) 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, for example the timer capture inputs or the SSP function, 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.

6.4 Register description

Table 60 shows the IOCONFIG registers. Each multiplexed pin is associated with one register which allows to program its function and electrical characteristics. The register name is derived from the pin's default function after reset. Note that some pins reset to functions other than GPIO. The corresponding registers are indicated by a prefix reflecting the pin's function after reset: Either a serial wire debug function (SWDIO or SWCLK) or a reserved function (R).

Table 60. Register overview: I/O configuration block (base address 0x4004 4000)

Name	Access	Address offset	Description	Reset value	Reference
-	R/W	0x000	Reserved.	-	-
-	R/W	0x004	Reserved.	-	-
PIO0_19	R/W	0x008	Configures pin PIO0_19/ACMP0_I0/CT32B0_1.	0x0000 0090	Table 63
PIO0_20	R/W	0x00C	Configures pin PIO0_20/ACMP0_I1/CT32B0_2.	0x0000 0090	Table 64
PIO0_21	R/W	0x010	Configures pin PIO0_21/ACMP0_I2/CT32B0_3.	0x0000 0090	Table 65
PIO0_22	R/W	0x014	Configures pin PIO0_22/ACMP0_I3.	0x0000 0090	Table 66
PIO0_23	R/W	0x018	Configures pin PIO0_23/ACMP1_I0/CT32B1_0.	0x0000 0090	Table 67
PIO0_24	R/W	0x01C	Configures pin PIO0_24/ACMP1_I1/CT32B1_1.	0x0000 0090	Table 68
SWDIO_PIO0_25	R/W	0x020	Configures pin SWDIO/ACMP1_I2/ CT32B1_2/PIO0_25.	0x0000 0090	Table 69
SWCLK_PIO0_26	R/W	0x024	Configures pin SWCLK/PIO0_26/ACMP1_I3/ CT32B1_3/PIO0_26	0x0000 0090	Table 70
PIO0_27	R/W	0x028	Configures pin PIO0_27/ACMP0_O.	0x0000 0090	Table 71
PIO2_12	R/W	0x02C	Configures pin PIO2_12/RXD1.	0x0000 0090	Table 72
PIO2_13	R/W	0x030	Configures pin PIO2_13/TXD1.	0x0000 0090	Table 73
PIO2_14	R/W	0x034	Configures pin PIO2_14.	0x0000 0090	Table 74
PIO2_15	R/W	0x038	Configures pin PIO2_15.	0x0000 0090	Table 75
PIO0_28	R/W	0x03C	Configures pin PIO0_28/ACMP1_O/CT16B0_0.	0x0000 0090	Table 76
PIO0_29	R/W	0x040	Configures pin PIO0_29/ROSC/CT16B0_1.	0x0000 0090	Table 77
PIO0_0	R/W	0x044	Configures pin PIO0_0/$\overline{\text{RTS0}}$.	0x0000 0090	Table 78
PIO0_1	R/W	0x048	Configures pin PIO0_1CT32B0_0/RXD0.	0x0000 0090	Table 79
PIO0_2	R/W	0x04C	Configures pin PIO0_2/TXD0/CT32B0_1.	0x0000 0090	Table 80
-	R/W	0x050	Reserved	-	-
PIO0_3	R/W	0x054	Configures pin PIO0_3/$\overline{\text{DTR0}}$/CT32B0_2.	0x0000 0090	Table 81
PIO0_4	R/W	0x058	Configures pin PIO0_4/$\overline{\text{DSR0}}$/CT32B0_3.	0x0000 0090	Table 82
PIO0_5	R/W	0x05C	Configures pin PIO0_5.	0x0000 0090	Table 83

Table 60. Register overview: I/O configuration block (base address 0x4004 4000) *…continued*

Name	Access	Address offset	Description	Reset value	Reference
PIO0_6	R/W	0x060	Configures pin PIO0_6/$\overline{\text{RI0}}$/CT32B1_0.	0x0000 0090	Table 84
PIO0_7	R/W	0x064	Configures pin PIO0_7$\overline{\text{CTS0}}$/CT32B1_1.	0x0000 0090	Table 85
PIO0_8	R/W	0x068	Configures pin PIO0_8/RXD1/CT32B1_2.	0x0000 0090	Table 86
PIO0_9	R/W	0x06C	Configures pin PIO0_9/TXD1/CT32B1_3.	0x0000 0090	Table 87
PIO2_0	R/W	0x070	Configures pin PIO2_0/CT16B0_0/$\overline{\text{RTS0}}$.	0x0000 0090	Table 88
PIO2_1	R/W	0x074	Configures pin PIO2_1/CT16B0_1/RXD0.	0x0000 0090	Table 89
PIO2_2	R/W	0x078	Configures pin PIO2_2/CT16B1_0/TXD0.	0x0000 0090	Table 90
PIO2_3	R/W	0x07C	Configures pin PIO2_3/CT16B1_1/$\overline{\text{DTR0}}$.	0x0000 0090	Table 91
PIO2_4	R/W	0x080	Configures pin PIO2_4/CT32B0_0/$\overline{\text{CTS0}}$.	0x0000 0090	Table 92
PIO2_5	R/W	0x084	Configures pin PIO2_5/CT32B0_1/$\overline{\text{DCD0}}$.	0x0000 0090	Table 93
PIO2_6	R/W	0x088	Configures pin PIO2_6/CT32B0_2/$\overline{\text{RI0}}$.	0x0000 0090	Table 94
PIO2_7	R/W	0x08C	Configures pin PIO2_7/CT32B0_3/$\overline{\text{DSR0}}$.	0x0000 0090	Table 95
PIO0_10	R/W	0x090	Configures pin PIO0_10/SCL.	0x0000 0080	Table 96
PIO0_11	R/W	0x094	Configures pin PIO0_11/SDA/CT16B0_0.	0x0000 0080	Table 97
PIO0_12	R/W	0x098	Configures pin PIO0_12/CLKOUT/CT16B0_1.	0x0000 0090	Table 98
RESET_PIO0_13	R/W	0x09C	Configures pin $\overline{\text{RESET}}$/PIO0_13.	0x0000 0090	Table 99
PIO0_14	R/W	0x0A0	Configures pin PIO0_14/SSP_CLK.	0x0000 0090	Table 100
PIO0_15	R/W	0x0A4	Configures pin PIO0_15/SSP_SSEL/CT16B1_0.	0x0000 0090	Table 101
PIO0_16	R/W	0x0A8	Configures pin PIO0_16/SSP_MISO/CT16B1_1.	0x0000 0090	Table 102
PIO0_17	R/W	0x0AC	Configures pin PIO0_17/SSP_MOSI.	0x0000 0090	Table 103
PIO0_18	R/W	0x0B0	Configures pin PIO0_18/SWCLK/CT32B0_0.	0x0000 0090	Table 104
R_PIO0_30	R/W	0x0B4	Configures pin R/PIO0_30/AD0.	0x0000 0090	Table 105
R_PIO0_31	R/W	0x0B8	Configures pin R/PIO0_31/AD1.	0x0000 0090	Table 106
R_PIO1_0	R/W	0x0BC	Configures pin R/PIO1_0/AD2.	0x0000 0090	Table 107
R_PIO1_1	R/W	0x0C0	Configures pin R/PIO1_1/AD3.	0x0000 0090	Table 108
PIO1_2	R/W	0x0C4	Configures pin PIO1_2/SWDIO/AD4.	0x0000 0090	Table 109
PIO1_3	R/W	0x0C8	Configures pin PIO1_3/AD5/WAKEUP.	0x0000 0090	Table 110
PIO1_4	R/W	0x0CC	Configures pin PIO1_4/AD6	0x0000 0090	Table 111
PIO1_5	R/W	0x0D0	Configures pin PIO1_5/AD7/CT16B1_0.	0x0000 0090	Table 112
PIO1_6	R/W	0x0D4	Configures pin PIO1_6/CT16B1_1.	0x0000 0090	Table 113
-	-	0x0D8	Reserved.	-	-
-	-	0x0DC	Reserved.	-	-
PIO2_8	R/W	0x0E0	Configures pin PIO2_8/CT32B1_0.	0x0000 0090	Table 114
PIO2_9	R/W	0x0E4	Configures pin PIO2_9/CT32B1_1.	0x0000 0090	Table 115
PIO2_10	R/W	0x0E8	Configures pin PIO2_10/CT32B1_2/TXD1.	0x0000 0090	Table 116
PIO2_11	R/W	0x0EC	Configures pin PIO2_11/CT32B1_3/RXD1.	0x0000 0090	Table 117

6.4.1 Pin configuration registers

Table 61 shows the register bit allocation for all IOCON registers (except PIO0_10 and PIO0_11). Table 62 shows the bit allocation for pins PIO0_10 and PIO0_11 which allow to configure a true open-drain mode to comply with the full I^2C-bus specification.

Table 61. IOCON register bit allocation (except I^2C-pins)

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		000	Selects function 0 (default).	
		001	Select function 1.	
		010	Select function 2.	
		011	Select function 3.	
		100	Select function 4.	
		101	Select function 5.	
		110	Select function 6.	
		111	Reserved.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	ADMODE		Analog/Digital mode	1
		0	Analog mode enabled.	
		1	Digital mode enabled.	
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Disable.	
		1	Open-drain mode enabled. **Remark:** This is not a true open-drain mode. Input cannot be pulled up above $V_{DD(IO)}$.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Sampling for 1 filter clock cycle. Input pulses shorter than one filter clock are rejected.	
		0x2	Sampling for 2 filter clock cycles. Input pulses shorter than two filter clocks are rejected.	
		0x3	Sampling for 3 filter clock cycles. Input pulses shorter than three filter clocks are rejected.	

Table 61. IOCON register bit allocation (except I^2C-pins)

Bit	Symbol	Value	Description	Reset value
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock. Value 0x7 is reserved. (see Table 31).	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

Table 62. IOCON register bit allocation (I^2C-bus pins PIO0_10 and PIO0_11)

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		000	Selects function 0 (default).	
		001	Select function 1.	
		010	Select function 2.	
		011	Select function 3.	
		100	Select function 4.	
		101	Select function 5.	
		110	Select function 6.	
		111	Reserved.	
5:3	-		Reserved.	000
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
9:7	-	-	Reserved.	001
10	TOD		True open-drain mode.	0
		0	Disable.	
		1	True open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		00	Bypass input filter.	
		01	Sampling for 1 filter clock cycle. Input pulses shorter than one filter clock are rejected.	
		10	Sampling for 2 filter clock cycles. Input pulses shorter than two filter clocks are rejected.	
		11	Sampling for 3 filter clock cycles. Input pulses shorter than three filter clocks are rejected.	

Table 62. IOCON register bit allocation (I^2C-bus pins PIO0_10 and PIO0_11)

Bit	Symbol	Value	Description	Reset value
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock (see Table 31).	000
		000	IOCONFIGCLKDIV0.	
		001	IOCONFIGCLKDIV1.	
		010	IOCONFIGCLKDIV2.	
		011	IOCONFIGCLKDIV3.	
		100	IOCONFIGCLKDIV4.	
		101	IOCONFIGCLKDIV5.	
		110	IOCONFIGCLKDIV6.	
		111	Reserved.	
31:16	-	-	Reserved.	0

6.4.2 PIO0_19 register

Table 63. PIO0_19 register (PIO0_19, address 0x4004 4008) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_19.	
		0x1	Reserved. Do not use.	
		0x2	Select function ACMP0_I0.	
		0x3	Select function CT32B0_CAP1.	
		0x4	Select function CT32B0_MAT1.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	ADMODE		Analog/Digital mode	1
		0	Analog mode enabled.	
		1	Digital mode enabled.	
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	High mode current selected.	
		1	Low mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	

Table 63. PIO0_19 register (PIO0_19, address 0x4004 4008) bit description

Bit	Symbol	Value	Description	Reset value
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.3 PIO0_20 register

Table 64. PIO0_20 register (PIO0_20, address 0x4004 400C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_20.	
		0x1	Reserved. Do not use.	
		0x2	Select function ACMP0_I1.	
		0x3	Select function CT32B0_CAP2.	
		0x4	Select function CT32B0_MAT2.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	ADMODE		Analog/Digital mode	1
		0	Analog mode enabled.	
		1	Digital mode enabled.	
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	High mode current selected.	
		1	Low mode current selected.	

Table 64. PIO0_20 register (PIO0_20, address 0x4004 400C) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.4 PIO0_21 register

Table 65. PIO0_21 register (PIO0_21, address 0x4004 4010) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_21.	
		0x1	Reserved. Do not use.	
		0x2	Select function ACMP0_I2.	
		0x3	Select function CT32B0_CAP3.	
		0x4	Select function CT32B0_MAT3.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	ADMODE		Analog/Digital mode	1
		0	Analog mode enabled.	
		1	Digital mode enabled.	
8	-		Reserved.	0

Table 65. PIO0_21 register (PIO0_21, address 0x4004 4010) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
9	DRV		Drive current mode (Normal-drive pin).	0
		0	High mode current selected.	
		1	Low mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.5 PIO0_22 register

Table 66. PIO0_22 register (PIO0_22, address 0x4004 4014) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_22.	
		0x1	Reserved. Do not use.	
		0x2	Select function ACMP0_I3.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	

Table 66. PIO0_22 register (PIO0_22, address 0x4004 4014) bit description

…continued

Bit	Symbol	Value	Description	Reset value
7	ADMODE		Analog/Digital mode	1
		0	Analog mode enabled.	
		1	Digital mode enabled.	
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	High mode current selected.	
		1	Low mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.6 PIO0_23 register

Table 67. PIO0_23 register (PIO0_23, address 0x4004 4018) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_23.	
		0x1	Reserved. Do not use.	
		0x2	Select function ACMP1_I0.	
		0x3	Select function CT32B1_CAP0.	
		0x4	Select function CT32B1_MAT0.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	

Table 67. PIO0_23 register (PIO0_23, address 0x4004 4018) bit description

Bit	Symbol	Value	Description	Reset value
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	ADMODE		Analog/Digital mode	1
		0	Analog mode enabled.	
		1	Digital mode enabled.	
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	High mode current selected.	
		1	Low mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.7 PIO0_24 register

Table 68. PIO0_24 register (PIO0_24, address 0x4004 401C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_24.	
		0x1	Reserved. Do not use.	
		0x2	Select function ACMP1_I1.	
		0x3	Select function CT32B1_CAP1.	
		0x4	Select function CT32B1_MAT1.	
3	-		Reserved	0

Table 68. PIO0_24 register (PIO0_24, address 0x4004 401C) bit description

Bit	Symbol	Value	Description	Reset value
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	ADMODE		Analog/Digital mode	1
		0	Analog mode enabled.	
		1	Digital mode enabled.	
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	High mode current selected.	
		1	Low mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.8 SWDIO_PIO0_25 register

Table 69. SWDIO_PIO0_25 register (SWDIO_PIO0_25, address 0x4004 4020) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function SWDIO.	
		0x1	Reserved. Do not use.	
		0x2	Select function ACMP1_I2.	
		0x3	Select function CT32B1_CAP2.	
		0x4	Select function CT32B1_MAT2.	
		0x5	Reserved.	
		0x6	Select function PIO0_25.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	ADMODE		Analog/Digital mode	1
		0	Analog mode enabled.	
		1	Digital mode enabled.	
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	High mode current selected.	
		1	Low mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	

Table 69. SWDIO_PIO0_25 register (SWDIO_PIO0_25, address 0x4004 4020) bit description
…continued

Bit	Symbol	Value	Description	Reset value
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.9 SWCLK_PIO0_26 register

Table 70. SWCLK_PIO0_26 register (SWCLK_PIO0_26, address 0x4004 4024) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function SWCLK.	
		0x1	Reserved. Do not use.	
		0x2	Select function ACMP1_I3.	
		0x3	Select function CT32B1_CAP3.	
		0x4	Select function CT32B1_MAT3.	
		0x5	Reserved.	
		0x6	Select function PIO0_26.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	ADMODE		Analog/Digital mode	1
		0	Analog mode enabled.	
		1	Digital mode enabled.	
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	High mode current selected.	
		1	Low mode current selected.	

Table 70. SWCLK_PIO0_26 register (SWCLK_PIO0_26, address 0x4004 4024) bit description

Bit	Symbol	Value	Description	Reset value
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.10 PIO0_27 register

Table 71. PIO0_27 register (PIO0_27, address 0x4004 4028) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_27.	
		0x1	Reserved. Do not use.	
		0x2	Select function ACMP0_O.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0

Table 71. PIO0_27 register (PIO0_27, address 0x4004 4028) bit description

…continued

Bit	Symbol	Value	Description	Reset value
9	DRV		Drive current mode (High-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.11 PIO2_12 register

Table 72. PIO2_12 register (PIO2_12, address 0x4004 402C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO2_12.	
		0x1	Reserved. Do not use.	
		0x2	Reserved.	
		0x3	Selects function RXD1.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved	1

Table 72. PIO2_12 register (PIO2_12, address 0x4004 402C) bit description

Bit	Symbol	Value	Description	Reset value
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.12 PIO2_13 register

Table 73. PIO2_13 register (PIO2_13, address 0x4004 4030) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO2_13.	
		0x1	Reserved. Do not use.	
		0x2	Reserved.	
		0x3	Select function TXD1.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	

Table 73. PIO2_13 register (PIO2_13, address 0x4004 4030) bit description
…continued

Bit	Symbol	Value	Description	Reset value
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.13 PIO2_14 register

Table 74. PIO2_14 register (PIO2_14, address 0x4004 4034) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO2_14.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0

Table 74. PIO2_14 register (PIO2_14, address 0x4004 4034) bit description

Bit	Symbol	Value	Description	Reset value
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.14 PIO2_15 register

Table 75. PIO2_15 register (PIO2_15, address 0x4004 4038) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO2_15.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0

Table 75. PIO2_15 register (PIO2_15, address 0x4004 4038) bit description

…continued

Bit	Symbol	Value	Description	Reset value
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.15 PIO0_28 register

Table 76. PIO0_28 register (PIO0_28, address 0x4004 403C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_28.	
		0x1	Reserved. Do not use.	
		0x2	Select function ACMP1_O.	
		0x3	Select function CT16B0_CAP0.	
		0x4	Select function CT16B0_MAT0.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	

Table 76. PIO0_28 register (PIO0_28, address 0x4004 403C) bit description

Bit	Symbol	Value	Description	Reset value
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (High-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.16 PIO0_29 register

Table 77. PIO0_29 register (PIO0_29, address 0x4004 4040) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_29.	
		0x1	Reserved. Do not use.	
		0x2	Select function ROSC.	
		0x3	Select function CT16B0_CAP1.	
		0x4	Select function CT16B0_MAT1.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0

Table 77. PIO0_29 register (PIO0_29, address 0x4004 4040) bit description

Bit	Symbol	Value	Description	Reset value
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (High-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.17 PIO0_0 register

Table 78. PIO0_0 register (PIO0_0, address 0x4004 4044) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_0.	
		0x1	Reserved. Do not use.	
		0x2	Select function $\overline{\text{RTS0}}$.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0

Table 78. PIO0_0 register (PIO0_0, address 0x4004 4044) bit description

Bit	Symbol	Value	Description	Reset value
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.18 PIO0_1 register

Table 79. PIO0_1 register (PIO0_1, address 0x4004 4048) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_1.	
		0x1	Reserved. Do not use.	
		0x2	Select function RXD0.	
		0x3	Select function CT32B0_CAP0.	
		0x4	Select function CT32B0_MAT0.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	

Table 79. PIO0_1 register (PIO0_1, address 0x4004 4048) bit description

Bit	Symbol	Value	Description	Reset value
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.19 PIO0_2 register

Table 80. PIO0_2 register (PIO0_2, address 0x4004 404C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_2.	
		0x1	Reserved. Do not use.	
		0x2	Select function TXD0.	
		0x3	Select function CT32B0_CAP1.	
		0x4	Select function CT32B0_MAT1.	
3	-		Reserved	0

Table 80. PIO0_2 register (PIO0_2, address 0x4004 404C) bit description

…continued

Bit	Symbol	Value	Description	Reset value
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.20 PIO0_3 register

Table 81. PIO0_3 register (PIO0_3, address 0x4004 4054) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_3.	
		0x1	Reserved. Do not use.	
		0x2	Select function $\overline{\text{DTR0}}$.	
		0x3	Select function CT32B0_CAP2.	
		0x4	Select function CT32B0_MAT2.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.21 PIO0_4 register

Table 82. PIO0_4 register (PIO0_4, address 0x4004 4058) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_4.	
		0x1	Reserved. Do not use.	
		0x2	Select function $\overline{\text{DSR0}}$.	
		0x3	Select function CT32B0_CAP3.	
		0x4	Select function CT32B0_MAT3.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.22 PIO0_5 register

Table 83. PIO0_5 register (PIO0_5, address 0x4004 405C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_5.	
		0x1	Reserved. Do not use.	
		0x2	Select function $\overline{\text{DCD0}}$.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.23 PIO0_6 register

Table 84. PIO0_6 register (PIO0_6, address 0x4004 4060) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_6.	
		0x1	Reserved. Do not use.	
		0x2	Select function $\overline{\text{RI0}}$.	
		0x3	Select function CT32B1_CAP0.	
		0x4	Select function CT32B1_MAT0.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.24 PIO0_7 register

Table 85. PIO0_7 register (PIO0_7, address 0x4004 4064) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_7.	
		0x1	Reserved. Do not use.	
		0x2	Select function $\overline{\text{CTS0}}$.	
		0x3	Select function CT32B1_CAP1.	
		0x4	Select function CT32B1_MAT1.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.25 PIO0_8 register

Table 86. PIO0_8 register (PIO0_8, address 0x4004 4068) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_8.	
		0x1	Reserved. Do not use.	
		0x2	Select function RXD1.	
		0x3	Select function CT32B1_CAP2.	
		0x4	Select function CT32B1_MAT2.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.26 PIO0_9 register

Table 87. PIO0_9 register (PIO0_9, address 0x4004 406C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_9.	
		0x1	Reserved. Do not use.	
		0x2	Select function TXD1.	
		0x3	Select function CT32B1_CAP3.	
		0x4	Select function CT32B1_MAT3.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.27 PIO2_0 register

Table 88. PIO2_0 register (PIO2_0, address 0x4004 4070) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO2_0.	
		0x1	Reserved. Do not use.	
		0x2	Select function CT16B0_CAP0.	
		0x3	Select function CT16B0_MAT0.	
		0x4	Select function $\overline{\text{RTS0}}$.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.28 PIO2_1 register

Table 89. PIO2_1 register (PIO2_1, address 0x4004 4074) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO2_1.	
		0x1	Reserved. Do not use.	
		0x2	Select function CT16B0_CAP1.	
		0x3	Select function CT16B0_MAT1.	
		0x4	Select function RXD0.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.29 PIO2_2 register

Table 90. PIO2_2 register (PIO2_2, address 0x4004 4078) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO2_2.	
		0x1	Reserved. Do not use.	
		0x2	Select function CT16B1_CAP0.	
		0x3	Select function CT16B1_MAT0.	
		0x4	Select function TXD0.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.30 PIO2_3 register

Table 91. PIO2_3 register (PIO2_3, address 0x4004 407C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO2_3.	
		0x1	Reserved. Do not use.	
		0x2	Select function CT16B1_CAP1.	
		0x3	Select function CT16B1_MAT1.	
		0x4	Select function $\overline{\text{DTR0}}$.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.31 PIO2_4 register

Table 92. PIO2_4 register (PIO2_4, address 0x4004 4080) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO2_4.	
		0x1	Reserved. Do not use.	
		0x2	Select function CT32B0_CAP0.	
		0x3	Select function CT32B0_MAT0.	
		0x4	Select function $\overline{\text{CTS0}}$.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.32 PIO2_5 register

Table 93. PIO2_5 register (PIO2_5, address 0x4004 4084) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO2_5.	
		0x1	Reserved. Do not use.	
		0x2	Select function CT32B0_CAP1.	
		0x3	Select function CT32B0_MAT1.	
		0x4	Select function $\overline{RI0}$.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.33 PIO2_6 register

Table 94. PIO2_6 register (PIO2_6, address 0x4004 4088) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO2_6.	
		0x1	Reserved. Do not use.	
		0x2	Select function CT32B0_CAP2.	
		0x3	Select function CT32B0_MAT2.	
		0x4	Select function $\overline{\text{DCD0}}$.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.34 PIO2_7 register

Table 95. PIO2_7 register (PIO2_7, address 0x4004 408C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO2_7.	
		0x1	Reserved. Do not use.	
		0x2	Select function CT32B0_CAP3.	
		0x3	Select function CT32B0_MAT3.	
		0x4	Select function $\overline{\text{DSR0}}$.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.35 PIO0_10 register

Table 96. PIO0_10 register (PIO0_10, address 0x4004 4090) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_10.	
		0x1	Reserved. Do not use.	
		0x2	Select I2C-bus function SCL.	
5:3	-		Reserved.	000
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
9:7	-	-	Reserved.	001
10	TOD		True open-drain mode.	0
		0	Disable.	
		1	True open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
		0x0	Reserved.	
31:16	-	-	Reserved.	0

6.4.36 PIO0_11 register

Table 97. PIO0_11 register (PIO0_11, address 0x4004 4094) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_11.	
		0x1	Reserved. Do not use.	
		0x2	Select I2C-bus function SDA.	
		0x3	Select function CT16B0_CAP0.	
		0x4	Select function CT16B0_MAT0.	

Table 97. PIO0_11 register (PIO0_11, address 0x4004 4094) bit description

Bit	Symbol	Value	Description	Reset value
5:3	-		Reserved.	000
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
9:7	-	-	Reserved.	001
10	TOD		True open-drain mode.	0
		0	Disable.	
		1	True open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
		0x0	Reserved.	
31:16	-	-	Reserved.	0

6.4.37 PIO0_12 register

Table 98. PIO0_12 register (PIO0_12, address 0x4004 4098) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_12.	
		0x1	Reserved. Do not use.	
		0x2	Select function CLKOUT.	
		0x3	Select function CT16B0_CAP1.	
		0x4	Select function CT16B0_MAT1.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0

Table 98. PIO0_12 register (PIO0_12, address 0x4004 4098) bit description

Bit	Symbol	Value	Description	Reset value
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (High-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.38 RESET_PIO0_13 register

Table 99. RESET_PIO0_13 register (RESET_PIO0_13, address 0x4004 409C) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function $\overline{\text{RESET}}$.	
		0x1	Select function PIO0_13.	
		0x2	Reserved. Do not use.	
		0x3	Reserved.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	

Table 99. RESET_PIO0_13 register (RESET_PIO0_13, address 0x4004 409C) bit description

Bit	Symbol	Value	Description	Reset value
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.39 PIO0_14 register

Table 100. PIO0_14 register (PIO0_14, address 0x4004 40A0) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_14.	
		0x1	Reserved. Do not use.	
		0x2	Select function SCK.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	

Table 100. PIO0_14 register (PIO0_14, address 0x4004 40A0) bit description

Bit	Symbol	Value	Description	Reset value
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.40 PIO0_15 register

Table 101. PIO0_15 register (PIO0_15, address 0x4004 40A4) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_15.	
		0x1	Reserved. Do not use.	
		0x2	Select function SSEL.	
		0x3	Select function CT16B1_CAP0.	
		0x4	Select function CT16B1_MAT0.	
3	-		Reserved	0

Table 101. PIO0_15 register (PIO0_15, address 0x4004 40A4) bit description

Bit	Symbol	Value	Description	Reset value
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.41 PIO0_16 register

Table 102. PIO0_16 register (PIO0_16, address 0x4004 40A8) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_16.	
		0x1	Reserved. Do not use.	
		0x2	Select function MISO.	
		0x3	Select function CT16B1_CAP1.	
		0x4	Select function CT16B1_MAT1.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.42 PIO0_17 register

Table 103. PIO0_17 register (PIO0_17, address 0x4004 40AC) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_17.	
		0x1	Reserved. Do not use.	
		0x2	Select function MOSI.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.43 PIO0_18 register

Table 104. PIO0_18 register (PIO0_18, address 0x4004 40B0) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO0_18.	
		0x1	Select function SWCLK.	
		0x2	Reserved. Do not use.	
		0x3	Select function CT32B0_CAP0.	
		0x4	Select function CT32B0_MAT0.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.44 R_PIO0_30 register

Table 105. R_PIO0_30 register (R_PIO0_30, address 0x4004 40B4) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Reserved.	
		0x1	Select function PIO0_30.	
		0x2	Reserved. Do not use.	
		0x3	Select function AD0.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	ADMODE		Analog/Digital mode	1
		0	Analog mode enabled.	
		1	Digital mode enabled.	
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.45 R_PIO0_31 register

Table 106. PIO0_31 register (R_PIO0_31, address 0x4004 40B8) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Reserved.	
		0x1	Select function PIO0_31.	
		0x2	Reserved. Do not use.	
		0x3	Select function AD1.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	ADMODE		Analog/Digital mode	1
		0	Analog mode enabled.	
		1	Digital mode enabled.	
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.46 R_PIO1_0 register

Table 107. R_PIO1_0 register (R_PIO1_0, address 0x4004 40BC) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Reserved.	
		0x1	Select function PIO1_0.	
		0x2	Select function AD2.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	ADMODE		Analog/Digital mode	1
		0	Analog mode enabled.	
		1	Digital mode enabled.	
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.47 R_PIO1_1 register

Table 108. R_PIO1_1 register (R_PIO1_1, address 0x4004 40C0) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Reserved.	
		0x1	Select function PIO1_0.	
		0x2	Select function AD3.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	ADMODE		Analog/Digital mode	1
		0	Analog mode enabled.	
		1	Digital mode enabled.	
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.48 PIO1_2 register

Table 109. PIO1_2 register (PIO1_2, address 0x4004 40C4) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO1_2.	
		0x1	Select function SWDIO.	
		0x2	Select function AD4.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	ADMODE		Analog/Digital mode	1
		0	Analog mode enabled.	
		1	Digital mode enabled.	
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.49 PIO1_3 register

Table 110. PIO1_3 register (PIO1_3, address 0x4004 40C8) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function. This pin functions as WAKEUP pin if the part is in Deep power-down mode regardless of the value of FUNC.	000
		0x0	Selects function PIO1_3.	
		0x1	Select function AD5.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	ADMODE		Analog/Digital mode	1
		0	Analog mode enabled.	
		1	Digital mode enabled.	
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.50 PIO1_4 register

Table 111. PIO1_4 register (PIO1_4, address 0x4004 40CC) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO1_4.	
		0x1	Select function AD6.	
		0x2	Reserved.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	ADMODE		Analog/Digital mode	1
		0	Analog mode enabled.	
		1	Digital mode enabled.	
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.51 PIO1_5 register

Table 112. PIO1_5 register (PIO1_5, address 0x4004 40D0) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO1_5.	
		0x1	Select function AD7.	
		0x2	Select function CT16B1_CAP0.	
		0x3	Select function CT16B1_MAT0.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	ADMODE		Analog/Digital mode	1
		0	Analog mode enabled.	
		1	Digital mode enabled.	
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode drive current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.52 PIO1_6 register

Table 113. PIO1_6 register (PIO1_6, address 0x4004 40D4) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO1_6.	
		0x1	Select function CT16B1_CAP1.	
		0x2	Select function CT16B1_MAT1.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode drive current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.53 PIO2_8 register

Table 114. PIO2_8 register (PIO2_8, address 0x4004 40E0) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO2_8.	
		0x1	Reserved. Do not use.	
		0x2	Select function CT32B1_CAP0.	
		0x3	Select function CT32B1_MAT0.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.54 PIO2_9 register

Table 115. PIO2_9 register (PIO2_9, address 0x4004 40E4) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO2_9.	
		0x1	Reserved. Do not use.	
		0x2	Select function CT32B1_CAP1.	
		0x3	Select function CT32B1_MAT1.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.55 PIO2_10 register

Table 116. PIO2_10 register (PIO2_10, address 0x4004 40E8) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO2_10.	
		0x1	Reserved. Do not use.	
		0x2	Select function CT32B1_CAP2.	
		0x3	Select function CT32B1_MAT2.	
		0x4	Reserved.	
		0x5	Select function TXD1.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

6.4.56 PIO2_11 register

Table 117. PIO2_11 register (PIO2_11, address 0x4004 40EC) bit description

Bit	Symbol	Value	Description	Reset value
2:0	FUNC		Selects pin function.	000
		0x0	Selects function PIO2_11.	
		0x1	Reserved. Do not use.	
		0x2	Select function CT32B1_CAP3.	
		0x3	Select function CT32B1_MAT3.	
		0x4	Reserved.	
		0x5	Select function RXD1.	
3	-		Reserved	0
4	MODE		Selects function mode (on-chip pull-up resistor control).	1
		0	Inactive (pull-up resistor not enabled).	
		1	Pull-up resistor enabled.	
5	-		Reserved.	0
6	INV		Invert input	0
		0	Input not inverted.	
		1	Input inverted.	
7	-		Reserved.	1
8	-		Reserved.	0
9	DRV		Drive current mode (Normal-drive pin).	0
		0	Low mode current selected.	
		1	High mode current selected.	
10	OD		Open-drain mode.	0
		0	Open-drain mode disabled.	
		1	Open-drain mode enabled.	
12:11	S_MODE		Sample mode	00
		0x0	Bypass input filter.	
		0x1	Input pulses shorter than one filter clock are rejected.	
		0x2	Input pulses shorter than two filter clocks are rejected.	
		0x3	Input pulses shorter than three filter clocks are rejected.	
15:13	CLK_DIV		Select peripheral clock divider for input filter sampling clock.	000
		0x0	IOCONFIGCLKDIV0.	
		0x1	IOCONFIGCLKDIV1.	
		0x2	IOCONFIGCLKDIV2.	
		0x3	IOCONFIGCLKDIV3.	
		0x4	IOCONFIGCLKDIV4.	
		0x5	IOCONFIGCLKDIV5.	
		0x6	IOCONFIGCLKDIV6.	
31:16	-	-	Reserved.	0

7.1 General description

All pins except the supply pins can have more than one function as shown in Table 118. The pin function is selected through the pin's IOCON register in the IOCONFIG block. The multiplexed functions include the counter/timer inputs and outputs, the UART receive, transmit, and control functions, and the serial wire debug functions (see Table 120).

For each pin, the default function is listed first together with the pin's reset state.

7.2 Pin description

Table 118. LPC122x pin description

Symbol	Pin LQFP48	Pin LQFP64		Start logic input	Type	Reset state [1]	Description
PIO0_0 to PIO0_31					I/O		**Port 0 —** Port 0 is a 32-bit I/O port with individual direction and function controls for each bit. The operation of port 0 pins depends on the function selected through the IOCONFIG register block.
PIO0_0/$\overline{\text{RTS0}}$	15	19	[2]	yes	I/O	I; PU	**PIO0_0 —** General purpose digital input/output pin.
					O	-	**$\overline{\text{RTS0}}$ —** Request To Send output for UART0.
PIO0_1/RXD0/ CT32B0_CAP0/ CT32B0_MAT0	16	20	[2]	yes	I/O	I; PU	**PIO0_1 —** General purpose digital input/output pin.
					I	-	**RXD0 —** Receiver input for UART0. Used in UART ISP mode.
					I	-	**CT32B0_CAP0 —** Capture input, channel 0 for 32-bit timer 0.
					O	-	**CT32B0_MAT0 —** Match output, channel 0 for 32-bit timer 0.
PIO0_2/TXD0/ CT32B0_CAP1/ CT32B0_MAT1	17	21	[2]	yes	I/O	I; PU	**PIO0_2 —** General purpose digital input/output pin.
					O	-	**TXD0 —** Transmitter output for UART0. Used in UART ISP mode.
					I	-	**CT32B0_CAP1 —** Capture input, channel 1 for 32-bit timer 0.
					O	-	**CT32B0_MAT1 —** Match output, channel 1 for 32-bit timer 0.
PIO0_3/$\overline{\text{DTR0}}$/ CT32B0_CAP2/ CT32B0_MAT2	18	22	[2]	yes	I/O	I; PU	**PIO0_3 —** General purpose digital input/output pin.
					O	-	**$\overline{\text{DTR0}}$ —** Data Terminal Ready output for UART0.
					I	-	**CT32B0_CAP2 —** Capture input, channel 2 for 32-bit timer 0.
					O	-	**CT32B0_MAT2 —** Match output, channel 2 for 32-bit timer 0.
PIO0_4/$\overline{\text{DSR0}}$/ CT32B0_CAP3/ CT32B0_MAT3	19	23	[2]	yes	I/O	I; PU	**PIO0_4 —** General purpose digital input/output pin.
					I	-	**$\overline{\text{DSR0}}$ —** Data Set Ready input for UART0.
					I	-	**CT32B0_CAP3 —** Capture input, channel 3 for 32-bit timer 0.
					O	-	**CT32B0_MAT3 —** Match output, channel 3 for 32-bit timer 0.
PIO0_5/$\overline{\text{DCD0}}$	20	24	[2]	yes	I/O	I; PU	**PIO0_5 —** General purpose digital input/output pin.
					I	-	**$\overline{\text{DCD0}}$ —** Data Carrier Detect input for UART0.

Table 118. LPC122x pin description *…continued*

Symbol	Pin LQFP48	Pin LQFP64		Start logic input	Type	Reset state [1]	Description
PIO0_6/$\overline{\text{RI0}}$/ CT32B1_CAP0/ CT32B1_MAT0	21	25	[2]	yes	I/O	I; PU	**PIO0_6 —** General purpose digital input/output pin.
					I	-	**$\overline{\text{RI0}}$ —** Ring Indicator input for UART0.
					I	-	**CT32B1_CAP0 —** Capture input, channel 0 for 32-bit timer 1.
					O	-	**CT32B1_MAT0 —** Match output, channel 0 for 32-bit timer 1.
PIO0_7/$\overline{\text{CTS0}}$/ CT32B1_CAP1/ CT32B1_MAT1	22	26	[2]	yes	I/O	I; PU	**PIO0_7 —** General purpose digital input/output pin.
					I	-	**$\overline{\text{CTS0}}$ —** Clear To Send input for UART0.
					I	-	**CT32B1_CAP1 —** Capture input, channel 1 for 32-bit timer 1.
					O	-	**CT32B1_MAT1 —** Match output, channel 1 for 32-bit timer 1.
PIO0_8/RXD1/ CT32B1_CAP2/ CT32B1_MAT2	23	27	[2]	yes	I/O	I; PU	**PIO0_8 —** General purpose digital input/output pin.
					I	-	**RXD1 —** Receiver input for UART1.
					I	-	**CT32B1_CAP2 —** Capture input, channel 2 for 32-bit timer 1.
					O	-	**CT32B1_MAT2 —** Match output, channel 2 for 32-bit timer 1.
PIO0_9/TXD1/ CT32B1_CAP3/ CT32B1_MAT3	24	28	[2]	yes	I/O	I; PU	**PIO0_9 —** General purpose digital input/output pin.
					O	-	**TXD1 —** Transmitter output for UART1.
					I	-	**CT32B1_CAP3 —** Capture input, channel 3 for 32-bit timer 1.
					O	-	**CT32B1_MAT3 —** Match output, channel 3 for 32-bit timer 1.
PIO0_10/SCL	25	37	[3]	yes	I/O	I; IA	**PIO0_10 —** General purpose digital input/output pin.
					I/O	-	**SCL —** I^2C-bus clock input/output.
PIO0_11/SDA/ CT16B0_CAP0/ CT16B0_MAT0	26	38	[3]	yes	I/O	I; IA	**PIO0_11 —** General purpose digital input/output pin.
					I/O	-	**SDA —** I^2C-bus data input/output.
					I	-	**CT16B0_CAP0 —** Capture input, channel 0 for 16-bit timer 0.
					O	-	**CT16B0_MAT0 —** Match output, channel 0 for 16-bit timer 0.
PIO0_12/CLKOUT/ CT16B0_CAP1/ CT16B0_MAT1	27	39	[7]	no	I/O	I; PU	**PIO0_12 —** General purpose digital input/output pin. A LOW level on this pin during reset starts the ISP command handler. High-current output driver.
					O	-	**CLKOUT —** Clock out pin.
					I	-	**CT16B0_CAP1 —** Capture input, channel 1 for 16-bit timer 0.
					O	-	**CT16B0_MAT1 —** Match output, channel 1 for 16-bit timer 0.
$\overline{\text{RESET}}$/PIO0_13	28	40	[4]	no	I	I; PU	**$\overline{\text{RESET}}$ —** External reset input: A LOW on this pin resets the device, causing I/O ports and peripherals to take on their default states, and processor execution to begin at address 0.
					I/O	-	**PIO0_13 —** General purpose digital input/output pin.
PIO0_14/SCK	29	41	[2]	no	I/O	I; PU	**PIO0_14 —** General purpose digital input/output pin.
					I/O	-	**SCK —** Serial clock for SSP/SPI.
PIO0_15/SSEL/ CT16B1_CAP0/ CT16B1_MAT0	30	42	[2]	no	I/O	I; PU	**PIO0_15 —** General purpose digital input/output pin.
					I/O	-	**SSEL —** Slave select for SSP/SPI.
					I	-	**CT16B1_CAP0 —** Capture input, channel 0 for 16-bit timer 1.
					O	-	**CT16B1_MAT0 —** Match output, channel 0 for 16-bit timer 1.

Table 118. LPC122x pin description *…continued*

Symbol	Pin LQFP48	Pin LQFP64		Start logic input	Type	Reset state [1]	Description
PIO0_16/MISO/ CT16B1_CAP1/ CT16B1_MAT1	31	43	[2]	no	I/O	I; PU	**PIO0_16 —** General purpose digital input/output pin.
					I/O	-	**MISO —** Master In Slave Out for SSP/SPI.
					I	-	**CT16B1_CAP1 —** Capture input, channel 1 for 16-bit timer 1.
					O	-	**CT16B1_MAT1 —** Match output, channel 1 for 16-bit timer 1.
PIO0_17/MOSI	32	44	[2]	no	I/O	I; PU	**PIO0_17 —** General purpose digital input/output pin.
					I/O	-	**MOSI —** Master Out Slave In for SSP/SPI.
PIO0_18/SWCLK/ CT32B0_CAP0/ CT32B0_MAT0	33	45	[2]	no	I/O	I; PU	**PIO0_18 —** General purpose digital input/output pin.
					I	-	**SWCLK —** Serial wire clock, alternate location.
					I	-	**CT32B0_CAP0 —** Capture input, channel 0 for 32-bit timer 0.
					O	-	**CT32B0_MAT0 —** Match output, channel 0 for 32-bit timer 0.
PIO0_19/ACMP0_I0/ CT32B0_CAP1/ CT32B0_MAT1	4	4	[5]	no	I/O	I; PU	**PIO0_19 —** General purpose digital input/output pin.
					I	-	**ACMP0_I0 —** Input 0 for comparator 0.
					I	-	**CT32B0_CAP1 —** Capture input, channel 1 for 32-bit timer 0.
					O	-	**CT32B0_MAT1 —** Match output, channel 1 for 32-bit timer 0
PIO0_20/ACMP0_I1/ CT32B0_CAP2/ CT32B0_MAT2	5	5	[5]	no	I/O	I; PU	**PIO0_20 —** General purpose digital input/output pin.
					I	-	**ACMP0_I1 —** Input 1 for comparator 0.
					I	-	**CT32B0_CAP2 —** Capture input, channel 2 for 32-bit timer 0.
					O	-	**CT32B0_MAT2 —** Match output, channel 2 for 32-bit timer 0.
PIO0_21/ACMP0_I2/ CT32B0_CAP3/ CT32B0_MAT3	6	6	[5]	no	I/O	I; PU	**PIO0_21 —** General purpose digital input/output pin.
					I	-	**ACMP0_I2 —** Input 2 for comparator 0.
					I	-	**CT32B0_CAP3 —** Capture input, channel 3 for 32-bit timer 0.
					O	-	**CT32B0_MAT3 —** Match output, channel 3 for 32-bit timer 0.
PIO0_22/ACMP0_I3	7	7	[5]	no	I/O	I; PU	**PIO0_22 —** General purpose digital input/output pin.
					I	-	**ACMP0_I3 —** Input 3 for comparator 0.
PIO0_23/ ACMP1_I0/ CT32B1_CAP0/ CT32B1_MAT0	8	8	[5]	no	I/O	I; PU	**PIO0_23 —** General purpose digital input/output pin.
					I	-	**ACMP1_I0 —** Input 0 for comparator 1.
					I	-	**CT32B1_CAP0 —** Capture input, channel 0 for 32-bit timer 1.
					O	-	**CT32B1_MAT0 —** Match output, channel 0 for 32-bit timer 1.
PIO0_24/ACMP1_I1/ CT32B1_CAP1/ CT32B1_MAT1	9	9	[5]	no	I/O	I; PU	**PIO0_24 —** General purpose digital input/output pin.
					I	-	**ACMP1_I1 —** Input 1 for comparator 1.
					I	-	**CT32B1_CAP1 —** Capture input, channel 1 for 32-bit timer 1.
					O	-	**CT32B1_MAT1 —** Match output, channel 1 for 32-bit timer 1.
SWDIO/ACMP1_I2/ CT32B1_CAP2/ CT32B1_MAT2/ PIO0_25	10	10	[5]	no	I/O	I; PU	**SWDIO —** Serial wire debug input/output, default location.
					I	-	**ACMP1_I2 —** Input 2 for comparator 1.
					I	-	**CT32B1_CAP2 —** Capture input, channel 2 for 32-bit timer 1.
					O	-	**CT32B1_MAT2 —** Match output, channel 2 for 32-bit timer 1.
					I/O	-	**PIO0_25 —** General purpose digital input/output pin.

Table 118. LPC122x pin description *…continued*

Symbol	Pin LQFP48	Pin LQFP64		Start logic input	Type	Reset state [1]	Description
SWCLK/ACMP1_I3/ CT32B1_CAP3/ CT32B1_MAT3/ PIO0_26	11	11	[5]	no	I	I; PU	**SWCLK — Serial wire clock, default location.**
					I	-	**ACMP1_I3 —** Input 3 for comparator 1.
					I	-	**CT32B1_CAP3 —** Capture input, channel 3 or 32-bit timer 1.
					O	-	**CT32B1_MAT3 —** Match output, channel 3 for 32-bit timer 1.
					I/O		**PIO0_26 —** General purpose digital input/output pin.
PIO0_27/ACMP0_O	12	12	[7]	no	I/O	I; PU	**PIO0_27 —** General purpose digital input/output pin (high-current output driver).
					O	-	**ACMP0_O —** Output for comparator 0.
PIO0_28/ACMP1_O/ CT16B0_CAP0/ CT16B0_MAT0	13	17	[7]	no	I/O	I; PU	**PIO0_28 —** General purpose digital input/output pin (high-current output driver).
					O	-	**ACMP1_O —** Output for comparator 1.
					I	-	**CT16B0_CAP0 —** Capture input, channel 0 for 16-bit timer 0.
					O	-	**CT16B0_MAT0 —** Match output, channel 0 for 16-bit timer 0.
PIO0_29/ROSC/ CT16B0_CAP1/ CT16B0_MAT1	14	18	[7]	no	I/O	I; PU	**PIO0_29 —** General purpose digital input/output pin (high-current output driver).
					I/O	-	**ROSC —** Relaxation oscillator for 555 timer applications.
					I	-	**CT16B0_CAP1 —** Capture input, channel 1 for 16-bit timer 0.
					O	-	**CT16B0_MAT1 —** Match output, channel 1 for 16-bit timer 0.
R/PIO0_30/AD0	34	46	[5]	no	I	I; PU	**R —** Reserved. Configure for an alternate function in the IOCONFIG block.
					I/O	-	**PIO0_30 —** General purpose digital input/output pin.
					I	-	**AD0 —** A/D converter, input 0.
R/PIO0_31/AD1	35	47	[5]	no	I	I; PU	**R —** Reserved. Configure for an alternate function in the IOCONFIG block.
					I/O	-	**PIO0_31 —** General purpose digital input/output pin.
					I	-	**AD1 —** A/D converter, input 1.
PIO1_0 to PIO1_6					I/O		**Port 1 —** Port 1 is a 32-bit I/O port with individual direction and function controls for each bit. The operation of port 1 pins depends on the function selected through the IOCONFIG register block. Pins PIO1_7 through PIO1_31 are not available.
R/PIO1_0/AD2	36	48	[5]	no	O	I; PU	**R —** Reserved. Configure for an alternate function in the IOCONFIG block.
					I/O	-	**PIO1_0 —** General purpose digital input/output pin.
					I	-	**AD2 —** A/D converter, input 2.
R/PIO1_1/AD3	37	49	[5]	no	I	I; PU	**R —** Reserved. Configure for an alternate function in the IOCONFIG block.
					I/O	-	**PIO1_1 —** General purpose digital input/output pin.
					I	-	**AD3 —** A/D converter, input 3.
PIO1_2/SWDIO/AD4	38	50	[5]	no	I/O	I; PU	**PIO1_2 —** General purpose digital input/output pin.
					I/O	-	**SWDIO — Serial wire debug input/output, alternate location.**
					I	-	**AD4 —** A/D converter, input 4.

Table 118. LPC122x pin description *…continued*

Symbol	Pin LQFP48	Pin LQFP64		Start logic input	Type	Reset state [1]	Description
PIO1_3/AD5/WAKEUP	39	51	[6]	no	I/O	I; PU	**PIO1_3 —** General purpose digital input/output pin.
					I	-	**AD5 —** A/D converter, input 5.
					I	-	**WAKEUP —** Deep power-down mode wake-up pin.
PIO1_4/AD6	40	52	[5]	no	I/O	I; PU	**PIO1_4 —** General purpose digital input/output pin.
					I	-	**AD6 —** A/D converter, input 6.
PIO1_5/AD7/ CT16B1_CAP0/ CT16B1_MAT0	41	53	[5]	no	I/O	I; PU	**PIO1_5 —** General purpose digital input/output pin.
					I	-	**AD7 —** A/D converter, input 7.
					I	-	**CT16B1_CAP0 —** Capture input, channel 0 for 16-bit timer 1.
					O	-	**CT16B1_MAT0 —** Match output, channel 0 for 16-bit timer 1.
PIO1_6/ CT16B1_CAP1/ CT16B1_MAT1	42	54	[2]	no	I/O	I; PU	**PIO1_6 —** General purpose digital input/output pin.
					I	-	**CT16B1_CAP1 —** Capture input, channel 1 for 16-bit timer 1.
					O	-	**CT16B1_MAT1 —** Match output, channel 1 for 16-bit timer 1.
PIO2_0 to PIO2_15					I/O		**Port 2 —** Port 2 is a 32-bit I/O port with individual direction and function controls for each bit. The operation of port 2 pins depends on the function selected through the IOCONFIG register block. Pins PIO2_16 through PIO2_31 are not available.
PIO2_0/ CT16B0_CAP0/ CT16B0_MAT0/ $\overline{\text{RTS0}}$	-	29	[2]	no	I/O	I; PU	**PIO2_0 —** General purpose digital input/output pin.
					I	-	**CT16B0_CAP0 —** Capture input, channel 0 for 16-bit timer 0.
					O	-	**CT16B0_MAT0 —** Match output, channel 0 for 16-bit timer 0.
					O	-	**$\overline{\text{RTS0}}$ —** Request To Send output for UART0.
PIO2_1/ CT16B0_CAP1/ CT16B0_MAT1/RXD0	-	30	[2]	no	I/O	I; PU	**PIO2_1 —** General purpose digital input/output pin.
					I	-	**CT16B0_CAP1 —** Capture input, channel 1 for 16-bit timer 0.
					O	-	**CT16B0_MAT1 —** Match output, channel 1 for 16-bit timer 0.
					I	-	**RXD0 —** Receiver input for UART0.
PIO2_2/ CT16B1_CAP0/ CT16B1_MAT0/TXD0	-	31	[2]	no	I/O	I; PU	**PIO2_2 —** General purpose digital input/output pin.
					I	-	**CT16B1_CAP0 —** Capture input, channel 0 for 16-bit timer 1.
					O	-	**CT16B1_MAT0 —** Match output, channel 0 for 16-bit timer 1.
					O	-	**TXD0 —** Transmitter output for UART0.
PIO2_3/ CT16B1_CAP1/ CT16B1_MAT1/$\overline{\text{DTR0}}$	-	32	[2]	no	I/O	I; PU	**PIO2_3 —** General purpose digital input/output pin.
					I	-	**CT16B1_CAP1 —** Capture input, channel 1 for 16-bit timer 1.
					O	-	**CT16B1_MAT1 —** Match output, channel 1 for 16-bit timer 1.
					O	-	**$\overline{\text{DTR0}}$ —** Data Terminal Ready output for UART0.
PIO2_4/ CT32B0_CAP0/ CT32B0_MAT0/$\overline{\text{CTS0}}$	-	33	[2]	no	I/O	I; PU	**PIO2_4 —** General purpose digital input/output pin.
					I	-	**CT32B0_CAP0 —** Capture input, channel 0 for 32-bit timer 0.
					O	-	**CT32B0_MAT0 —** Match output, channel 0 for 32-bit timer 0.
					I	-	**$\overline{\text{CTS0}}$ —** Clear To Send input for UART0.

Table 118. LPC122x pin description *…continued*

Symbol	Pin LQFP48	Pin LQFP64		Start logic input	Type	Reset state [1]	Description
PIO2_5/ CT32B0_CAP1/ CT32B0_MAT1/RI0	-	34	[2]	no	I/O	I; PU	**PIO2_5 —** General purpose digital input/output pin.
					I	-	**CT32B0_CAP1 —** Capture input, channel 1 for 32-bit timer 0.
					O	-	**CT32B0_MAT1 —** Match output, channel 1 for 32-bit timer 0.
					I	-	**RI0 —** Ring Indicator input for UART0.
PIO2_6/ CT32B0_CAP2/ CT32B0_MAT2/DCD0	-	35	[2]	no	I/O	I; PU	**PIO2_6 —** General purpose digital input/output pin.
					I	-	**CT32B0_CAP2 —** Capture input, channel 2 for 32-bit timer 0.
					O	-	**CT32B0_MAT2 —** Match output, channel 2 for 32-bit timer 0.
					I	-	**DCD0 —** Data Carrier Detect input for UART0.
PIO2_7/ CT32B0_CAP3/ CT32B0_MAT3/DSR0	-	36	[2]	no	I/O	I; PU	**PIO2_7 —** General purpose digital input/output pin.
					I	-	**CT32B0_CAP3 —** Capture input, channel 3 for 32-bit timer 0.
					O	-	**CT32B0_MAT3 —** Match output, channel 3 for 32-bit timer 0.
					I	-	**DSR0 —** Data Set Ready input for UART0.
PIO2_8/ CT32B1_CAP0/ CT32B1_MAT0	-	59	[2]	no	I/O	I; PU	**PIO2_8 —** General purpose digital input/output pin.
					I	-	**CT32B1_CAP0 —** Capture input, channel 0 for 32-bit timer 1.
					O	-	**CT32B1_MAT0 —** Match output, channel 0 for 32-bit timer 1.
PIO2_9/ CT32B1_CAP1/ CT32B1_MAT1	-	60	[2]	no	I/O	I; PU	**PIO2_9 —** General purpose digital input/output pin.
					I	-	**CT32B1_CAP1 —** Capture input, channel 1 for 32-bit timer 1.
					O	-	**CT32B1_MAT1 —** Match output, channel 1 for 32-bit timer 1.
PIO2_10/ CT32B1_CAP2/ CT32B1_MAT2/TXD1	-	61	[2]	no	I/O	I; PU	**PIO2_10 —** General purpose digital input/output pin.
					I	-	**CT32B1_CAP2 —** Capture input, channel 2 for 32-bit timer 1.
					O	-	**CT32B1_MAT2 —** Match output, channel 2 for 32-bit timer 1.
					O	-	**TXD1 —** Transmitter output for UART1.
PIO2_11/ CT32B1_CAP3/ CT32B1_MAT3/RXD1	-	62	[2]	no	I/O	I; PU	**PIO2_11 —** General purpose digital input/output pin.
					I	-	**CT32B1_CAP3 —** Capture input, channel 3 for 32-bit timer 1.
					O	-	**CT32B1_MAT3 —** Match output, channel 3 for 32-bit timer 1.
					I	-	**RXD1 —** Receiver input for UART1.
PIO2_12/RXD1	-	13	[2]	no	I/O	I; PU	**PIO2_12 —** General purpose digital input/output pin.
					I	-	**RXD1 —** Receiver input for UART1.
PIO2_13/TXD1	-	14	[2]	no	I/O	I; PU	**PIO2_13 —** General purpose digital input/output pin.
					O	-	**TXD1 —** Transmitter output for UART1.
PIO2_14	-	15	[2]	no	I/O	I; PU	**PIO2_14 —** General purpose digital input/output pin.
PIO2_15	-	16	[2]	no	I/O	I; PU	**PIO2_15 —** General purpose digital input/output pin.
RTCXIN	46	58		-	I	-	Input to the 32 kHz oscillator circuit.
RTCXOUT	45	57		-	O	-	Output from the 32 kHz oscillator amplifier.
XTALIN	1	1		-	I	-	Input to the system oscillator circuit and internal clock generator circuits.
XTALOUT	2	2		-	O	-	Output from the system oscillator amplifier.
VREF_CMP	3	3		-	I	-	Reference voltage for comparator.

Table 118. LPC122x pin description *…continued*

Symbol	Pin LQFP48	Pin LQFP64		Start logic input	Type	Reset state [1]	Description
$V_{DD(IO)}$	47	63		-	I	-	Input/output supply voltage.
$V_{DD(3V3)}$	44	56		-	I	-	3.3 V supply voltage to the internal regulator and the ADC. Also used as the ADC reference voltage.
V_{SSIO}	48	64		-	I	-	Ground.
V_{SS}	43	55		-	I	-	Ground.

[1] Pin state at reset for default function: I = Input; O = Output; PU = internal pull-up enabled; IA = inactive, no pull-up/down enabled.

[2] 3.3 V tolerant, digital I/O pin; default: pull-up enabled, no hysteresis.

[3] I^2C-bus pins; 5 V tolerant; open-drain; default: no pull-up/pull-down; no hysteresis.

[4] 3.3 V tolerant, digital I/O pin with RESET function; default: pull-up enabled, no hysteresis. An external pull-up resistor is required on this pin for the Deep power-down mode.

[5] 3.3 V tolerant, digital I/O pin with analog function; default: pull-up enabled, no hysteresis.

[6] 3.3 V tolerant, digital I/O pin with analog function and WAKEUP function; default: pull-up enabled, no hysteresis.

[7] 3.3 V tolerant, high-drive digital I/O pin; default: pull-up enabled, no hysteresis.

Table 119. LPC12D27 LQFP100 pin description

Symbol	Pin	Start logic input	Reset state [1]	Type	Description
Microcontroller pins					
PIO0_0 to PIO0_31				I/O	**Port 0 —** Port 0 is a 32-bit I/O port with individual direction and function controls for each bit. The operation of port 0 pins depends on the function selected through the IOCONFIG register block.
PIO0_0/$\overline{\text{RTS0}}$	6[2]	yes	I; PU	I/O	**PIO0_0 —** General purpose digital input/output pin. Also serves as wake-up pin from Deep-sleep mode.
				O	**$\overline{\text{RTS0}}$ —** Request To Send output for UART0.
PIO0_1/RXD0/ CT32B0_CAP0/ CT32B0_MAT0	7[2]	yes	I; PU	I/O	**PIO0_1 —** General purpose digital input/output pin. Also serves as wake-up pin from Deep-sleep mode.
				I	**RXD0 —** Receiver input for UART0. Used in UART ISP mode.
				I	**CT32B0_CAP0 —** Capture input, channel 0 for 32-bit timer 0.
				O	**CT32B0_MAT0 —** Match output, channel 0 for 32-bit timer 0.
PIO0_2/TXD0/ CT32B0_CAP1/ CT32B0_MAT1	8[2]	yes	I; PU	I/O	**PIO0_2 —** General purpose digital input/output pin. Also serves as wake-up pin from Deep-sleep mode.
				O	**TXD0 —** Transmitter output for UART0. Used in UART ISP mode.
				I	**CT32B0_CAP1 —** Capture input, channel 1 for 32-bit timer 0.
				O	**CT32B0_MAT1 —** Match output, channel 1 for 32-bit timer 0.
PIO0_3/$\overline{\text{DTR0}}$/ CT32B0_CAP2/ CT32B0_MAT2	9[2]	yes	I; PU	I/O	**PIO0_3 —** General purpose digital input/output pin. Also serves as wake-up pin from Deep-sleep mode.
				O	**$\overline{\text{DTR0}}$ —** Data Terminal Ready output for UART0.
				I	**CT32B0_CAP2 —** Capture input, channel 2 for 32-bit timer 0.
				O	**CT32B0_MAT2 —** Match output, channel 2 for 32-bit timer 0.

Table 119. LPC12D27 LQFP100 pin description *…continued*

Symbol	Pin	Start logic input	Reset state [1]	Type	Description
PIO0_4/ CT32B0_CAP3/ CT32B0_MAT3	10[2]	yes	I; PU	I/O	**PIO0_4 —** General purpose digital input/output pin. Also serves as wake-up pin from Deep-sleep mode.
				I	**$\overline{\text{DSR0}}$ —** Data Set Ready input for UART0.
				I	**CT32B0_CAP3 —** Capture input, channel 3 for 32-bit timer 0.
				O	**CT32B0_MAT3 —** Match output, channel 3 for 32-bit timer 0.
PIO0_5/$\overline{\text{DCD0}}$	11[2]	yes	I; PU	I/O	**PIO0_5 —** General purpose digital input/output pin. Also serves as wake-up pin from Deep-sleep mode.
				I	**$\overline{\text{DCD0}}$ —** Data Carrier Detect input for UART0.
PIO0_6/$\overline{\text{RI0}}$/ CT32B1_CAP0/ CT32B1_MAT0	12[2]	yes	I; PU	I/O	**PIO0_6 —** General purpose digital input/output pin. Also serves as wake-up pin from Deep-sleep mode.
				I	**$\overline{\text{RI0}}$ —** Ring Indicator input for UART0.
				I	**CT32B1_CAP0 —** Capture input, channel 0 for 32-bit timer 1.
				O	**CT32B1_MAT0 —** Match output, channel 0 for 32-bit timer 1.
PIO0_7/$\overline{\text{CTS0}}$/ CT32B1_CAP1/ CT32B1_MAT1	13[2]	yes	I; PU	I/O	**PIO0_7 —** General purpose digital input/output pin. Also serves as wake-up pin from Deep-sleep mode.
				I	**$\overline{\text{CTS0}}$ —** Clear To Send input for UART0.
				I	**CT32B1_CAP1 —** Capture input, channel 1 for 32-bit timer 1.
				O	**CT32B1_MAT1 —** Match output, channel 1 for 32-bit timer 1.
PIO0_8/RXD1 /CT32B1_CAP2/ CT32B1_MAT2	14[2]	yes	I; PU	I/O	**PIO0_8 —** General purpose digital input/output pin. Also serves as wake-up pin from Deep-sleep mode.
				I	**RXD1 —** Receiver input for UART1.
				I	**CT32B1_CAP2 —** Capture input, channel 2 for 32-bit timer 1.
				O	**CT32B1_MAT2 —** Match output, channel 2 for 32-bit timer 1.
PIO0_9/TXD1/ CT32B1_CAP3/ CT32B1_MAT3	15[2]	yes	I; PU	I/O	**PIO0_9 —** General purpose digital input/output pin. Also serves as wake-up pin from Deep-sleep mode.
				O	**TXD1 —** Transmitter output for UART1.
				I	**CT32B1_CAP3 —** Capture input, channel 3 for 32-bit timer 1.
				O	**CT32B1_MAT3 —** Match output, channel 3 for 32-bit timer 1.
PIO0_10/SCL	17[3]	yes	I; IA	I/O	**PIO0_10 —** General purpose digital input/output pin. Also serves as wake-up pin from Deep-sleep mode.
				I/O	**SCL —** I^2C-bus clock input/output.
PIO0_11/SDA/ CT16B0_CAP0/ CT16B0_MAT0	18[3]	yes	I; IA	I/O	**PIO0_11 —** General purpose digital input/output pin. Also serves as wake-up pin from Deep-sleep mode.
				I/O	**SDA —** I^2C-bus data input/output.
				I	**CT16B0_CAP0 —** Capture input, channel 0 for 16-bit timer 0.
				O	**CT16B0_MAT0 —** Match output, channel 0 for 16-bit timer 0.
PIO0_12/CLKOUT/ CT16B0_CAP1/ CT16B0_MAT1	19[7]	no	I; PU	I/O	**PIO0_12 —** General purpose digital input/output pin. A LOW level on this pin in during reset starts the ISP command handler. High-current output driver.
				O	**CLKOUT —** Clock out pin.
				I	**CT16B0_CAP1 —** Capture input, channel 0 for 16-bit timer 0.
				O	**CT16B0_MAT1 —** Match output, channel 1 for 16-bit timer 0.

Table 119. LPC12D27 LQFP100 pin description *…continued*

Symbol	Pin	Start logic input	Reset state [1]	Type	Description
$\overline{\text{RESET}}$/PIO0_13	20[4]	no	I; PU	I	**$\overline{\text{RESET}}$ —** External reset input: A LOW on this pin resets the device, causing I/O ports and peripherals to take on their default states, and processor execution to begin at address 0.
				I/O	**PIO0_13 —** General purpose digital input/output pin.
PIO0_14/SCK	21[2]	no	I; PU	I/O	**PIO0_14 —** General purpose digital input/output pin.
				I/O	**SCK —** Serial clock for SSP.
PIO0_15/SSEL/ CT16B1_CAP0/ CT16B1_MAT0	22[2]	no	I; PU	I/O	**PIO0_15 —** General purpose digital input/output pin.
				I/O	**SSEL —** Slave select for SSP.
				I	**CT16B1_CAP0 —** Capture input, channel 0 for 16-bit timer 1.
				O	**CT16B1_MAT0 —** Match output, channel 0 for 16-bit timer 1.
PIO0_16/MISO/ CT16B1_CAP1/ CT16B1_MAT1	23[2]	no	I; PU	I/O	**PIO0_16 —** General purpose digital input/output pin.
				I/O	**MISO —** Master In Slave Out for SSP.
				I	**CT16B1_CAP1 —** Capture input, channel 1 for 16-bit timer 1.
				O	**CT16B1_MAT1 —** Match output, channel 1 for 16-bit timer 1.
PIO0_17/MOSI	24[2]	no	I; PU	I/O	**PIO0_17 —** General purpose digital input/output pin.
				I/O	**MOSI —** Master Out Slave In for SSP.
PIO0_18/SWCLK/ CT32B0_CAP0/ CT32B0_MAT0	25[2]	no	I; PU	I/O	**PIO0_18 —** General purpose digital input/output pin.
				I	**SWCLK —** Serial wire clock, alternate location.
				I	**CT32B0_CAP0 —** Capture input, channel 0 for 32-bit timer 0.
				O	**CT32B0_MAT0 —** Match output, channel 0 for 32-bit timer 0.
PIO0_19/ACMP0_I0/ CT32B0_CAP1/ CT32B0_MAT1	95[5]	no	I; PU	I/O	**PIO0_19 —** General purpose digital input/output pin.
				I	**ACMP0_I0 —** Input 0 for comparator 0.
				I	**CT32B0_CAP1 —** Capture input, channel 1 for 32-bit timer 0.
				O	**CT32B0_MAT1 —** Match output, channel 1 for 32-bit timer 0
PIO0_20/ACMP0_I1/ CT32B0_CAP2/ CT32B0_MAT2	96[5]	no	I; PU	I/O	**PIO0_20 —** General purpose digital input/output pin.
				I	**ACMP0_I1 —** Input 1 for comparator 0.
				I	**CT32B0_CAP2 —** Capture input, channel 2 for 32-bit timer 0.
				O	**CT32B0_MAT2 —** Match output, channel 2 for 32-bit timer 0.
PIO0_21/ACMP0_I2/ CT32B0_CAP3/ CT32B0_MAT3	97[5]	no	I; PU	I/O	**PIO0_21 —** General purpose digital input/output pin.
				I	**ACMP0_I2 —** Input 2 for comparator 0.
				I	**CT32B0_CAP3 —** Capture input, channel 3 for 32-bit timer 0.
				O	**CT32B0_MAT3 —** Match output, channel 3 for 32-bit timer 0.
PIO0_22/ACMP0_I3	98[5]	no	I; PU	I/O	**PIO0_22 —** General purpose digital input/output pin.
				I	**ACMP0_I3 —** Input 3 for comparator 0.
PIO0_23/ ACMP1_I0/ CT32B1_CAP0/ CT32B1_MAT0	99[5]	no	I; PU	I/O	**PIO0_23 —** General purpose digital input/output pin.
				I	**ACMP1_I0 —** Input 0 for comparator 1.
				I	**CT32B1_CAP0 —** Capture input, channel 0 for 32-bit timer 1.
				O	**CT32B1_MAT0 —** Match output, channel 0 for 32-bit timer 1.

Table 119. LPC12D27 LQFP100 pin description *…continued*

Symbol	Pin	Start logic input	Reset state [1]	Type	Description
PIO0_24/ACMP1_I1/ CT32B1_CAP1/ CT32B1_MAT1	100[5]	no	I; PU	I/O	**PIO0_24 —** General purpose digital input/output pin.
				I	**ACMP1_I1 —** Input 1 for comparator 1.
				I	**CT32B1_CAP1 —** Capture input, channel 1 for 32-bit timer 1.
				O	**CT32B1_MAT1 —** Match output, channel 1 for 32-bit timer 1.
SWDIO/ACMP1_I2/ CT32B1_CAP2/ CT32B1_MAT2/PIO0_25	1[5]	no	I; PU	I/O	**SWDIO —** Serial wire debug input/output, default location.
				I	**ACMP1_I2 —** Input 2 for comparator 1.
				I	**CT32B1_CAP2 —** Capture input, channel 2 for 32-bit timer 1.
				O	**CT32B1_MAT2 —** Match output, channel 2 for 32-bit timer 1.
				I/O	**PIO0_25 —** General purpose digital input/output pin.
SWCLK/ ACMP1_I3/ CT32B1_CAP3/ CT32B1_MAT3/PIO0_26	2[5]	no	I; PU	I	**SWCLK —** Serial wire clock, default location.
				I	**ACMP1_I3 —** Input 3 for comparator 1.
				I	**CT32B1_CAP3 —** Capture input, channel 3 or 32-bit timer 1.
				O	**CT32B1_MAT3 —** Match output, channel 3 for 32-bit timer 1.
				I/O	**PIO0_26 —** General purpose digital input/output pin.
PIO0_27/ACMP0_O	3[7]	no	I; PU	I/O	**PIO0_27 —** General purpose digital input/output pin (high-current output driver).
				O	**ACMP0_O —** Output for comparator 0.
PIO0_28/ACMP1_O/ CT16B0_CAP0/ CT16B0_MAT0	4[7]	no	I; PU	I/O	**PIO0_28 —** General purpose digital input/output pin (high-current output driver).
				O	**ACMPC1_O —** Output for comparator 1.
				I	**CT16B0_CAP0 —** Capture input, channel 0 for 16-bit timer 0.
				O	**CT16B0_MAT0 —** Match output, channel 0 for 16-bit timer 0.
PIO0_29/ROSC/ CT16B0_CAP1/ CT16B0_MAT1	5[7]	no	I; PU	I/O	**PIO0_29 —** General purpose digital input/output pin (high-current output driver).
				I/O	**ROSC —** Relaxation oscillator for 555 timer applications.
				I	**CT16B0_CAP1 —** Capture input, channel 1 for 16-bit timer 0.
				O	**CT16B0_MAT1 —** Match output, channel 1 for 16-bit timer 0.
R/PIO0_30/AD0	26[5]	no	I; PU	I	**R —** Reserved. Configure for an alternate function in the IOCONFIG block.
				I/O	**PIO0_30 —** General purpose digital input/output pin.
				I	**AD0 —** A/D converter, input 0.
R/PIO0_31/AD1	27[5]	no	I; PU	I	**R —** Reserved. Configure for an alternate function in the IOCONFIG block.
				I/O	**PIO0_31 —** General purpose digital input/output pin.
				I	**AD1 —** A/D converter, input 1.
PIO1_0 to PIO1_6				I/O	**Port 1 —** Port 1 is a 32-bit I/O port with individual direction and function controls for each bit. The operation of port 1 pins depends on the function selected through the IOCONFIG register block. Pins PIO1_7 through PIO1_31 are not available.

Table 119. LPC12D27 LQFP100 pin description *…continued*

Symbol	Pin	Start logic input	Reset state [1]	Type	Description
R/PIO1_0/AD2	28[5]	no	I; PU	O	**R —** Reserved. Configure for an alternate function in the IOCONFIG block.
				I/O	**PIO1_0 —** General purpose digital input/output pin.
				I	**AD2 —** A/D converter, input 2.
R/PIO1_1/AD3	80[5]	no	I; PU	I	**R —** Reserved. Configure for an alternate function in the IOCONFIG block.
				I/O	**PIO1_1 —** General purpose digital input/output pin.
				I	**AD3 —** A/D converter, input 3.
PIO1_2/SWDIO/AD4	81[5]	no	I; PU	I/O	**PIO1_2 —** General purpose digital input/output pin.
				I/O	**SWDIO —** Serial wire debug input/output, alternate location.
				I	**AD4 —** A/D converter, input 4.
PIO1_3/AD5/WAKEUP	82[6]	no	I; PU	I/O	**PIO1_3 —** General purpose digital input/output pin.
				I	**AD5 —** A/D converter, input 5.
				I	**WAKEUP —** Deep power-down mode wake-up pin.
PIO1_4/AD6	83[5]	no	I; PU	I/O	**PIO1_4 —** General purpose digital input/output pin.
				I	**AD6 —** A/D converter, input 6.
PIO1_5/AD7/ CT16B1_CAP0/ CT16B1_MAT0	84[5]	no	I; PU	I/O	**PIO1_5 —** General purpose digital input/output pin.
				I	**AD7 —** A/D converter, input 7.
				I	**CT16B1_CAP0 —** Capture input, channel 0 for 16-bit timer 1.
				O	**CT16B1_MAT0 —** Match output, channel 0 for 16-bit timer 1.
PIO1_6/CT16B1_CAP1/ CT16B1_MAT1	85[2]	no	I; PU	I/O	**PIO1_6 —** General purpose digital input/output pin.
				I	**CT16B1_CAP1 —** Capture input, channel 1 for 16-bit timer 1.
				O	**CT16B1_MAT1 —** Match output, channel 1 for 16-bit timer 1.
PIO2_0				I/O	**Port 2 —** Port 2 is a 32-bit I/O port with individual direction and function controls for each bit. The operation of port 2 pins depends on the function selected through the IOCONFIG register block. Pins PIO2_1 through PIO2_31 are not available.
PIO2_0/CT16B0_CAP0/ CT16B0_MAT0	16[2]	no	I; PU	I/O	**PIO2_0 —** General purpose digital input/output pin.
				I	**CT16B0_CAP0 —** Capture input, channel 0 for 16-bit timer 0.
				O	**CT16B0_MAT0 —** Match output, channel 0 for 16-bit timer 0.
RTCXIN	89	-	-	-	Input to the 32 kHz oscillator circuit.
RTCXOUT	88	-	-	-	Output from the 32 kHz oscillator amplifier.
XTALIN	92	-	-	-	Input to the system oscillator circuit and internal clock generator circuits.
XTALOUT	93	-	-	-	Output from the system oscillator amplifier.
VREF_CMP	94	-	-	-	Reference voltage for comparator.
$V_{DD(IO)}$	90	-	-	-	Input/output supply voltage.
$V_{DD(3V3)}$	87	-	-	-	3.3 V supply voltage to the internal regulator and the ADC. Also used as the ADC reference voltage.
V_{SSIO}	91	-	-	-	Ground.

Table 119. LPC12D27 LQFP100 pin description *…continued*

Symbol	Pin	Start logic input	Reset state [1]	Type	Description
V_{SS}	86	-	-	-	Ground.
LCD display pins					
S0	46	-	-	O	LCD segment output.
S1	47	-	-	O	LCD segment output.
S2	48	-	-	O	LCD segment output.
S3	49	-	-	O	LCD segment output.
S4	50	-	-	O	LCD segment output.
S5	51	-	-	O	LCD segment output.
S6	52	-	-	O	LCD segment output.
S7	53	-	-	O	LCD segment output.
S8	54	-	-	O	LCD segment output.
S9	55	-	-	O	LCD segment output.
S10	56	-	-	O	LCD segment output.
S11	57	-	-	O	LCD segment output.
S12	58	-	-	O	LCD segment output.
S13	59	-	-	O	LCD segment output.
S14	60	-	-	O	LCD segment output.
S15	61	-	-	O	LCD segment output.
S16	62	-	-	O	LCD segment output.
S17	63	-	-	O	LCD segment output.
S18	64	-	-	O	LCD segment output.
S19	65	-	-	O	LCD segment output.
S20	66	-	-	O	LCD segment output.
S21	67	-	-	O	LCD segment output.
S22	68	-	-	O	LCD segment output.
S23	69	-	-	O	LCD segment output.
S24	70	-	-	O	LCD segment output.
S25	71	-	-	O	LCD segment output.
S26	72	-	-	O	LCD segment output.
S27	73	-	-	O	LCD segment output.
S28	74	-	-	O	LCD segment output.
S29	75	-	-	O	LCD segment output.
S30	76	-	-	O	LCD segment output.
S31	77	-	-	O	LCD segment output.
S32	78	-	-	O	LCD segment output.
S33	79	-	-	O	LCD segment output.
S34	29	-	-	O	LCD segment output.
S35	30	-	-	O	LCD segment output.
S36	31	-	-	O	LCD segment output.

Table 119. LPC12D27 LQFP100 pin description *…continued*

Symbol	Pin	Start logic input	Reset state [1]	Type	Description
S37	32	-	-	O	LCD segment output.
S38	33	-	-	O	LCD segment output.
S39	34	-	-	O	LCD segment output.
BP0	42	-	-	O	LCD backplane output.
BP1	44	-	-	O	LCD backplane output.
BP2	43	-	-	O	LCD backplane output.
BP3	45	-	-	O	LCD backplane output.
LCD_SDA	35	-	-	I/O	I^2C-bus serial data input/output.
LCD_SCL	36	-	-	I/O	I^2C-bus serial clock input.
SYNC	37	-	-	I/O	Cascade synchronization input/output.
CLK	38	-	-	I/O	External clock input/output.
V_{DD}	39	-	-	-	1.8 V to 5.5 V power supply: Power supply voltage for the PCF8576D.
$V_{SS(LCD)}$	40	-	-	-	LCD ground.
V_{LCD}	41	-	-	-	LCD power supply: LCD voltage.

[1] Pin state at reset for default function: I = Input; O = Output; PU = internal pull-up enabled; IA = inactive, no pull-up/down enabled.

[2] Digital I/O pin; default: pull-up enabled, no hysteresis.

[3] I^2C-bus pins; 5 V tolerant; open-drain; default: no pull-up/pull-down, no hysteresis.

[4] Digital I/O pin with RESET function; default: pull-up enabled, no hysteresis.

[5] Digital I/O pin with analog function; default: pull-up enabled, no hysteresis.

[6] Digital I/O pin with analog function and WAKEUP function; default: pull-up enabled, no hysteresis.

[7] High-drive digital I/O pin; default: pull-up enabled, no hysteresis.

7.3 Pin multiplexing

To enable a peripheral function on a pin, find the corresponding port pin, or select a port pin if the function is multiplexed, and program the port pin's IOCONFIG register to enable that function. The primary SWD functions and RESET are the default functions on their pins after reset, all other digital pins default to GPIO.

Table 120. Pin multiplexing

Peripheral	Function	Type	Available on ports:		
Analog comparators	ROSC	I/O	PIO0_29	-	-
	ACMP0_I0	I	PIO0_19	-	-
	ACMP0_I1	I	PIO0_20	-	-
	ACMP0_I2	I	PIO0_21	-	-
	ACMP0_I3	I	PIO0_22	-	-
	ACMP0_O	O	PIO0_27	-	-
	ACMP1_I0	I	PIO0_23	-	-
	ACMP1_I1	I	PIO0_24	-	-
	ACMP1_I2	I	PIO0_25	-	-

Table 120. Pin multiplexing

Peripheral	Function	Type	Available on ports:		
	ACMP1_I3	I	PIO0_26	-	-
	ACMP1_O	O	PIO0_28	-	-
ADC	AD0	I	PIO0_30	-	-
	AD1	I	PIO0_31	-	-
	AD2	I	PIO1_0	-	-
	AD3	I	PIO1_1	-	-
	AD4	I	PIO1_2	-	-
	AD5	I	PIO1_3	-	-
	AD6	I	PIO1_4	-	-
	AD7	I	PIO1_5	-	-
CT16B0	CT16B0_CAP0	I	PIO0_11	PIO0_28	PIO2_0
	CT16B0_CAP1	I	PIO0_12	PIO0_29	PIO2_1
	CT16B0_MAT0	O	PIO0_11	PIO0_28	PIO2_0
	CT16B0_MAT1	O	PIO0_12	PIO0_29	PIO2_1
CT16B1	CT16B1_CAP0	I	PIO0_15	PIO1_5	PIO2_2
	CT16B1_CAP1	I	PIO0_16	PIO1_6	PIO2_3
	CT16B1_MAT0	O	PIO0_15	PIO1_5	PIO2_2
	CT16B1_MAT1	O	PIO0_16	PIO1_6	PIO2_3
CT32B0	CT32B0_CAP0	I	PIO0_1	PIO0_18	PIO2_4
	CT32B0_CAP1	I	PIO0_2	PIO0_19	PIO2_5
	CT32B0_CAP2	I	PIO0_3	PIO0_20	PIO2_6
	CT32B0_CAP3	I	PIO0_4	PIO0_21	PIO2_7
	CT32B0_MAT0	O	PIO0_1	PIO0_18	PIO2_4
	CT32B0_MAT1	O	PIO0_2	PIO0_19	PIO2_5
	CT32B0_MAT2	O	PIO0_3	PIO0_20	PIO2_6
	CT32B0_MAT3	O	PIO0_4	PIO0_21	PIO2_7
CT32B1	CT32B1_CAP0	I	PIO0_6	PIO0_23	PIO2_8
	CT32B1_CAP1	I	PIO0_7	PIO0_24	PIO2_9
	CT32B1_CAP2	I	PIO0_8	PIO0_25	PIO2_10
	CT32B1_CAP3	I	PIO0_9	PIO0_26	PIO2_11
	CT32B1_MAT0	O	PIO0_6	PIO0_23	PIO2_8
	CT32B1_MAT1	O	PIO0_7	PIO0_24	PIO2_9
	CT32B1_MAT2	O	PIO0_8	PIO0_25	PIO2_10
	CT32B1_MAT3	O	PIO0_9	PIO0_26	PIO2_11

Table 120. Pin multiplexing

Peripheral	Function	Type	Available on ports:		
UART0	RXD0	I	PIO0_1	PIO2_1	-
	TXD0	O	PIO0_2	PIO2_2	-
	$\overline{\text{CTS0}}$	I	PIO0_7	PIO2_4	-
	$\overline{\text{DCD0}}$	I	PIO0_5	PIO2_6	-
	$\overline{\text{DSR0}}$	I	PIO0_4	PIO2_7	-
	$\overline{\text{DTR0}}$	O	PIO0_3	PIO2_3	-
	$\overline{\text{RI0}}$	I	PIO0_6	PIO2_5	-
	$\overline{\text{RTS0}}$	O	PIO0_0	PIO2_0	-
UART1	RXD1	I	PIO0_8	PIO2_11	PIO2_12
	TXD1	O	PIO0_9	PIO2_10	PIO2_13
SSP/SPI	SCK	I/O	PIO0_14	-	-
	MISO	I/O	PIO0_16	-	-
	MOSI	I/O	PIO0_17	-	-
	SSEL	I/O	PIO0_15	-	-
I2C	SCL	I/O	PIO0_10	-	-
	SDA	I/O	PIO0_11	-	-
SWD	SWCLK[1]	I	PIO0_18	PIO0_26	-
	SWDIO[1]	I/O	PIO0_25	PIO1_2	-
Reset	$\overline{\text{RESET}}$	I	PIO0_13	-	-
Clockout pin	CLKOUT	O	PIO0_12	-	-

[1] After reset, the SWD functions are selected by default on pins PIO0_26 and PIO0_25.

8.1 How to read this chapter

Each pin has one bit in one of the GPIO registers assigned. GPIO registers for pins not available are reserved (see Table 121). GPIO port 2 registers are only available on 64-pin packages.

Table 121. Available GPIO pins/ports

Port	Pins	GPIO register bits used	LQFP48	LQFP64	LQFP100[1]
GPIO0	PIO0_0 to PIO0_31	31:0	yes	yes	yes
GPIO1	PIO1_0 to PIO1_6	6:0	yes	yes	yes
GPIO2	PIO2_0 to PIO2_15	15:0	no	yes	yes

[1] For part LPC12D27.

8.2 Introduction

8.2.1 Features

- Digital ports can be configured as input or output by software.
- Read and write data operations from/to the port pins are maskable.
- Bit-level set and clear registers allow a single-instruction set or clear of any number of pins in one port.
- Bit-level invert registers allow inverting the output of any number of pins in one port.
- Each individual port pin can serve as external interrupt input.
- Interrupts can be configured on single falling or rising edges and on both edges.
- Individual interrupt levels can be programmed.
- All GPIO pins are configured as inputs (with pull-up resistors enabled, see Section 6.3.2) after reset.

8.3 Register description

Each GPIO register is 32-bit wide. The MASK register masks all operations on the PIN, OUT, SET, CLR, and NOT registers. The registers DIR to IC are not affected by the bits set in the MASK register.

Each bit in the GPIO registers represents one GPIO pin. The pin configuration for each port determines which pins are used or reserved:

- Port 0: All GPIO0 registers use bits 0 to 31.
- Port 1: All GPIO1 registers use bits 0 to 6. Bits 7 to 31 are reserved.
- Port 2: All GPIO2 registers use bits 0 to 15. Bits 16 to 31 are reserved.

Remark: Do not write 1's to the reserved bits of ports GPIO1 and GPIO2.

Table 122. Register overview: GPIO (base address port 0: 0x5000 0000; port 1: 0x5001 0000, port 2: 0x5002 0000)

Name	Access	Address offset	Description	Reset value
MASK	R/W	0x000	Pin value mask register. Affects operations on PIN, OUT, SET, CLR, and NOT registers.	0x0000 0000
PIN	R	0x004	Pin value register.	configuration dependent
OUT	R/W	0x008	Pin output value register.	0x0000 0000
SET	W	0x00C	Pin output value set register.	n/a
CLR	W	0x010	Pin output value clear register.	n/a
NOT	W	0x014	Pin output value invert register.	0x0000 0000
DIR	R/W	0x020	Data direction register.	0x0000 0000
IS	R/W	0x024	Interrupt sense register.	0x0000 0000
IBE	R/W	0x028	Interrupt both edges register.	0x0000 0000
IEV	R/W	0x02C	Interrupt event register.	0x0000 0000
IE	R/W	0x030	Interrupt mask register.	0x0000 0000
RIS	R	0x034	Raw interrupt status register.	0x0000 0000
MIS	R	0x038	Masked interrupt status register.	0x0000 0000
IC	W	0x03C	Interrupt clear register.	0x0000 0000
-	-	0x040	Reserved.	0x0000 0000

8.3.1 GPIO mask register

This register masks the read and/or write accesses to the following masked registers: PIN, OUT, SET, CLR, and NOT. Only bits set to 0 in the MASK register enable the corresponding bits in the masked registers to be changed or their value to be read.

Setting any mask bit to 0 allows the pin output to be changed by write operations to the pin's OUT, SET, CLR, and NOT registers. The current state of the pin can be read from the PIN registers and the current value of the OUT registers can be read.

Setting any mask bit to 1 allows write operations to the pin's OUT, SET, CLR, and NOT registers to have no effect on the pin's output level. Read operations return 0 regardless of the pin's level or the value of the OUT register.

Table 123. GPIO mask register (MASK - address 0x5000 0000 (GPIO0), 0x5001 0000 (GPIO1), 0x5002 0000 (GPIO2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	MASK	GPIO pin PIOn_x access control. 0 = read/write not masked. 1 = read/write masked.	0x0	R/W

8.3.2 GPIO pin value register

This register provides the current logic state of port pins that are configured to perform digital functions. A read operation on this register will return the logic value of the pin regardless of whether the pin is configured for input or output, or whether it is configured as GPIO or any other applicable alternate digital function. As an example, a particular port

pin may have GPIO input, GPIO output, and counter/timer match output and capture input as selectable functions. Through the PIN register, the current logic state of the pin can be read in any configuration, e.g. the state of the capture input could be read.

As an exception, the pin state cannot be read if its analog function is selected (if applicable) because selecting the pin as an ADC input disconnects the digital features of the pin. In that case, the pin value read in the PIN register is not valid.

Note that read operations are masked by the MASK bits. Read operations on masked bits always return 0 regardless of the pin's actual level.

Table 124. GPIO pin value register (PIN - address 0x5000 0004 (GPIO0), 0x5001 0004 (GPIO1); 0x5002 0004 (GPIO2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	PIN	GPIO pin PIOn_x value. 0 = Digital pin level is LOW. 1 = Digital pin level is HIGH.	0x0	R

8.3.3 GPIO pin output register

Writing 0 or 1 to this register produces LOW or HIGH levels at the corresponding port pins. The port pin is set to this value if it is configured as GPIO output. For all other configurations (input, non-GPIO function), the value of the OUT register bit has no effect on the pin output level. Write operations are masked by the MASK registers.

Reading this register returns the contents of the GPIO output register regardless of the digital pin configuration and direction. Read operations are masked by the MASK registers.

The SET, CLR, and NOT registers write to the OUT register to allow bit-wise setting, clearing, and inverting of individual port pins. The port output state is determined by the contents of the OUT register only.

Table 125. GPIO pin output register (OUT - address 0x5000 0008 (GPIO0), 0x5001 0008 (GPIO1), 0x5002 0008 (GPIO2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	OUT	GPIO pin PIOn_x output value. 0 = Write: Set GPIO output pin to LOW. Read: GPIO output value is LOW. 1 = Write: Set GPIO output pin to HIGH. Read: GPIO output value is HIGH.	0x00	R/W

8.3.4 GPIO pin output set register

This register is used to produce a HIGH level output at the port pins configured as GPIO output in the DIR register (see Table 129) and as GPIO in the corresponding IOCONFIG register (see Table 60). Writing 1 sets the corresponding port pin to HIGH. Writing 0 has no effect on the GPIO output level. If a pin is not configured as GPIO and output, the SET register has no effect on the pin level.

This register is a write-only register. Note that write operations to the SET register are masked by the MASK register.

Table 126. GPIO pin output set register (SET - address 0x5000 000C (GPIO0), 0x5001 000C (GPIO1), 0x5002 000C (GPIO2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	SET	Set GPIO pin PIOn_x output value. 0 = No effect on the GPIO output level. 1 = GPIO output is set to HIGH.	0x00	W

8.3.5 GPIO pin output clear register

This register is used to produce a LOW level output at the port pins configured as GPIO output in the DIR register (see Table 129) and as GPIO in the corresponding IOCONFIG register (see Table 60). Writing 1 sets the corresponding port pin to LOW. Writing 0 has no effect on the GPIO output level. If a pin is not configured as GPIO and output, the CLR register has no effect on the pin level.

This register is a write-only register. Note that write operations to the CLR register are masked by the MASK register.

Remark: Do not write 1's to the reserved bits of ports GPIO1 and GPIO2. See Section 8.1.

Table 127. GPIO pin output clear register (CLR - address 0x5000 0010 (GPIO0), 0x5000 1010 (GPIO1), 0x5002 0010 (GPIO2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	CLEAR	Clear GPIO pin PIOn_x output value. 0 = No effect on the GPIO output level. 1 = GPIO output is set to LOW.	0x00	W

8.3.6 GPIO NOT register

This register is used to invert the output level at the port pins configured as GPIO output in the DIR register (see Table 129) and as GPIO in the corresponding IOCONFIG register (see Table 60). Writing 1 inverts the corresponding port pin. Writing 0 has no effect on the GPIO output level. If a pin is not configured as GPIO and output, the NOT register has no effect on the pin level.

This register is a write-only register. Note that write operations to the NOT register are masked by the MASK register.

Table 128. GPIO NOT register (NOT - address 0x5000 0014 (GPIO0), 0x5001 0014 (GPIO1), 0x5002 0014 (GPIO2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	NOT	Invert GPIO pin PIOn_x output value. 0 = No effect on the GPIO output level. 1 = GPIO output is inverted from its current value.	0x00	-

8.3.7 GPIO data direction register

Remark: Do not write 1's to the reserved bits of ports GPIO1 and GPIO2. See Section 8.1.

Table 129. GPIO data direction register (DIR - address 0x5000 0020 (GPIO0), 0x5001 0020 (GPIO1), 0x5002 0020 (GPIO2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	IO	Selects GPIO pin PIOn_x as input or output. 0 = Pin PIOn_x is configured as input. 1 = Pin PIOn_x is configured as output.	0x00	R/W

8.3.8 GPIO interrupt sense register

Table 130. GPIO interrupt sense register (IS - address 0x5000 0024 (GPIO0), 0x5001 0024 (GPIO1), 0x5002 0024 (GPIO2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	ISENSE	Selects interrupt on pin PIOn_x as level or edge sensitive. 0 = Interrupt on pin PIOn_x is configured as edge sensitive. 1 = Interrupt on pin PIOn_x is configured as level sensitive.	0x00	R/W

8.3.9 GPIO interrupt both edges sense register

Table 131. GPIO interrupt both edges sense register (IBE - address 0x5000 0028 (GPIO0), 0x5001 0028 (GPIO1), 0x5002 0028 (GPIO2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	IBE	Selects interrupt on pin PIOn_x to be triggered on both edges. 0 = Interrupt on pin PIOn_x is controlled through register IEV. 1 = Both edges on pin PIOn_x trigger an interrupt.	0x00	R/W

8.3.10 GPIO interrupt event register

Table 132. GPIO interrupt event register (IEV - address 0x5000 002C (GPIO0), 0x5001 002C (GPIO1), 0x5002 002C (GPIO2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	IEV	Selects interrupt on pin PIOn_x to be triggered rising or falling edges. 0 = Depending on setting in register IS, falling edges or LOW level on pin PIOn_x trigger an interrupt. 1 = Depending on setting in register IS, rising edges or HIGH level on pin PIOn_x trigger an interrupt.	0x00	R/W

8.3.11 GPIO interrupt mask register

Bits set to HIGH in the IE register allow the corresponding pins to trigger their individual interrupts and the combined INTR line. Clearing a bit disables interrupt triggering on that pin.

Table 133. GPIO interrupt mask register (IE - address 0x5000 0030, 0x5001 0030 (GPIO1), 0x5002 0030 (GPIO2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	MASK	Selects interrupt on pin PIOn_x to be masked. 0 = Interrupt on pin PIOn_x is masked. 1 = Interrupt on pin PIOn_x is not masked.	0x00	R/W

8.3.12 GPIO raw interrupt status register

Bits read HIGH in the IRS register reflect the raw (prior to masking) interrupt status of the corresponding pins indicating that all the requirements have been met before they are allowed to trigger the IE. Bits read as zero indicate that the corresponding input pins have not initiated an interrupt. The register is read-only.

Table 134. GPIO raw interrupt mask status register (RIS - address 0x5000 0034 (GPIO0), 0x5001 0034 (GPIO1), 0x5002 0034 (GPIO2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	RAWST	Raw interrupt status. 0 = No interrupt on pin PIOn_x. 1 = Interrupt requirements met on PIOn_x.	0x00	R

8.3.13 GPIO masked interrupt status register

Bits read HIGH in the MIS register reflect the status of the input lines triggering an interrupt. Bits read as LOW indicate that either no interrupt on the corresponding input pins has been generated or that the interrupt is masked. MIS is the state of the interrupt after masking. The register is read-only.

Table 135. GPIO masked interrupt status register (MIS - address 0x5000 0038 (GPIO0), 0x5001 0038 (GPIO1), 0x5002 0038 (GPIO2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	MASK	Selects interrupt on pin PIOn_x to be masked. 0 = No interrupt or interrupt masked on pin PIOn_x. 1 = Interrupt on PIOn_x.	0x00	R

8.3.14 GPIO interrupt clear register

Remark: The synchronizer between the GPIO and the NVIC blocks causes a delay of 2 clocks. It is recommended to add two NOPs after the clear of the interrupt edge detection logic, before the exit of the interrupt service routine.

Table 136. GPIO interrupt clear register (IC - address 0x5000 003C, 0x5001 003C (GPIO1), 0x5002 003C (GPIO2)) bit description

Bit	Symbol	Description	Reset value	Access
31:0	CLR	Selects interrupt on pin PIOn_x to be cleared. Clears the interrupt edge detection logic. 0 = No effect. 1 = Clears edge detection logic for pin PIOn_x.	0x00	W

9.1 How to read this chapter

UART0 is available on all LPC122x parts.

9.2 Basic configuration

Clocks and power to the UART0 block are controlled by:

1. the SYSAHBCLKCTRL register (see Table 21).
2. the UART0_PCLK which is enabled in the UART0 clock divider register (see Table 23). This clock is used by the UART baud rate generator.

Remark: The UART0 pins must be configured in the corresponding IOCON registers **before** the UART0 clocks are enabled.

The UART0_PCLK can be disabled in the UART0CLKDIV register (see Table 23) and the UART block can be disabled through the System AHB clock control register bit 12 (see Table 21) for power savings.

9.3 Features

- 16-byte receive and transmit FIFOs.
- Register locations conform to ‘550 industry standard.
- Receiver FIFO trigger points at 1, 4, 8, and 14 byte.
- Built-in baud rate generator.
- UART allows for implementation of either software or hardware flow control.
- RS-485/EIA-485 9-bit mode support with output enable.
- Modem control.

9.4 Pin description

Table 137. UART0 pin description

Pin	Type	Description
RXD0	Input	**Serial Input.** Serial receive data.
TXD0	Output	**Serial Output.** Serial transmit data.
$\overline{\text{RTS0}}$	Output	Request To Send. RS-485 direction control pin.
$\overline{\text{DTR0}}$	Output	Data Terminal Ready.
$\overline{\text{DSR0}}$	Input	Data Set Ready.
$\overline{\text{CTS0}}$	Input	Clear To Send.
$\overline{\text{DCD0}}$	Input	Data Carrier Detect.
$\overline{\text{RI0}}$	Input	Ring Indicator.

9.5 Register description

The UART contains registers organized as shown in Table 138. The Divisor Latch Access Bit (DLAB) is contained in LCR[7] and enables access to the Divisor Latches.

Table 138. Register overview: UART0 (base address: 0x4000 8000)

Name	Access	Address offset	Description	Reset value[1]
RBR	RO	0x000	Receiver Buffer Register. Contains the next received character to be read. (DLAB=0)	NA
THR	WO	0x000	Transmit Holding Register. The next character to be transmitted is written here. (DLAB=0)	NA
DLL	R/W	0x000	Divisor Latch LSB. Least significant byte of the baud rate divisor value. The full divisor is used to generate a baud rate from the fractional rate divider. (DLAB = 1)	0x01
DLM	R/W	0x004	Divisor Latch MSB. Most significant byte of the baud rate divisor value. The full divisor is used to generate a baud rate from the fractional rate divider. (DLAB = 1)	0x00
IER	R/W	0x004	Interrupt Enable Register. Contains individual interrupt enable bits for the 7 potential UART interrupts. (DLAB=0)	0x00
IIR	RO	0x008	Interrupt ID Register. Identifies which interrupt(s) are pending.	0x01
FCR	WO	0x008	FIFO Control Register. Controls UART FIFO usage and modes.	0x00
LCR	R/W	0x00C	Line Control Register. Contains controls for frame formatting and break generation.	0x00
MCR	R/W	0x010	Modem control register	0x00
LSR	RO	0x014	Line Status Register. Contains flags for transmit and receive status, including line errors.	0x60
MSR	RO	0x018	Modem status register	0x00
SCR	R/W	0x01C	Scratch Pad Register. Eight-bit temporary storage for software.	0x00
ACR	R/W	0x020	Auto-baud Control Register. Contains controls for the auto-baud feature.	0x00
-	-	0x024	Reserved	-
FDR	R/W	0x028	Fractional Divider Register. Generates a clock input for the baud rate divider.	0x10
-	-	0x02C	Reserved	-
TER	R/W	0x030	Transmit Enable Register. Turns off UART transmitter for use with software flow control.	0x80
-	-	0x034 - 0x048	Reserved	-
RS485CTRL	R/W	0x04C	RS-485/EIA-485 Control. Contains controls to configure various aspects of RS-485/EIA-485 modes.	0x00
ADRMATCH	R/W	0x050	RS-485/EIA-485 address match. Contains the address match value for RS-485/EIA-485 mode.	0x00
RS485DLY	R/W	0x054	RS-485/EIA-485 direction control delay.	0x00
FIFOLVL	RO	0x058	FIFO Level register. Provides the current fill levels of the transmit and receive FIFOs.	0x00

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

9.5.1 UART Receiver Buffer Register (when DLAB = 0, Read Only)

The RBR is the top byte of the UART RX FIFO. The top byte of the RX FIFO contains the oldest character received and can be read via the bus interface. The LSB (bit 0) represents the “oldest” received data bit. If the character received is less than 8 bits, the unused MSBs are padded with zeroes.

The Divisor Latch Access Bit (DLAB) in LCR must be zero in order to access the RBR. The RBR is always Read Only.

Since PE, FE and BI bits (see Table 150) correspond to the byte sitting on the top of the RBR FIFO (i.e. the one that will be read in the next read from the RBR), the right approach for fetching the valid pair of received byte and its status bits is first to read the content of the LSR register, and then to read a byte from the RBR.

Table 139. UART Receiver Buffer Register (RBR - address 0x4000 8000 when DLAB = 0, Read Only) bit description

Bit	Symbol	Description	Reset value
7:0	RBR	The UART Receiver Buffer Register contains the oldest received byte in the UART RX FIFO.	undefined
31:8	-	Reserved	-

9.5.2 UART Transmitter Holding Register (when DLAB = 0, Write Only)

The THR is the top byte of the UART TX FIFO. The top byte is the newest character in the TX FIFO and can be written via the bus interface. The LSB represents the first bit to transmit.

The Divisor Latch Access Bit (DLAB) in LCR must be zero in order to access the THR. The THR is always Write Only.

Table 140. UART Transmitter Holding Register (THR - address 0x4000 8000 when DLAB = 0, Write Only) bit description

Bit	Symbol	Description	Reset value
7:0	THR	Writing to the UART Transmit Holding Register causes the data to be stored in the UART transmit FIFO. The byte will be sent when it reaches the bottom of the FIFO and the transmitter is available.	NA
31:8	-	Reserved	-

9.5.3 UART Divisor Latch LSB and MSB Registers (when DLAB = 1)

The UART Divisor Latch is part of the UART Baud Rate Generator and holds the value used, along with the Fractional Divider, to divide the UART_PCLK clock in order to produce the baud rate clock, which must be 16x the desired baud rate. The DLL and DLM registers together form a 16-bit divisor where DLL contains the lower 8 bits of the divisor and DLM contains the higher 8 bits of the divisor. A 0x0000 value is treated like a 0x0001 value as division by zero is not allowed.The Divisor Latch Access Bit (DLAB) in LCR must be one in order to access the UART Divisor Latches. Details on how to select the right value for DLL and DLM can be found in Section 9.5.13.

Table 141. UART Divisor Latch LSB Register (DLL - address 0x4000 8000 when DLAB = 1) bit description

Bit	Symbol	Description	Reset value
7:0	DLLSB	The UART Divisor Latch LSB Register, along with the DLM register, determines the baud rate of the UART.	0x01
31:8	-	Reserved	-

Table 142. UART Divisor Latch MSB Register (DLM - address 0x4000 8004 when DLAB = 1) bit description

Bit	Symbol	Description	Reset value
7:0	DLMSB	The UART Divisor Latch MSB Register, along with the DLL register, determines the baud rate of the UART.	0x00
31:8	-	Reserved	-

9.5.4 UART Interrupt Enable Register (when DLAB = 0)

The IER is used to enable the four UART interrupt sources.

Table 143. UART Interrupt Enable Register (IER - address 0x4000 8004 when DLAB = 0) bit description

Bit	Symbol	Value	Description	Reset value
0	RBRIE		RBR Interrupt Enable. Enables the Receive Data Available interrupt for UART. It also controls the Character Receive Time-out interrupt.	0
		0	Disable the RDA interrupts.	
		1	Enable the RDA interrupts.	
1	THREIE		THRE Interrupt Enable. Enables the THRE interrupt for UART. The status of this interrupt can be read from LSR[5].	0
		0	Disable the THRE interrupts.	
		1	Enable the THRE interrupts.	
2	RXLIE		RX Line Interrupt Enable. Enables the UART RX line status interrupts. The status of this interrupt can be read from LSR[4:1].	0
		0	Disable the RX line status interrupts.	
		1	Enable the RX line status interrupts.	
3	-	-	Reserved	-
6:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	-	-	Reserved	0
8	ABEOINTEN		Enables the end of auto-baud interrupt.	0
		0	Disable end of auto-baud Interrupt.	
		1	Enable end of auto-baud Interrupt.	
9	ABTOINTEN		Enables the auto-baud time-out interrupt.	0
		0	Disable auto-baud time-out Interrupt.	
		1	Enable auto-baud time-out Interrupt.	
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

9.5.5 UART Interrupt Identification Register

The IIR provides a status code that denotes the priority and source of a pending interrupt. The interrupts are frozen during an IIR access. If an interrupt occurs during an IIR access, the interrupt is recorded for the next IIR access.

Table 144. UART Interrupt Identification Register (IIR - address 0x4004 8008, Read Only) bit description

Bit	Symbol	Value	Description	Reset value
0	INTSTATUS		Interrupt status. Note that IIR[0] is active low. The pending interrupt can be determined by evaluating IIR[3:1].	1
		0	At least one interrupt is pending.	
		1	No interrupt is pending.	
3:1	INTID		Interrupt identification. IER[3:1] identifies an interrupt corresponding to the UART RX FIFO. All other combinations of IER[3:1] not listed below are reserved (100,101,111).	0
		0x3	1 - Receive Line Status (RLS).	
		0x2	2a - Receive Data Available (RDA).	
		0x6	2b - Character Time-out Indicator (CTI).	
		0x1	3 - THRE Interrupt.	
		0x0	4 - Modem interrupt.	
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	FIFOENA		These bits are equivalent to FCR[0].	0
8	ABEOINT		End of auto-baud interrupt. True if auto-baud has finished successfully and interrupt is enabled.	0
9	ABTOINT		Auto-baud time-out interrupt. True if auto-baud has timed out and interrupt is enabled.	0
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Bit IIR[9:8] are set by the auto-baud function and signal a time-out or end of auto-baud condition. The auto-baud interrupt conditions are cleared by setting the corresponding Clear bits in the Auto-baud Control Register.

If the IntStatus bit is 1 no interrupt is pending and the IntId bits will be zero. If the IntStatus is 0, a non auto-baud interrupt is pending in which case the IntId bits identify the type of interrupt and handling as described in Table 145. Given the status of IIR[3:0], an interrupt handler routine can determine the cause of the interrupt and how to clear the active interrupt. The IIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

The UART RLS interrupt (IIR[3:1] = 011) is the highest priority interrupt and is set whenever any one of four error conditions occur on the UART RX input: overrun error (OE), parity error (PE), framing error (FE) and break interrupt (BI). The UART RX error condition that set the interrupt can be observed via LSR[4:1]. The interrupt is cleared upon an LSR read.

The UART RDA interrupt (IIR[3:1] = 010) shares the second level priority with the CTI interrupt (IIR[3:1] = 110). The RDA is activated when the UART RX FIFO reaches the trigger level defined in FCR7:6 and is reset when the UART RX FIFO depth falls below the trigger level. When the RDA interrupt goes active, the CPU can read a block of data defined by the trigger level.

The CTI interrupt (IIR[3:1] = 110) is a second level interrupt and is set when the UART RX FIFO contains at least one character and no UART RX FIFO activity has occurred in 3.5 to 4.5 character times. Any UART RX FIFO activity (read or write of UART RSR) will clear the interrupt. This interrupt is intended to flush the UART RBR after a message has been received that is not a multiple of the trigger level size. For example, if a peripheral wished to send a 105 character message and the trigger level was 10 characters, the CPU would receive 10 RDA interrupts resulting in the transfer of 100 characters and 1 to 5 CTI interrupts (depending on the service routine) resulting in the transfer of the remaining 5 characters.

Table 145. UART Interrupt Handling

IIR[3:0] value[1]	Priority	Interrupt type	Interrupt source	Interrupt reset
0001	-	None	None	-
0110	Highest	RX Line Status / Error	OE[2] or PE[2] or FE[2] or BI[2]	LSR Read[2]
0100	Second	RX Data Available	RX data available or trigger level reached in FIFO (FCR0=1)	RBR Read[3] or UART FIFO drops below trigger level
1100	Second	Character Time-out indication	Minimum of one character in the RX FIFO and no character input or removed during a time period depending on how many characters are in FIFO and what the trigger level is set at (3.5 to 4.5 character times). The exact time will be: [(word length) × 7 - 2] × 8 + [(trigger level - number of characters) × 8 + 1] RCLKs	RBR Read[3]
0010	Third	THRE	THRE[2]	IIR Read[4] (if source of interrupt) or THR write

[1] Values "0000", “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”,”1101”,”1110”,”1111” are reserved.

[2] For details see Section 9.5.9 “UART Line Status Register”

[3] For details see Section 9.5.1 “UART Receiver Buffer Register (when DLAB = 0, Read Only)”

[4] For details see Section 9.5.5 “UART Interrupt Identification Register” and Section 9.5.2 “UART Transmitter Holding Register (when DLAB = 0, Write Only)”

The UART THRE interrupt (IIR[3:1] = 001) is a third level interrupt and is activated when the UART THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the UART THR FIFO a chance to fill up with data to eliminate many THRE interrupts from occurring at system start-up. The initialization conditions implement a one character delay minus the stop bit whenever THRE = 1 and there have not been at least two characters in the THR at one time since

the last THRE = 1 event. This delay is provided to give the CPU time to write data to THR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the UART THR FIFO has held two or more characters at one time and currently, the THR is empty. The THRE interrupt is reset when a THR write occurs or a read of the IIR occurs and the THRE is the highest interrupt (IIR[3:1] = 001).

9.5.6 UART FIFO Control Register

The FCR controls the operation of the UART RX and TX FIFOs.

Table 146. UART FIFO Control Register (FCR - address 0x4000 8008, Write Only) bit description

Bit	Symbol	Value	Description	Reset value
0	FIFOEN		FIFO Enable	0
		0	UART FIFOs are disabled. Must not be used in the application.	
		1	Active high enable for both UART RX and TX FIFOs and FCR[7:1] access. This bit must be set for proper UART operation. Any transition on this bit will automatically clear the UART FIFOs.	
1	RXFIFORS		RX FIFO Reset	0
		0	No impact on either of UART FIFOs.	
		1	Writing a logic 1 to FCR[1] will clear all bytes in UART RX FIFO, reset the pointer logic. This bit is self-clearing.	
2	TXFIFORS		TX FIFO Reset	0
		0	No impact on either of UART FIFOs.	
		1	Writing a logic 1 to FCR[2] will clear all bytes in UART TX FIFO, reset the pointer logic. This bit is self-clearing.	
3	DMAMODE		DMA Mode Select. When the FIFO enable bit (bit 0 of this register) is set, this bit selects the DMA mode. See Section 9.5.6.1.	0
		0	DMA not used.	
		1	DMA mode enabled.	
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	RXTL		RX Trigger Level. These two bits determine how many receiver UART FIFO characters must be written before an interrupt is activated.	0
		0x0	Trigger level 0 (1 character or 0x01).	
		0x1	Trigger level 1 (4 characters or 0x04).	
		0x2	Trigger level 2 (8 characters or 0x08).	
		0x3	Trigger level 3 (14 characters or 0x0E).	
31:8	-	-	Reserved	-

9.5.6.1 DMA Operation

The user can optionally operate the UART transmit and/or receive using the micro DMA. The DMA mode is determined by the DMA Mode Select bit in the FCR register. This bit only has an effect when the FIFOs are enabled via the FIFO Enable bit in the FCR register.

9.5.6.1.1 UART receiver DMA

In DMA mode, the receiver DMA request is asserted when the receiver FIFO level is equal to or greater than trigger level, or if a character time-out occurs. See the description of the RX Trigger Level above. The receiver DMA request is cleared by the DMA controller.

9.5.6.1.2 UART transmitter DMA

In DMA mode, the transmitter DMA request is asserted when the transmitter FIFO transitions to not full. The transmitter DMA request is cleared by the DMA controller.

9.5.7 UART Line Control Register

The LCR determines the format of the data character that is to be transmitted or received.

Table 147. UART Line Control Register (LCR - address 0x4000 800C) bit description

Bit	Symbol	Value	Description	Reset value
1:0	WLS		Word Length Select	0
		0x0	5-bit character length.	
		0x1	6-bit character length.	
		0x2	7-bit character length.	
		0x3	8-bit character length.	
2	SBS		Stop Bit Select	0
		0	1 stop bit.	
		1	2 stop bits (1.5 if LCR[1:0]=00).	
3	PE		Parity Enable	0
		0	Disable parity generation and checking.	
		1	Enable parity generation and checking.	
5:4	PST		Parity Select	0
		0x0	Odd parity. Number of 1s in the transmitted character and the attached parity bit will be odd.	
		0x1	Even Parity. Number of 1s in the transmitted character and the attached parity bit will be even.	
		0x2	Forced 1 stick parity.	
		0x3	Forced 0 stick parity.	
6	BC		Break Control	0
		0	Disable break transmission.	
		1	Enable break transmission. Output pin UART TXD is forced to logic 0 when LCR[6] is active high.	
7	DLAB		Divisor Latch Access Bit (DLAB)	0
		0	Disable access to Divisor Latches.	
		1	Enable access to Divisor Latches.	
31:8	-	-	Reserved	-

9.5.8 UART Modem Control Register

The MCR enables the modem loopback mode and controls the modem output signals.

Table 148. UART Modem Control Register (MCR - address 0x4000 8010) bit description

Bit	Symbol	Value	Description	Reset value
0	DTRCTRL		Source for modem output pin $\overline{DTR}$. This bit reads as 0 when modem loopback mode is active.	0
1	RTSCTRL		Source for modem output pin $\overline{RTS}$. This bit reads as 0 when modem loopback mode is active.	0
3:2	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
4	LMS		Loopback Mode Select. The modem loopback mode provides a mechanism to perform diagnostic loopback testing. Serial data from the transmitter is connected internally to serial input of the receiver. Input pin, RXD, has no effect on loopback and output pin, TXD is held in marking state. The $\overline{DSR}$, $\overline{CTS}$, $\overline{DCD}$, and $\overline{RI}$ pins are ignored. Externally, $\overline{DTR}$ and $\overline{RTS}$ are set inactive. Internally, the upper four bits of the MSR are driven by the lower four bits of the MCR. This permits modem status interrupts to be generated in loopback mode by writing the lower four bits of MCR.	0
		0	Disable modem loopback mode.	
		1	Enable modem loopback mode.	
5	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
6	RTSEN		RTS enable	0
		0	Disable auto-rts flow control.	
		1	Enable auto-rts flow control.	
7	CTSEN		CTS enable	0
		0	Disable auto-cts flow control.	
		1	Enable auto-cts flow control.	
31:8	-	-	Reserved	-

9.5.8.1 Auto-flow control

If auto-RTS mode is enabled the UART receiver FIFO hardware controls the $\overline{RTS}$ output of the UART. If the auto-CTS mode is enabled the UART TSR hardware will only start transmitting if the $\overline{CTS}$ input signal is asserted.

9.5.8.1.1 Auto-RTS

The auto-RTS function is enabled by setting the RTSen bit. Auto-RTS data flow control originates in the RBR module and is linked to the programmed receiver FIFO trigger level. If auto-RTS is enabled, the data-flow is controlled as follows:

When the receiver FIFO level reaches the programmed trigger level, $\overline{RTS}$ is deasserted (to a high value). It is possible that the sending UART sends an additional byte after the trigger level is reached (assuming the sending UART has another byte to send) because it might not recognize the deassertion of $\overline{RTS}$ until after it has begun sending the additional byte. $\overline{RTS}$ is automatically reasserted (to a low value) once the receiver FIFO has reached the previous trigger level. The reassertion of $\overline{RTS}$ signals to the sending UART to continue transmitting data.

If Auto-RTS mode is disabled, the RTSen bit controls the $\overline{RTS}$ output of the UART. If Auto-RTS mode is enabled, hardware controls the $\overline{RTS}$ output, and the actual value of $\overline{RTS}$ will be copied in the RTS Control bit of the UART. As long as Auto-RTS is enabled, the value of the RTS Control bit is read-only for software.

Example: Suppose the UART operating in type '550 mode has the trigger level in FCR set to 0x2, then, if Auto-RTS is enabled, the UART will deassert the $\overline{RTS}$ output as soon as the receive FIFO contains 8 bytes (Table 146 on page 145). The $\overline{RTS}$ output will be reasserted as soon as the receive FIFO hits the previous trigger level: 4 bytes.

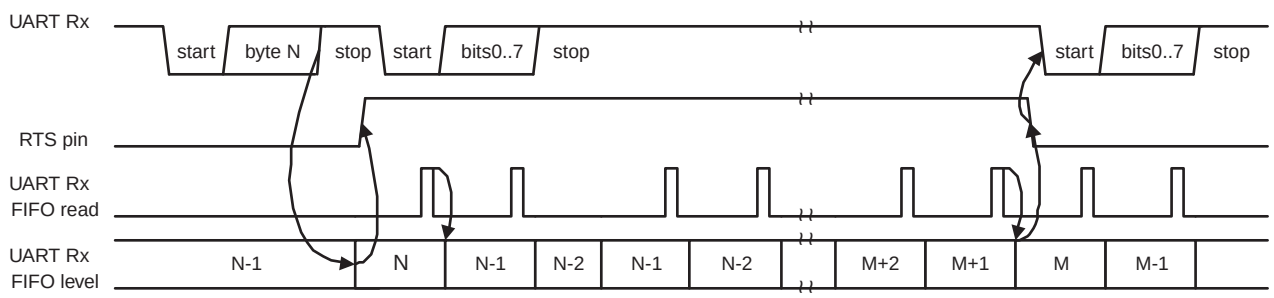


Fig 5. Auto-RTS Functional Timing

9.5.8.1.2 Auto-CTS

The Auto-CTS function is enabled by setting the CTSen bit. If Auto-CTS is enabled the transmitter circuitry in the TSR module checks $\overline{CTS}$ input before sending the next data byte. When $\overline{CTS}$ is active (low), the transmitter sends the next byte. To stop the transmitter from sending the following byte, $\overline{CTS}$ must be released before the middle of the last stop bit that is currently being sent. In Auto-CTS mode a change of the $\overline{CTS}$ signal does not trigger a modem status interrupt unless the CTS Interrupt Enable bit is set, Delta CTS bit in the MSR will be set though. Table 149 lists the conditions for generating a Modem Status interrupt.

Table 149. Modem status interrupt generation

Enable modem status interrupt (ER[3])	CTSen (MCR[7])	CTS interrupt enable (IER[7])	Delta CTS (MSR[0])	Delta DCD or trailing edge RI or Delta DSR (MSR[3] or MSR[2] or MSR[1])	Modem status interrupt
0	x	x	x	x	No
1	0	x	0	0	No
1	0	x	1	x	Yes
1	0	x	x	1	Yes
1	1	0	x	0	No
1	1	0	x	1	Yes
1	1	1	0	0	No
1	1	1	1	x	Yes
1	1	1	x	1	Yes

The auto-CTS function reduces interrupts to the host system. When flow control is enabled, a $\overline{CTS}$ state change does not trigger host interrupts because the device automatically controls its own transmitter. Without Auto-CTS, the transmitter sends any data present in the transmit FIFO and a receiver overrun error can result. Figure 6 illustrates the Auto-CTS functional timing.

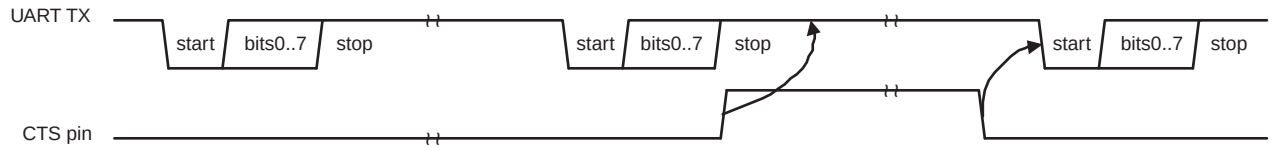


Fig 6. Auto-CTS Functional Timing

While starting transmission of the initial character the $\overline{CTS}$ signal is asserted. Transmission will stall as soon as the pending transmission has completed. The UART will continue transmitting a 1 bit as long as $\overline{CTS}$ is deasserted (high). As soon as $\overline{CTS}$ gets deasserted transmission resumes and a start bit is sent followed by the data bits of the next character.

9.5.9 UART Line Status Register

The LSR is a Read Only register that provides status information on the UART TX and RX blocks.

Table 150. UART Line Status Register (LSR - address 0x4000 8014, Read Only) bit description

Bit	Symbol	Value	Description	Reset Value
0	RDR		Receiver Data Ready:LSR[0] is set when the RBR holds an unread character and is cleared when the UART RBR FIFO is empty.	0
		0	RBR is empty.	
		1	RBR contains valid data.	
1	OE		Overrun Error. The overrun error condition is set as soon as it occurs. A LSR read clears LSR[1]. LSR[1] is set when UART RSR has a new character assembled and the UART RBR FIFO is full. In this case, the UART RBR FIFO will not be overwritten and the character in the UART RSR will be lost.	0
		0	Overrun error status is inactive.	
		1	Overrun error status is active.	
2	PE		Parity Error. When the parity bit of a received character is in the wrong state, a parity error occurs. A LSR read clears LSR[2]. Time of parity error detection is dependent on FCR[0]. **Note:** A parity error is associated with the character at the top of the UART RBR FIFO.	0
		0	Parity error status is inactive.	
		1	Parity error status is active.	

Table 150. UART Line Status Register (LSR - address 0x4000 8014, Read Only) bit description *…continued*

Bit	Symbol	Value	Description	Reset Value
3	FE		Framing Error. When the stop bit of a received character is a logic 0, a framing error occurs. A LSR read clears LSR[3]. The time of the framing error detection is dependent on FCR0. Upon detection of a framing error, the RX will attempt to re-synchronize to the data and assume that the bad stop bit is actually an early start bit. However, it cannot be assumed that the next received byte will be correct even if there is no Framing Error. **Note:** A framing error is associated with the character at the top of the UART RBR FIFO.	0
		0	Framing error status is inactive.	
		1	Framing error status is active.	
4	BI		Break Interrupt. When RXD1 is held in the spacing state (all zeros) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RXD1 goes to marking state (all ones). A LSR read clears this status bit. The time of break detection is dependent on FCR[0]. **Note:** The break interrupt is associated with the character at the top of the UART RBR FIFO.	0
		0	Break interrupt status is inactive.	
		1	Break interrupt status is active.	
5	THRE		Transmitter Holding Register Empty. THRE is set immediately upon detection of an empty UART THR and is cleared on a THR write.	1
		0	THR contains valid data.	
		1	THR is empty.	
6	TEMT		Transmitter Empty. TEMT is set when both THR and TSR are empty; TEMT is cleared when either the TSR or the THR contain valid data.	1
		0	THR and/or the TSR contains valid data.	
		1	THR and the TSR are empty.	
7	RXFE		Error in RX FIFO. LSR[7] is set when a character with a RX error such as framing error, parity error or break interrupt, is loaded into the RBR. This bit is cleared when the LSR register is read and there are no subsequent errors in the UART FIFO.	0
		0	RBR contains no UART RX errors or FCR[0]=0.	
		1	UART RBR contains at least one UART RX error.	
31:8	-	-	Reserved	-

9.5.10 UART Modem Status Register

The MSR is a read-only register that provides status information on the modem input signals. MSR[3:0] is cleared on MSR read. Note that modem signals have no direct effect on the UART operation. They facilitate the software implementation of modem signal operations.

Table 151: UART Modem Status Register (MSR - address 0x4000 8018) bit description

Bit	Symbol	Value	Description	Reset value
0	DCTS		Delta CTS. Set upon state change of input CTS. Cleared on an MSR read.	0
		0	No change detected on modem input, CTS.	
		1	State change detected on modem input, CTS.	
1	DDSR		Delta DSR. Set upon state change of input DSR. Cleared on an MSR read.	0
		0	No change detected on modem input, DSR.	
		1	State change detected on modem input, DSR.	
2	TERI		Trailing Edge RI. Set upon low to high transition of input RI. Cleared on an MSR read.	0
		0	No change detected on modem input, RI.	
		1	Low-to-high transition detected on RI.	
3	DDCD		Delta DCD. Set upon state change of input DCD. Cleared on an MSR read.	0
		0	No change detected on modem input, DCD.	
		1	State change detected on modem input, DCD.	
4	CTS		Clear To Send State. Complement of input signal CTS. This bit is connected to MCR[1] in modem loopback mode.	0
5	DSR		Data Set Ready State. Complement of input signal DSR. This bit is connected to MCR[0] in modem loopback mode.	0
6	RI		Ring Indicator State. Complement of input RI. This bit is connected to MCR[2] in modem loopback mode.	0
7	DCD		Data Carrier Detect State. Complement of input DCD. This bit is connected to MCR[3] in modem loopback mode.	0
31:8	-	-	Reserved, the value read from a reserved bit is not defined.	NA

9.5.11 UART Scratch Pad Register

The SCR has no effect on the UART operation. This register can be written and/or read at user's discretion. There is no provision in the interrupt interface that would indicate to the host that a read or write of the SCR has occurred.

Table 152. UART Scratch Pad Register (SCR - address 0x4000 801C) bit description

Bit	Symbol	Description	Reset value
7:0	PAD	A readable, writable byte.	0x00
31:8	-	Reserved	-

9.5.12 UART Auto-baud Control Register

The UART Auto-baud Control Register (ACR) controls the process of measuring the incoming clock/data rate for the baud rate generation and can be read and written at user's discretion.

Table 153. Auto-baud Control Register (ACR - address 0x4000 8020) bit description

Bit	Symbol	Value	Description	Reset value
0	START		Start bit. This bit is automatically cleared after auto-baud completion.	0
		0	Auto-baud stop (auto-baud is not running).	
		1	Auto-baud start (auto-baud is running). Auto-baud run bit. This bit is automatically cleared after auto-baud completion.	
1	MODE		Auto-baud mode select bit.	0
		0	Mode 0.	
		1	Mode 1.	
2	AUTORESTART		Start mode.	0
		0	No restart	
		1	Restart in case of time-out (counter restarts at next UART RX falling edge)	
7:3	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
8	ABEOINTCLR		End of auto-baud interrupt clear bit (write only accessible).	0
		0	Writing a 0 has no impact.	
		1	Writing a 1 will clear the corresponding interrupt in the IIR.	
9	ABTOINTCLR		Auto-baud time-out interrupt clear bit (write only accessible).	0
		0	Writing a 0 has no impact.	
		1	Writing a 1 will clear the corresponding interrupt in the IIR.	
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

9.5.12.1 Auto-baud

The UART auto-baud function can be used to measure the incoming baud-rate based on the ”AT" protocol (Hayes command). If enabled the auto-baud feature will measure the bit time of the receive data stream and set the divisor latch registers DLM and DLL accordingly.

Auto-baud is started by setting the ACR Start bit. Auto-baud can be stopped by clearing the ACR Start bit. The Start bit will clear once auto-baud has finished and reading the bit will return the status of auto-baud (pending/finished).

Two auto-baud measuring modes are available which can be selected by the ACR Mode bit. In mode 0 the baud-rate is measured on two subsequent falling edges of the UART RX pin (the falling edge of the start bit and the falling edge of the least significant bit). In mode 1 the baud-rate is measured between the falling edge and the subsequent rising edge of the UART RX pin (the length of the start bit).

The ACR AutoRestart bit can be used to automatically restart baud-rate measurement if a time-out occurs (the rate measurement counter overflows). If this bit is set the rate measurement will restart at the next falling edge of the UART RX pin.

The auto-baud function can generate two interrupts.

- The IIR ABTOInt interrupt will get set if the interrupt is enabled (IER ABToIntEn is set and the auto-baud rate measurement counter overflows).
- The IIR ABEOInt interrupt will get set if the interrupt is enabled (IER ABEOIntEn is set and the auto-baud has completed successfully).

The auto-baud interrupts have to be cleared by setting the corresponding ACR ABTOIntClr and ABEOIntEn bits.

Typically the fractional baud-rate generator is disabled (DIVADDVAL = 0) during auto-baud. However, if the fractional baud-rate generator is enabled (DIVADDVAL > 0), it is going to impact the measuring of UART RX pin baud-rate, but the value of the FDR register is not going to be modified after rate measurement. Also, when auto-baud is used, any write to DLM and DLL registers should be done before ACR register write. The minimum and the maximum baudrates supported by UART are function of UART_PCLK, number of data bits, stop bits and parity bits.

(2)

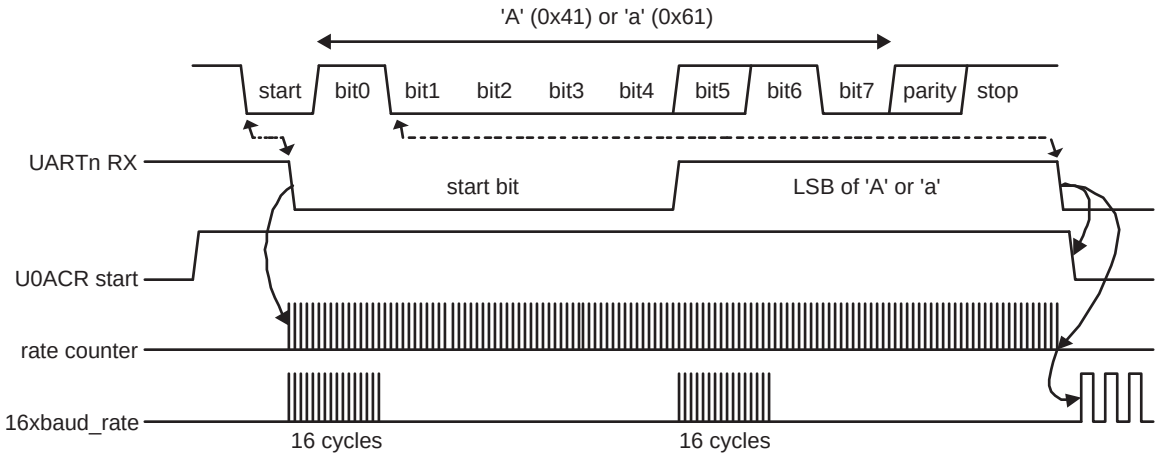
$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UART_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax$$

9.5.12.2 Auto-baud modes

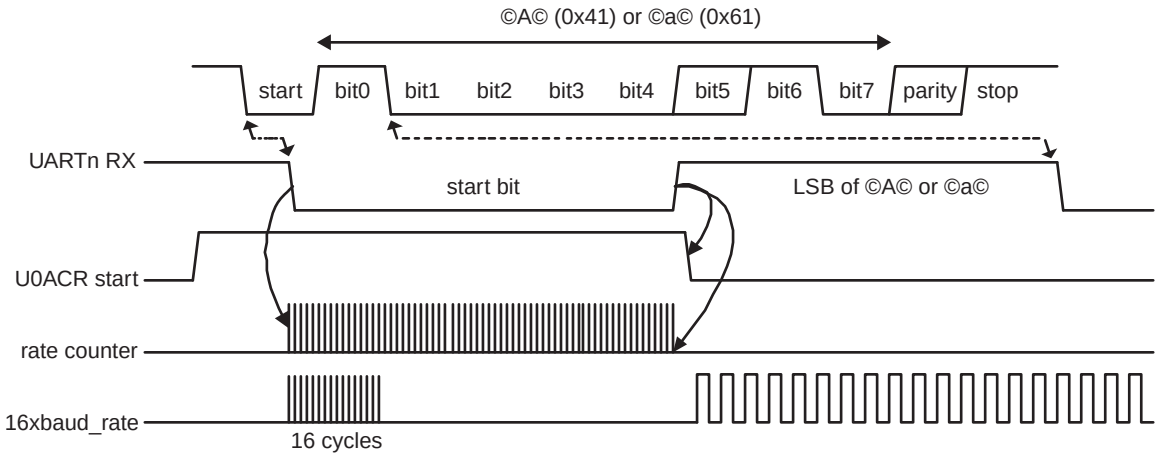
When the software is expecting an ”AT" command, it configures the UART with the expected character format and sets the ACR Start bit. The initial values in the divisor latches DLM and DLM don‘t care. Because of the ”A" or ”a" ASCII coding (”A" = 0x41, ”a" = 0x61), the UART RX pin sensed start bit and the LSB of the expected character are delimited by two falling edges. When the ACR Start bit is set, the auto-baud protocol will execute the following phases:

1. On ACR Start bit setting, the baud-rate measurement counter is reset and the UART RSR is reset. The RSR baud rate is switch to the highest rate.
2. A falling edge on UART RX pin triggers the beginning of the start bit. The rate measuring counter will start counting UART_PCLK cycles optionally pre-scaled by the fractional baud-rate generator.
3. During the receipt of the start bit, 16 pulses are generated on the RSR baud input with the frequency of the (fractional baud-rate pre-scaled) UART input clock, guaranteeing the start bit is stored in the RSR.
4. During the receipt of the start bit (and the character LSB for mode = 0) the rate counter will continue incrementing with the pre-scaled UART input clock (UART_PCLK).

5. If Mode = 0 then the rate counter will stop on next falling edge of the UART RX pin. If Mode = 1 then the rate counter will stop on the next rising edge of the UART RX pin.
6. The rate counter is loaded into DLM/DLL and the baud-rate will be switched to normal operation. After setting the DLM/DLL the end of auto-baud interrupt IIR ABEOInt will be set, if enabled. The RSR will now continue receiving the remaining bits of the ”A/a" character.



a. Mode 0 (start bit and LSB are used for auto-baud)



b. Mode 1 (only start bit is used for auto-baud)

Fig 7. Auto-baud a) mode 0 and b) mode 1 waveform

9.5.13 UART Fractional Divider Register

The UART Fractional Divider Register (FDR) controls the clock pre-scaler for the baud rate generation and can be read and written at the user’s discretion. This pre-scaler takes the APB clock and generates an output clock according to the specified fractional requirements.

Important: If the fractional divider is active (DIVADDVAL > 0) and DLM = 0, the value of the DLL register must be 3 or greater.

Table 154. UART Fractional Divider Register (FDR - address 0x4000 8028) bit description

Bit	Function	Description	Reset value
3:0	DIVADDVAL	Baud-rate generation pre-scaler divisor value. If this field is 0, fractional baud-rate generator will not impact the UARTn baudrate.	0
7:4	MULVAL	Baud-rate pre-scaler multiplier value. This field must be greater or equal 1 for UARTn to operate properly, regardless of whether the fractional baud-rate generator is used or not.	1
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

This register controls the clock pre-scaler for the baud rate generation. The reset value of the register keeps the fractional capabilities of UART disabled making sure that UART is fully software and hardware compatible with UARTs not equipped with this feature.

The UART baudrate can be calculated as (n = 1):

(3)

$$UART_{baudrate} = \frac{PCLK}{16 \times (256 \times DLM + DLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

Where UART_PCLK is the peripheral clock, DLM and DLL are the standard UART baud rate divider registers, and DIVADDVAL and MULVAL are UART fractional baudrate generator specific parameters.

The value of MULVAL and DIVADDVAL should comply to the following conditions:

1. $1 \leq MULVAL \leq 15$
2. $0 \leq DIVADDVAL \leq 14$
3. DIVADDVAL< MULVAL

The value of the FDR should not be modified while transmitting/receiving data or data may be lost or corrupted.

If the FDR register value does not comply to these two requests, then the fractional divider output is undefined. If DIVADDVAL is zero then the fractional divider is disabled, and the clock will not be divided.

9.5.13.1 Baudrate calculation

UART can operate with or without using the Fractional Divider. In real-life applications it is likely that the desired baudrate can be achieved using several different Fractional Divider settings. The following algorithm illustrates one way of finding a set of DLM, DLL, MULVAL, and DIVADDVAL values. Such set of parameters yields a baudrate with a relative error of less than 1.1% from the desired one.

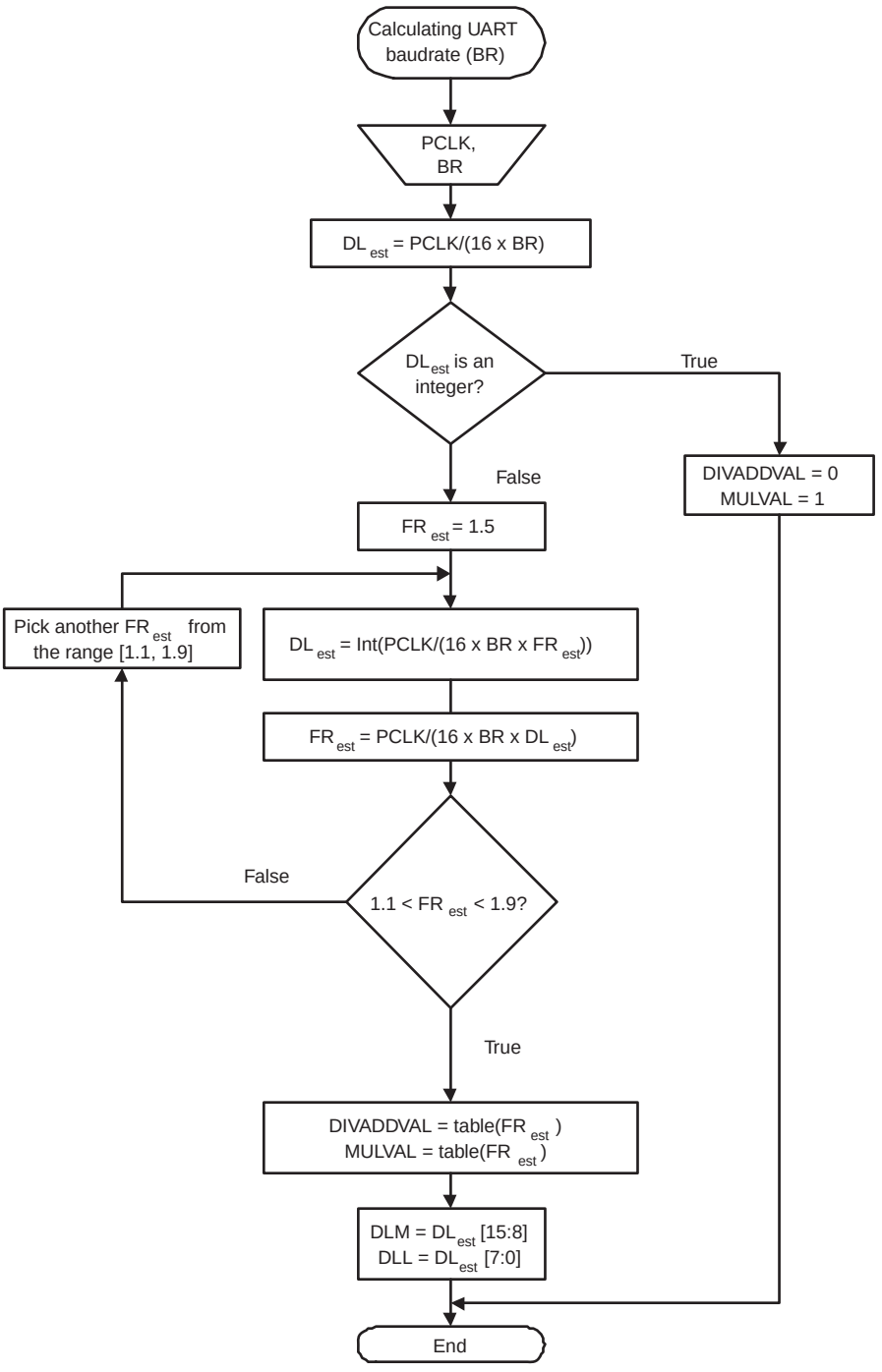


Fig 8. Algorithm for setting UART dividers

Table 155. Fractional Divider setting look-up table

FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal
1.000	0/1	1.250	1/4	1.500	1/2	1.750	3/4
1.067	1/15	1.267	4/15	1.533	8/15	1.769	10/13
1.071	1/14	1.273	3/11	1.538	7/13	1.778	7/9
1.077	1/13	1.286	2/7	1.545	6/11	1.786	11/14
1.083	1/12	1.300	3/10	1.556	5/9	1.800	4/5
1.091	1/11	1.308	4/13	1.571	4/7	1.818	9/11
1.100	1/10	1.333	1/3	1.583	7/12	1.833	5/6
1.111	1/9	1.357	5/14	1.600	3/5	1.846	11/13
1.125	1/8	1.364	4/11	1.615	8/13	1.857	6/7
1.133	2/15	1.375	3/8	1.625	5/8	1.867	13/15
1.143	1/7	1.385	5/13	1.636	7/11	1.875	7/8
1.154	2/13	1.400	2/5	1.643	9/14	1.889	8/9
1.167	1/6	1.417	5/12	1.667	2/3	1.900	9/10
1.182	2/11	1.429	3/7	1.692	9/13	1.909	10/11
1.200	1/5	1.444	4/9	1.700	7/10	1.917	11/12
1.214	3/14	1.455	5/11	1.714	5/7	1.923	12/13
1.222	2/9	1.462	6/13	1.727	8/11	1.929	13/14
1.231	3/13	1.467	7/15	1.733	11/15	1.933	14/15

9.5.13.1.1 Example 1: UART_PCLK = 14.7456 MHz, BR = 9600

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 14.7456 MHz / (16 x 9600) = 96. Since this DL_{est} is an integer number, DIVADDVAL = 0, MULVAL = 1, DLM = 0, and DLL = 96.

9.5.13.1.2 Example 2: UART_PCLK = 12 MHz, BR = 115200

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 12 MHz / (16 x 115200) = 6.51. This DL_{est} is not an integer number and the next step is to estimate the FR parameter. Using an initial estimate of FR_{est} = 1.5 a new DL_{est} = 4 is calculated and FR_{est} is recalculated as FR_{est} = 1.628. Since FRest = 1.628 is within the specified range of 1.1 and 1.9, DIVADDVAL and MULVAL values can be obtained from the attached look-up table.

The closest value for FRest = 1.628 in the look-up Table 155 is FR = 1.625. It is equivalent to DIVADDVAL = 5 and MULVAL = 8.

Based on these findings, the suggested UART setup would be: DLM = 0, DLL = 4, DIVADDVAL = 5, and MULVAL = 8. According to Equation 3 UART's is 115384. This rate has a relative error of 0.16% from the originally specified 115200.

9.5.14 UART Transmit Enable Register (TER - 0x4000 8030)

In addition to being equipped with full hardware flow control (auto-cts and auto-rts mechanisms described above), TER enables implementation of software flow control, too. When TXEn =1, UART transmitter will keep sending data as long as they are available. As soon as TXEn becomes 0, UART transmission will stop.

Although Table 156 describes how to use TXEn bit in order to achieve hardware flow control, it is strongly suggested to let UART hardware implemented auto flow control features take care of this, and limit the scope of TXEn to software flow control.

TER enables implementation of software and hardware flow control. When TXEn =1, UART transmitter will keep sending data as long as they are available. As soon as TXEn becomes 0, UART transmission will stop.

Table 156 describes how to use TXEn bit in order to achieve software flow control.

Table 156. UART Transmit Enable Register (TER - address 0x4000 8030) bit description

Bit	Symbol	Description	Reset value
6:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	TXEN	When this bit is 1, as it is after a Reset, data written to the THR is output on the TXD pin as soon as any preceding data has been sent. If this bit cleared to 0 while a character is being sent, the transmission of that character is completed, but no further characters are sent until this bit is set again. In other words, a 0 in this bit blocks the transfer of characters from the THR or TX FIFO into the transmit shift register. Software can clear this bit when it detects that the a hardware-handshaking TX-permit signal ($\overline{CTS}$) has gone false, or with software handshaking, when it receives an XOFF character (DC3). Software can set this bit again when it detects that the TX-permit signal has gone true, or when it receives an XON (DC1) character.	1
31:8	-	Reserved	-

9.5.15 UART RS485 Control register

The RS485CTRL register controls the configuration of the UART in RS-485/EIA-485 mode.

Table 157. UART RS485 Control register (RS485CTRL - address 0x4000 804C) bit description

Bit	Symbol	Value	Description	Reset value
0	NMMEN		NMM enable.	0
		0	RS-485/EIA-485 Normal Multidrop Mode (NMM) is disabled.	
		1	RS-485/EIA-485 Normal Multidrop Mode (NMM) is enabled. In this mode, an address is detected when a received byte causes the UART to set the parity error and generate an interrupt.	
1	RXDIS		Receiver enable.	0
		0	The receiver is enabled.	
		1	The receiver is disabled.	
2	AADEN		AAD enable.	0
		0	Auto Address Detect (AAD) is disabled.	
		1	Auto Address Detect (AAD) is enabled.	

Table 157. UART RS485 Control register (RS485CTRL - address 0x4000 804C) bit description
…continued

Bit	Symbol	Value	Description	Reset value
3	SEL		Select direction control pin	0
		0	If direction control is enabled (bit DCTRL = 1), pin RTS is used for direction control.	
		1	If direction control is enabled (bit DCTRL = 1), pin DTR is used for direction control.	
4	DCTRL		Auto direction control enable.	0
		0	Disable Auto Direction Control.	
		1	Enable Auto Direction Control.	
5	OINV		Polarity control. This bit reverses the polarity of the direction control signal on the RTS (or DTR) pin.	0
		0	The direction control pin will be driven to logic 0 when the transmitter has data to be sent. It will be driven to logic 1 after the last bit of data has been transmitted.	
		1	The direction control pin will be driven to logic 1 when the transmitter has data to be sent. It will be driven to logic 0 after the last bit of data has been transmitted.	
31:6	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

9.5.16 UART RS-485 Address Match register

The RS485ADRMATCH register contains the address match value for RS-485/EIA-485 mode.

Table 158. UART RS-485 Address Match register (RS485ADRMATCH - address 0x4000 8050) bit description

Bit	Symbol	Description	Reset value
7:0	ADRMATCH	Contains the address match value.	0x00
31:8	-	Reserved	-

9.5.17 UART1 RS-485 Delay value register

The user may program the 8-bit RS485DLY register with a delay between the last stop bit leaving the TXFIFO and the de-assertion of RTS (or DTR). This delay time is in periods of the baud clock. Any delay time from 0 to 255 bit times may be programmed.

Table 159. UART RS-485 Delay value register (RS485DLY - address 0x4000 8054) bit description

Bit	Symbol	Description	Reset value
7:0	DLY	Contains the direction control (RTS or DTR) delay value. This register works in conjunction with an 8-bit counter.	0x00
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

9.5.18 RS-485/EIA-485 modes of operation

The RS-485/EIA-485 feature allows the UART to be configured as an addressable slave. The addressable slave is one of multiple slaves controlled by a single master.

The UART master transmitter will identify an address character by setting the parity (9th) bit to ‘1’. For data characters, the parity bit is set to ‘0’.

Each UART slave receiver can be assigned a unique address. The slave can be programmed to either manually or automatically reject data following an address which is not theirs.

9.5.18.1 RS-485/EIA-485 Normal Multidrop Mode (NMM)

Setting the RS485CTRL bit 0 enables this mode. In this mode, an address is detected when a received byte causes the UART to set the parity error and generate an interrupt.

If the receiver is disabled (RS485CTRL bit 1 = ‘1’), any received data bytes will be ignored and will not be stored in the RXFIFO. When an address byte is detected (parity bit = ‘1’) it will be placed into the RXFIFO and an RX Data Ready Interrupt will be generated. The processor can then read the address byte and decide whether or not to enable the receiver to accept the following data.

While the receiver is enabled (RS485CTRL bit 1 =’0’), all received bytes will be accepted and stored in the RXFIFO regardless of whether they are data or address. When an address character is received a parity error interrupt will be generated and the processor can decide whether or not to disable the receiver.

9.5.18.2 RS-485/EIA-485 Auto Address Detection (AAD) mode

When both RS485CTRL register bits 0 (9-bit mode enable) and 2 (AAD mode enable) are set, the UART is in auto address detect mode.

In this mode, the receiver will compare any address byte received (parity = ‘1’) to the 8-bit value programmed into the RS485ADRMATCH register.

If the receiver is disabled (RS485CTRL bit 1 = ‘1’), any received byte will be discarded if it is either a data byte OR an address byte which fails to match the RS485ADRMATCH value.

When a matching address character is detected it will be pushed onto the RXFIFO along with the parity bit, and the receiver will be automatically enabled (RS485CTRL bit 1 will be cleared by hardware). The receiver will also generate n RX Data Ready Interrupt.

While the receiver is enabled (RS485CTRL bit 1 = ‘0’), all bytes received will be accepted and stored in the RXFIFO until an address byte which does not match the RS485ADRMATCH value is received. When this occurs, the receiver will be automatically disabled in hardware (RS485CTRL bit 1 will be set), The received non-matching address character will not be stored in the RXFIFO.

9.5.18.3 RS-485/EIA-485 Auto Direction Control

RS485/EIA-485 mode includes the option of allowing the transmitter to automatically control the state of the DIR pin as a direction control output signal.

Setting RS485CTRL bit 4 = ‘1’ enables this feature.

Direction control, if enabled, will use the $\overline{RTS}$ pin when RS485CTRL bit 3 = '0'. It will use the $\overline{DTR}$ pin when RS485CTRL bit 3 = '1'.

When Auto Direction Control is enabled, the selected pin will be asserted (driven LOW) when the CPU writes data into the TXFIFO. The pin will be de-asserted (driven HIGH) once the last bit of data has been transmitted. See bits 4 and 5 in the RS485CTRL register.

The RS485CTRL bit 4 takes precedence over all other mechanisms controlling the direction control pin with the exception of loopback mode.

9.5.18.4 RS485/EIA-485 driver delay time

The driver delay time is the delay between the last stop bit leaving the TXFIFO and the de-assertion of $\overline{RTS}$. This delay time can be programmed in the 8-bit RS485DLY register. The delay time is in periods of the baud clock. Any delay time from 0 to 255 bit times may be used.

9.5.18.5 RS485/EIA-485 output inversion

The polarity of the direction control signal on the $\overline{RTS}$ (or $\overline{DTR}$) pins can be reversed by programming bit 5 in the RS485CTRL register. When this bit is set, the direction control pin will be driven to logic 1 when the transmitter has data waiting to be sent. The direction control pin will be driven to logic 0 after the last bit of data has been transmitted.

9.5.19 UART FIFO Level register

FIFOLVL register is a Read Only register that allows software to read the current FIFO level status. Both the transmit and receive FIFO levels are present in this register.

Table 160. UART FIFO Level register (FIFOLVL - address 0x4000 8058, Read Only) bit description

Bit	Symbol	Description	Reset value
3:0	RXFIFILVL	Reflects the current level of the UART receiver FIFO. 0 = empty, 0xF = FIFO full.	0x00
7:4	-	Reserved. The value read from a reserved bit is not defined.	NA
11:8	TXFIFOLVL	Reflects the current level of the UART transmitter FIFO. 0 = empty, 0xF = FIFO full.	0x00
31:12	-	Reserved. The value read from a reserved bit is not defined.	NA

9.6 Architecture

The architecture of the UART is shown below in the block diagram.

The APB interface provides a communications link between the CPU or host and the UART.

The UART receiver block, RX, monitors the serial input line, RXD, for valid input. The UART RX Shift Register (RSR) accepts valid characters via RXD. After a valid character is assembled in the RSR, it is passed to the UART RX Buffer Register FIFO to await access by the CPU or host via the generic host interface.

The UART transmitter block, TX, accepts data written by the CPU or host and buffers the data in the UART TX Holding Register FIFO (THR). The UART TX Shift Register (TSR) reads the data stored in the THR and assembles the data to transmit via the serial output pin, TXD1.

The UART Baud Rate Generator block, BRG, generates the timing enables used by the UART TX block. The BRG clock input source is UART_PCLK. The main clock is divided down per the divisor specified in the DLL and DLM registers. This divided down clock is a 16x oversample clock, NBAUDOUT.

The interrupt interface contains registers IER and IIR. The interrupt interface receives several one clock wide enables from the TX and RX blocks.

Status information from the TX and RX is stored in the LSR. Control information for the TX and RX is stored in the LCR.

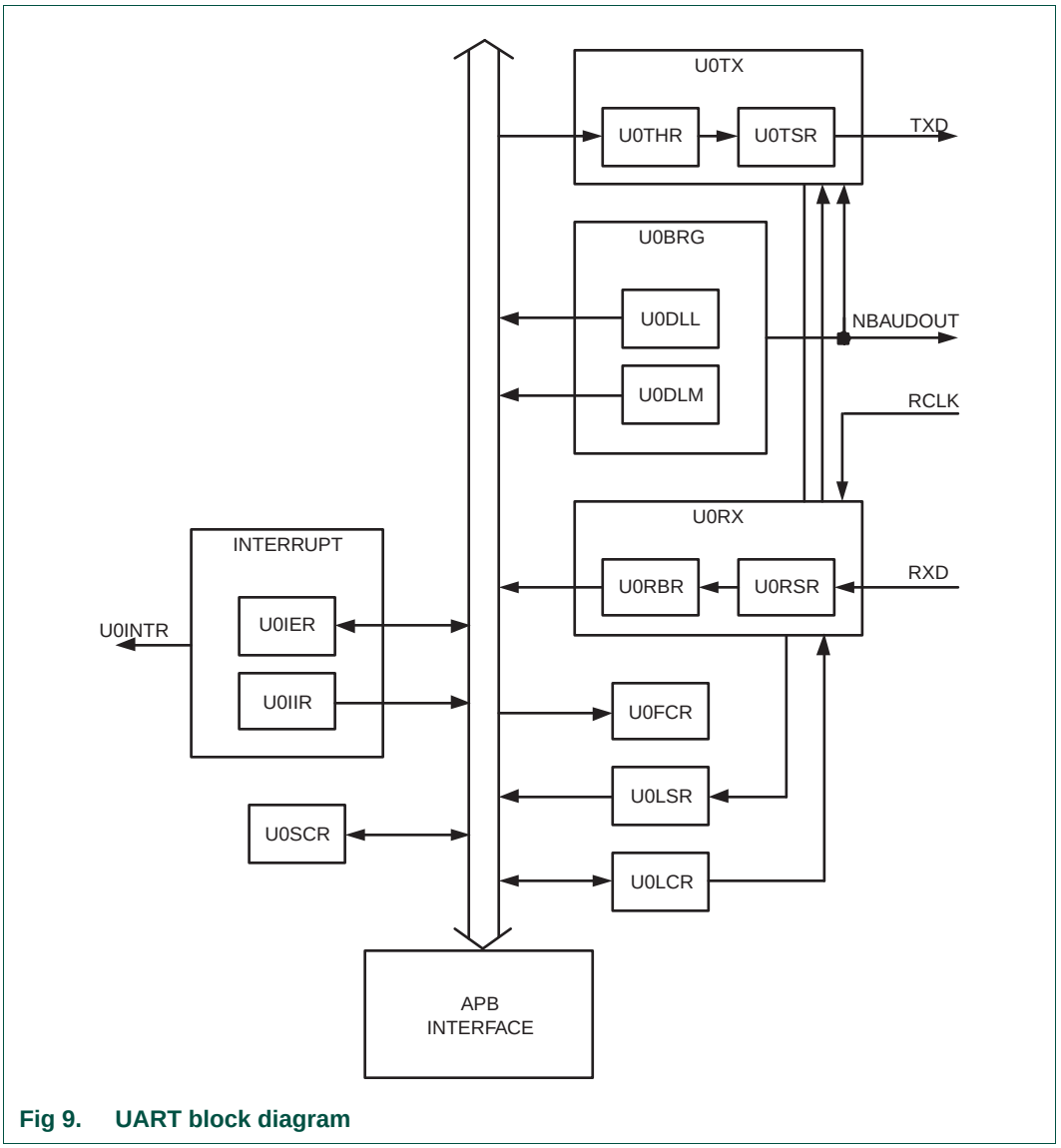


Fig 9. UART block diagram

10.1 How to read this chapter

The UART1 is available on all LPC122x parts.

10.2 Basic configuration

Clocks and power to the UART1 block are controlled by:

1. The SYSAHBCLKCTRL register (see Table 21).
2. The UART1_PCLK which is enabled in the UART1 clock divider register (see Table 24). This clock is used by the UART baud rate generator.

Remark: The UART1 pins must be configured in the corresponding IOCON registers **before** the UART1 clocks are enabled.

The UART1_PCLK can be disabled in the UART1CLKDIV register (see Table 24) and the UART block can be disabled through the System AHB clock control register bit 13 (see Table 21) for power savings.

10.3 Features

- 16-byte receive and transmit FIFOs.
- Register locations conform to ‘550 industry standard.
- Receiver FIFO trigger points at 1, 4, 8, and 14 byte.
- Built-in baud rate generator.
- UART allows for implementation of either software or hardware flow control.
- IrDA mode to support infrared communication.

10.4 Pin description

Table 161. UART1 pin description

Pin	Type	Description
RXD1	Input	**Serial Input.** Serial receive data.
TXD1	Output	**Serial Output.** Serial transmit data.

10.5 Register description

The UART contains registers organized as shown in Table 162. The Divisor Latch Access Bit (DLAB) is contained in LCR[7] and enables access to the Divisor Latches.

Table 162. Register overview: UART0 (base address: 0x4000 C000)

Name	Access	Address offset	Description	Reset value[1]	Notes
RBR	RO	0x000	Receiver Buffer Register. Contains the next received character to be read.	NA	when DLAB=0
THR	WO	0x000	Transmit Holding Register. The next character to be transmitted is written here.	NA	when DLAB=0
DLL	R/W	0x000	Divisor Latch LSB. Least significant byte of the baud rate divisor value. The full divisor is used to generate a baud rate from the fractional rate divider.	0x01	when DLAB=1
DLM	R/W	0x004	Divisor Latch MSB. Most significant byte of the baud rate divisor value. The full divisor is used to generate a baud rate from the fractional rate divider.	0x00	when DLAB=1
IER	R/W	0x004	Interrupt Enable Register. Contains individual interrupt enable bits for the 7 potential UART interrupts.	0x00	when DLAB=0
IIR	RO	0x008	Interrupt ID Register. Identifies which interrupt(s) are pending.	0x01	-
FCR	WO	0x008	FIFO Control Register. Controls UART FIFO usage and modes.	0x00	-
LCR	R/W	0x00C	Line Control Register. Contains controls for frame formatting and break generation.	0x00	-
-	-	0x010	Reserved	0x00	-
LSR	RO	0x014	Line Status Register. Contains flags for transmit and receive status, including line errors.	0x60	-
-	-	0x018	Reserved	0x00	-
SCR	R/W	0x01C	Scratch Pad Register. Eight-bit temporary storage for software.	0x00	-
ACR	R/W	0x020	Auto-baud Control Register. Contains controls for the auto-baud feature.	0x00	-
ICR	R/W	0x024	IrDA Control Register. Enables and configures the IrDA mode.	0x00	-
FDR	R/W	0x028	Fractional Divider Register. Generates a clock input for the baud rate divider.	0x10	-
-	-	0x02C	Reserved	-	-
TER	R/W	0x030	Transmit Enable Register. Turns off UART transmitter for use with software flow control.	0x80	-
-	-	0x034 - 0x054	Reserved	-	-
FIFOLVL	RO	0x058	FIFO Level register. Provides the current fill levels of the transmit and receive FIFOs.	0x00	-

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

10.5.1 UART Receiver Buffer Register (when DLAB = 0, Read Only)

The RBR is the top byte of the UART RX FIFO. The top byte of the RX FIFO contains the oldest character received and can be read via the bus interface. The LSB (bit 0) represents the “oldest” received data bit. If the character received is less than 8 bits, the unused MSBs are padded with zeroes.

The Divisor Latch Access Bit (DLAB) in LCR must be zero in order to access the RBR. The RBR is always Read Only.

Since PE, FE and BI bits (see Table 172) correspond to the byte sitting on the top of the RBR FIFO (i.e. the one that will be read in the next read from the RBR), the right approach for fetching the valid pair of received byte and its status bits is first to read the content of the LSR register, and then to read a byte from the RBR.

Table 163. UART Receiver Buffer Register (RBR - address 0x4000 C000 when DLAB = 0, Read Only) bit description

Bit	Symbol	Description	Reset value
7:0	RBR	The UART Receiver Buffer Register contains the oldest received byte in the UART RX FIFO.	undefined
31:8	-	Reserved	-

10.5.2 UART Transmitter Holding Register (when DLAB = 0, Write Only)

The THR is the top byte of the UART TX FIFO. The top byte is the newest character in the TX FIFO and can be written via the bus interface. The LSB represents the first bit to transmit.

The Divisor Latch Access Bit (DLAB) in LCR must be zero in order to access the THR. The THR is always Write Only.

Table 164. UART Transmitter Holding Register (THR - address 0x4000 C000 when DLAB = 0, Write Only) bit description

Bit	Symbol	Description	Reset value
7:0	THR	Writing to the UART Transmit Holding Register causes the data to be stored in the UART transmit FIFO. The byte will be sent when it reaches the bottom of the FIFO and the transmitter is available.	NA
31:8	-	Reserved	-

10.5.3 UART Divisor Latch LSB and MSB Registers (when DLAB = 1)

The UART Divisor Latch is part of the UART Baud Rate Generator and holds the value used, along with the Fractional Divider, to divide the UART_PCLK clock in order to produce the baud rate clock, which must be 16x the desired baud rate. The DLL and DLM registers together form a 16-bit divisor where DLL contains the lower 8 bits of the divisor and DLM contains the higher 8 bits of the divisor. A 0x0000 value is treated like a 0x0001 value as division by zero is not allowed.The Divisor Latch Access Bit (DLAB) in LCR must be one in order to access the UART Divisor Latches. Details on how to select the right value for DLL and DLM can be found in Section 10.5.12.

Table 165. UART Divisor Latch LSB Register (DLL - address 0x4000 C000 when DLAB = 1) bit description

Bit	Symbol	Description	Reset value
7:0	DLLSB	The UART Divisor Latch LSB Register, along with the DLM register, determines the baud rate of the UART.	0x01
31:8	-	Reserved	-

Table 166. UART Divisor Latch MSB Register (DLM - address 0x4000 C004 when DLAB = 1) bit description

Bit	Symbol	Description	Reset value
7:0	DLMSB	The UART Divisor Latch MSB Register, along with the DLL register, determines the baud rate of the UART.	0x00
31:8	-	Reserved	-

10.5.4 UART Interrupt Enable Register (when DLAB = 0)

The IER is used to enable the four UART interrupt sources.

Table 167. UART Interrupt Enable Register (IER - address 0x4000 C004 when DLAB = 0) bit description

Bit	Symbol	Value	Description	Reset value
0	RBRIE		RBR Interrupt Enable. Enables the Receive Data Available interrupt for UART. It also controls the Character Receive Time-out interrupt.	0
		0	Disable the RDA interrupts.	
		1	Enable the RDA interrupts.	
1	THREIE		THRE Interrupt Enable. Enables the THRE interrupt for UART. The status of this interrupt can be read from LSR[5].	0
		0	Disable the THRE interrupts.	
		1	Enable the THRE interrupts.	
2	RXIE		RX Line Interrupt Enable. Enables the UART RX line status interrupts. The status of this interrupt can be read from LSR[4:1].	0
		0	Disable the RX line status interrupts.	
		1	Enable the RX line status interrupts.	
3	-	-	Reserved	-
6:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	-	-	Reserved	0
8	ABEOINTEN		Enables the end of auto-baud interrupt.	0
		0	Disable end of auto-baud Interrupt.	
		1	Enable end of auto-baud Interrupt.	

Table 167. UART Interrupt Enable Register (IER - address 0x4000 C004 when DLAB = 0) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
9	ABTOINTEN		Enables the auto-baud time-out interrupt.	0
		0	Disable auto-baud time-out Interrupt.	
		1	Enable auto-baud time-out Interrupt.	
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

10.5.5 UART Interrupt Identification Register

The IIR provides a status code that denotes the priority and source of a pending interrupt. The interrupts are frozen during an IIR access. If an interrupt occurs during an IIR access, the interrupt is recorded for the next IIR access.

Table 168. UART Interrupt Identification Register (IIR - address 0x4004 C008, Read Only) bit description

Bit	Symbol	Value	Description	Reset value
0	INTSTATUS		Interrupt status. Note that IIR[0] is active low. The pending interrupt can be determined by evaluating IIR[3:1].	1
		0	At least one interrupt is pending.	
		1	No interrupt is pending.	
3:1	INTID		Interrupt identification. IER[3:1] identifies an interrupt corresponding to the UART RX FIFO. All other combinations of IER[3:1] not listed below are reserved (000,100,101,111).	0
		0x3	1 - Receive Line Status (RLS).	
		0x2	2a - Receive Data Available (RDA).	
		0x6	2b - Character Time-out Indicator (CTI).	
		0x1	3 - THRE Interrupt.	
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7:6	FIFOEN		FIFO enable. These bits are equivalent to FCR[0].	0
8	ABEOINT		End of auto-baud interrupt. True if auto-baud has finished successfully and interrupt is enabled.	0
9	ABTOINT		Auto-baud time-out interrupt. True if auto-baud has timed out and interrupt is enabled.	0
31:10	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Bit IIR[9:8] are set by the auto-baud function and signal a time-out or end of auto-baud condition. The auto-baud interrupt conditions are cleared by setting the corresponding Clear bits in the Auto-baud Control Register.

If the IntStatus bit is 1 no interrupt is pending and the IntId bits will be zero. If the IntStatus is 0, a non auto-baud interrupt is pending in which case the IntId bits identify the type of interrupt and handling as described in Table 169. Given the status of IIR[3:0], an interrupt

handler routine can determine the cause of the interrupt and how to clear the active interrupt. The IIR must be read in order to clear the interrupt prior to exiting the Interrupt Service Routine.

The UART RLS interrupt (IIR[3:1] = 011) is the highest priority interrupt and is set whenever any one of four error conditions occur on the UART RX input: overrun error (OE), parity error (PE), framing error (FE) and break interrupt (BI). The UART RX error condition that set the interrupt can be observed via LSR[4:1]. The interrupt is cleared upon an LSR read.

The UART RDA interrupt (IIR[3:1] = 010) shares the second level priority with the CTI interrupt (IIR[3:1] = 110). The RDA is activated when the UART RX FIFO reaches the trigger level defined in FCR7:6 and is reset when the UART RX FIFO depth falls below the trigger level. When the RDA interrupt goes active, the CPU can read a block of data defined by the trigger level.

The CTI interrupt (IIR[3:1] = 110) is a second level interrupt and is set when the UART RX FIFO contains at least one character and no UART RX FIFO activity has occurred in 3.5 to 4.5 character times. Any UART RX FIFO activity (read or write of UART RSR) will clear the interrupt. This interrupt is intended to flush the UART RBR after a message has been received that is not a multiple of the trigger level size. For example, if a peripheral wished to send a 105 character message and the trigger level was 10 characters, the CPU would receive 10 RDA interrupts resulting in the transfer of 100 characters and 1 to 5 CTI interrupts (depending on the service routine) resulting in the transfer of the remaining 5 characters.

Table 169. UART Interrupt Handling

IIR[3:0] value[1]	Priority	Interrupt type	Interrupt source	Interrupt reset
0001	-	None	None	-
0110	Highest	RX Line Status / Error	OE[2] or PE[2] or FE[2] or BI[2]	LSR Read[2]
0100	Second	RX Data Available	RX data available or trigger level reached in FIFO (FCR0=1)	RBR Read[3] or UART FIFO drops below trigger level
1100	Second	Character Time-out indication	Minimum of one character in the RX FIFO and no character input or removed during a time period depending on how many characters are in FIFO and what the trigger level is set at (3.5 to 4.5 character times). The exact time will be: [(word length) × 7 - 2] × 8 + [(trigger level - number of characters) × 8 + 1] RCLKs	RBR Read[3]
0010	Third	THRE	THRE[2]	IIR Read[4] (if source of interrupt) or THR write

[1] Values "0000", “0011”, “0101”, “0111”, “1000”, “1001”, “1010”, “1011”,”1101”,”1110”,”1111” are reserved.

[2] For details see Section 10.5.8 “UART Line Status Register”

[3] For details see Section 10.5.1 “UART Receiver Buffer Register (when DLAB = 0, Read Only)”

[4] For details see Section 10.5.5 “UART Interrupt Identification Register” and Section 10.5.2 “UART Transmitter Holding Register (when DLAB = 0, Write Only)”

The UART THRE interrupt (IIR[3:1] = 001) is a third level interrupt and is activated when the UART THR FIFO is empty provided certain initialization conditions have been met. These initialization conditions are intended to give the UART THR FIFO a chance to fill up with data to eliminate many THRE interrupts from occurring at system start-up. The initialization conditions implement a one character delay minus the stop bit whenever THRE = 1 and there have not been at least two characters in the THR at one time since the last THRE = 1 event. This delay is provided to give the CPU time to write data to THR without a THRE interrupt to decode and service. A THRE interrupt is set immediately if the UART THR FIFO has held two or more characters at one time and currently, the THR is empty. The THRE interrupt is reset when a THR write occurs or a read of the IIR occurs and the THRE is the highest interrupt (IIR[3:1] = 001).

10.5.6 UART FIFO Control Register

The FCR controls the operation of the UART RX and TX FIFOs.

Table 170. UART FIFO Control Register (FCR - address 0x4000 C008, Write Only) bit description

Bit	Symbol	Value	Description	Reset value
0	FIFOEN		FIFO Enable	0
		0	UART FIFOs are disabled. Must not be used in the application.	
		1	Active high enable for both UART RX and TX FIFOs and FCR[7:1] access. This bit must be set for proper UART operation. Any transition on this bit will automatically clear the UART FIFOs.	
1	RXFIFORES		RX FIFO Reset	0
		0	No impact on either of UART FIFOs.	
		1	Writing a logic 1 to FCR[1] will clear all bytes in UART RX FIFO, reset the pointer logic. This bit is self-clearing.	
2	TXFIFORES		TX FIFO Reset	0
		0	No impact on either of UART FIFOs.	
		1	Writing a logic 1 to FCR[2] will clear all bytes in UART TX FIFO, reset the pointer logic. This bit is self-clearing.	
3	DMAMODE		DMA Mode Select. When the FIFO enable bit (bit 0 of this register) is set, this bit selects the DMA mode. See Section 10.5.6.1.	0
		0	DMA not used.	
		1	DMA mode enabled.	
5:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 170. UART FIFO Control Register (FCR - address 0x4000 C008, Write Only) bit description

Bit	Symbol	Value	Description	Reset value
7:6	RXTL		RX Trigger Level. These two bits determine how many receiver UART FIFO characters must be written before an interrupt is activated.	0
		0x0	Trigger level 0 (1 character or 0x01).	
		0x1	Trigger level 1 (4 characters or 0x04).	
		0x2	Trigger level 2 (8 characters or 0x08).	
		0x3	Trigger level 3 (14 characters or 0x0E).	
31:8	-	-	Reserved	-

10.5.6.1 DMA operation

The user can optionally operate the UART transmit and/or receive using the micro DMA. The DMA mode is determined by the DMA Mode Select bit in the FCR register. This bit only has an effect when the FIFOs are enabled via the FIFO Enable bit in the FCR register.

10.5.6.1.1 UART receiver DMA

In DMA mode, the receiver DMA request is asserted when the receiver FIFO level is equal to or greater than trigger level, or if a character time-out occurs. See the description of the RX Trigger Level above. The receiver DMA request is cleared by the DMA controller.

10.5.6.1.2 UART transmitter DMA

In DMA mode, the transmitter DMA request is asserted when the transmitter FIFO transitions to not full. The transmitter DMA request is cleared by the DMA controller.

10.5.7 UART Line Control Register

The LCR determines the format of the data character that is to be transmitted or received.

Table 171. UART Line Control Register (LCR - address 0x4000 C00C) bit description

Bit	Symbol	Value	Description	Reset value
1:0	WLS		Word Length Select	0
		0x0	5-bit character length.	
		0x1	6-bit character length.	
		0x2	7-bit character length.	
		0x3	8-bit character length.	
2	SBS		Stop Bit Select	0
		0	1 stop bit.	
		1	2 stop bits (1.5 if LCR[1:0]=00).	
3	PE		Parity Enable	0
		0	Disable parity generation and checking.	
		1	Enable parity generation and checking.	

Table 171. UART Line Control Register (LCR - address 0x4000 C00C) bit description

Bit	Symbol	Value	Description	Reset value
5:4	PS		Parity Select	0
		0x0	Odd parity. Number of 1s in the transmitted character and the attached parity bit will be odd.	
		0x1	Even Parity. Number of 1s in the transmitted character and the attached parity bit will be even.	
		0x2	Forced 1 stick parity.	
		0x3	Forced 0 stick parity.	
6	BC		Break Control	0
		0	Disable break transmission.	
		1	Enable break transmission. Output pin UART TXD is forced to logic 0 when LCR[6] is active high.	
7	DLAB		Divisor Latch Access Bit (DLAB)	0
		0	Disable access to Divisor Latches.	
		1	Enable access to Divisor Latches.	
31:8	-	-	Reserved	-

10.5.8 UART Line Status Register

The LSR is a Read Only register that provides status information on the UART TX and RX blocks.

Table 172. UART Line Status Register (LSR - address 0x4000 C014, Read Only) bit description

Bit	Symbol	Value	Description	Reset value
0	RDR		Receiver Data Ready. LSR[0] is set when the RBR holds an unread character and is cleared when the UART RBR FIFO is empty.	0
		0	RBR is empty.	
		1	RBR contains valid data.	
1	OE		Overrun Error. The overrun error condition is set as soon as it occurs. An LSR read clears LSR[1]. LSR[1] is set when UART RSR has a new character assembled and the UART RBR FIFO is full. In this case, the UART RBR FIFO will not be overwritten and the character in the UART RSR will be lost.	0
		0	Overrun error status is inactive.	
		1	Overrun error status is active.	
2	PE		Parity Error. When the parity bit of a received character is in the wrong state, a parity error occurs. An LSR read clears LSR[2]. Time of parity error detection is dependent on FCR[0]. **Note:** A parity error is associated with the character at the top of the UART RBR FIFO.	0
		0	Parity error status is inactive.	
		1	Parity error status is active.	

Table 172. UART Line Status Register (LSR - address 0x4000 C014, Read Only) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
3	FE		Framing Error. When the stop bit of a received character is a logic 0, a framing error occurs. An LSR read clears LSR[3]. The time of the framing error detection is dependent on FCR0. Upon detection of a framing error, the RX will attempt to resynchronize to the data and assume that the bad stop bit is actually an early start bit. However, it cannot be assumed that the next received byte will be correct even if there is no Framing Error. **Note:** A framing error is associated with the character at the top of the UART RBR FIFO.	0
		0	Framing error status is inactive.	
		1	Framing error status is active.	
4	BI		Break Interrupt. When RXD1 is held in the spacing state (all zeros) for one full character transmission (start, data, parity, stop), a break interrupt occurs. Once the break condition has been detected, the receiver goes idle until RXD1 goes to marking state (all ones). An LSR read clears this status bit. The time of break detection is dependent on FCR[0]. **Note:** The break interrupt is associated with the character at the top of the UART RBR FIFO.	0
		0	Break interrupt status is inactive.	
		1	Break interrupt status is active.	
5	THRE		Transmitter Holding Register Empty. THRE is set immediately upon detection of an empty UART THR and is cleared on a THR write.	1
		0	THR contains valid data.	
		1	THR is empty.	
6	TEMT		Transmitter Empty. TEMT is set when both THR and TSR are empty; TEMT is cleared when either the TSR or the THR contain valid data.	1
		0	THR and/or the TSR contains valid data.	
		1	THR and the TSR are empty.	
7	RXFE		Error in RX FIFO. LSR[7] is set when a character with a RX error such as framing error, parity error or break interrupt, is loaded into the RBR. This bit is cleared when the LSR register is read and there are no subsequent errors in the UART FIFO.	0
		0	RBR contains no UART RX errors or FCR[0]=0.	
		1	UART RBR contains at least one UART RX error.	
31:8	-	-	Reserved	-

10.5.9 UART Scratch Pad Register

The SCR has no effect on the UART operation. This register can be written and/or read at user's discretion. There is no provision in the interrupt interface that would indicate to the host that a read or write of the SCR has occurred.

Table 173. UART Scratch Pad Register (SCR - address 0x4000 C01C) bit description

Bit	Symbol	Description	Reset value
7:0	Pad	A readable, writable byte.	0x00
31:8	-	Reserved	-

10.5.10 UART Auto-baud Control Register

The UART Auto-baud Control Register (ACR) controls the process of measuring the incoming clock/data rate for the baud rate generation and can be read and written at user's discretion.

Table 174. Auto-baud Control Register (ACR - address 0x4000 C020) bit description

Bit	Symbol	Value	Description	Reset value
0	START		This bit is automatically cleared after auto-baud completion.	0
		0	Auto-baud stop (auto-baud is not running).	
		1	Auto-baud start (auto-baud is running). Auto-baud run bit. This bit is automatically cleared after auto-baud completion.	
1	MODE		Auto-baud mode select bit.	0
		0	Mode 0.	
		1	Mode 1.	
2	AUTORESTART	0	No restart	0
		1	Restart in case of time-out (counter restarts at next UART RX falling edge)	
7:3	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0
8	ABEOINTCLR		End of auto-baud interrupt clear bit (write only accessible).	0
		0	Writing a 0 has no impact.	
		1	Writing a 1 will clear the corresponding interrupt in the IIR.	
9	ABTOINTCLR		Auto-baud time-out interrupt clear bit (write only accessible).	0
		0	Writing a 0 has no impact.	
		1	Writing a 1 will clear the corresponding interrupt in the IIR.	
31:10	-	NA	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

10.5.10.1 Auto-baud

The UART auto-baud function can be used to measure the incoming baud-rate based on the "AT" protocol (Hayes command). If enabled the auto-baud feature will measure the bit time of the receive data stream and set the divisor latch registers DLM and DLL accordingly.

Auto-baud is started by setting the ACR Start bit. Auto-baud can be stopped by clearing the ACR Start bit. The Start bit will clear once auto-baud has finished and reading the bit will return the status of auto-baud (pending/finished).

Two auto-baud measuring modes are available which can be selected by the ACR Mode bit. In mode 0 the baud-rate is measured on two subsequent falling edges of the UART RX pin (the falling edge of the start bit and the falling edge of the least significant bit). In mode 1 the baud-rate is measured between the falling edge and the subsequent rising edge of the UART RX pin (the length of the start bit).

The ACR AutoRestart bit can be used to automatically restart baud-rate measurement if a time-out occurs (the rate measurement counter overflows). If this bit is set the rate measurement will restart at the next falling edge of the UART RX pin.

The auto-baud function can generate two interrupts.

- The IIR ABTOInt interrupt will get set if the interrupt is enabled (IER ABToIntEn is set and the auto-baud rate measurement counter overflows).
- The IIR ABEOInt interrupt will get set if the interrupt is enabled (IER ABEOIntEn is set and the auto-baud has completed successfully).

The auto-baud interrupts have to be cleared by setting the corresponding ACR ABTOIntClr and ABEOIntEn bits.

Typically the fractional baud-rate generator is disabled (DIVADDVAL = 0) during auto-baud. However, if the fractional baud-rate generator is enabled (DIVADDVAL > 0), it is going to impact the measuring of UART RX pin baud-rate, but the value of the FDR register is not going to be modified after rate measurement. Also, when auto-baud is used, any write to DLM and DLL registers should be done before ACR register write. The minimum and the maximum baudrates supported by UART are function of UART_PCLK, number of data bits, stop bits and parity bits.

(4)

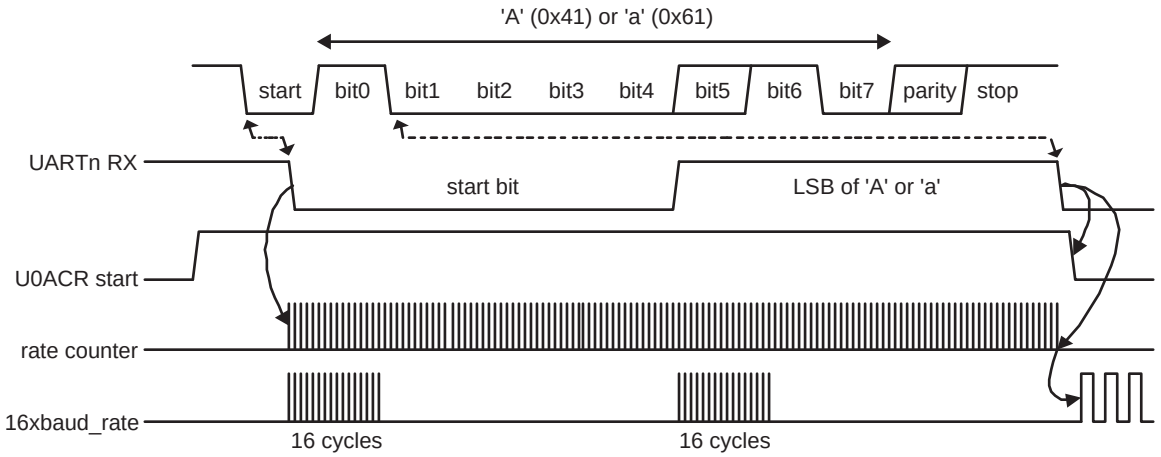
$$ratemin = \frac{2 \times PCLK}{16 \times 2^{15}} \leq UART_{baudrate} \leq \frac{PCLK}{16 \times (2 + databits + paritybits + stopbits)} = ratemax$$

10.5.10.2 Auto-baud modes

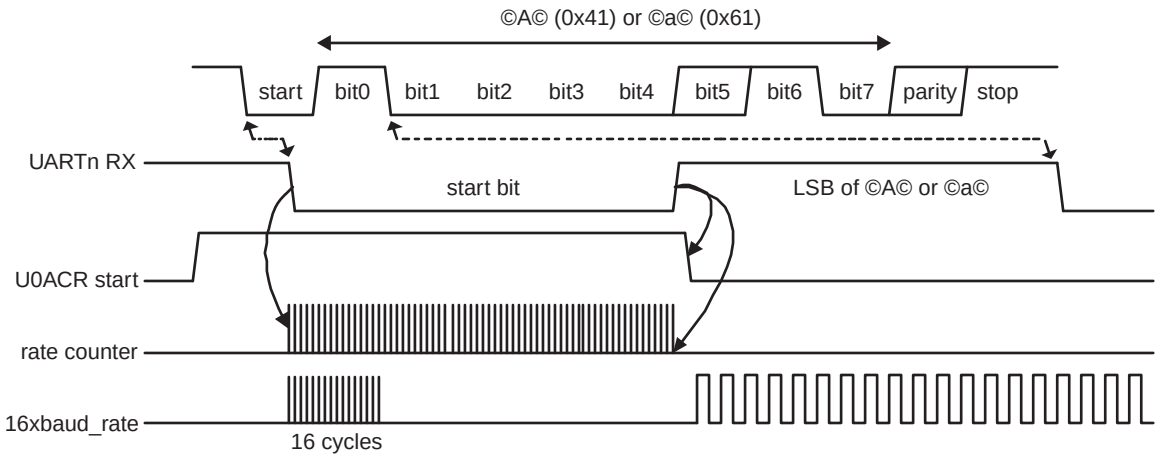
When the software is expecting an ”AT" command, it configures the UART with the expected character format and sets the ACR Start bit. The initial values in the divisor latches DLM and DLM don‘t care. Because of the ”A" or ”a" ASCII coding (”A" = 0x41, ”a" = 0x61), the UART RX pin sensed start bit and the LSB of the expected character are delimited by two falling edges. When the ACR Start bit is set, the auto-baud protocol will execute the following phases:

1. On ACR Start bit setting, the baud-rate measurement counter is reset and the UART RSR is reset. The RSR baud rate is switch to the highest rate.

2. A falling edge on UART RX pin triggers the beginning of the start bit. The rate measuring counter will start counting UART_PCLK cycles optionally pre-scaled by the fractional baud-rate generator.
3. During the receipt of the start bit, 16 pulses are generated on the RSR baud input with the frequency of the (fractional baud-rate pre-scaled) UART input clock, guaranteeing the start bit is stored in the RSR.
4. During the receipt of the start bit (and the character LSB for mode = 0) the rate counter will continue incrementing with the pre-scaled UART input clock (UART_PCLK).
5. If Mode = 0 then the rate counter will stop on next falling edge of the UART RX pin. If Mode = 1 then the rate counter will stop on the next rising edge of the UART RX pin.
6. The rate counter is loaded into DLM/DLL and the baud-rate will be switched to normal operation. After setting the DLM/DLL the end of auto-baud interrupt IIR ABEOInt will be set, if enabled. The RSR will now continue receiving the remaining bits of the ”A/a" character.



a. Mode 0 (start bit and LSB are used for auto-baud)



b. Mode 1 (only start bit is used for auto-baud)

Fig 10. Auto-baud a) mode 0 and b) mode 1 waveform

10.5.11 UART IrDA Control Register

The IrDA Control Register enables and configures the IrDA mode on each UART. The value of ICR should not be changed while transmitting or receiving data, or data loss or corruption may occur.

Table 175: UART IrDA Control Register (ICR - 0x4000 C024) bit description

Bit	Symbol	Value	Description	Reset value
0	IRDAEN		IrDA mode enable.	0
		0	IrDA mode on UARTn is disabled, UARTn acts as a standard UART.	
		1	IrDA mode on UARTn is enabled.	
1	IRDAINV		IrDA input polarity.	0
		0	The serial input is not inverted.	
		1	The serial input is inverted. This has no effect on the serial output.	
2	FIXPULSEEN		IrDA fixed pulse width mode.	0
		0	IrDA fixed pulse width mode disabled.	
		1	IrDA fixed pulse width mode enabled.	

Table 175: UART IrDA Control Register (ICR - 0x4000 C024) bit description

Bit	Symbol	Value	Description	Reset value
5:3	PULSEDIV		Configures the pulse when FixPulseEn = 1.	0
		0x0	$2 \times T_{PCLK}$	
		0x1	$4 \times T_{PCLK}$	
		0x2	$8 \times T_{PCLK}$	
		0x3	$16 \times T_{PCLK}$	
		0x4	$32 \times T_{PCLK}$	
		0x5	$64 \times T_{PCLK}$	
		0x6	$128 \times T_{PCLK}$	
		0x7	$256 \times T_{PCLK}$	
31:6	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

The PulseDiv bits in ICR are used to select the pulse width when the fixed pulse width mode is used in IrDA mode (IrDAEn = 1 and FixPulseEn = 1). The value of these bits should be set so that the resulting pulse width is at least 1.63 µs. Table 176 shows the possible pulse widths.

Table 176: IrDA Pulse Width

FixPulseEn	PulseDiv	IrDA Transmitter Pulse width (µs)
0	x	3 / (16 × baud rate)
1	0	$2 \times T_{PCLK}$
1	1	$4 \times T_{PCLK}$
1	2	$8 \times T_{PCLK}$
1	3	$16 \times T_{PCLK}$
1	4	$32 \times T_{PCLK}$
1	5	$64 \times T_{PCLK}$
1	6	$128 \times T_{PCLK}$
1	7	$256 \times T_{PCLK}$

10.5.12 UART Fractional Divider Register

The UART Fractional Divider Register (FDR) controls the clock pre-scaler for the baud rate generation and can be read and written at the user's discretion. This pre-scaler takes the APB clock and generates an output clock according to the specified fractional requirements.

Important: If the fractional divider is active (DIVADDVAL > 0) and DLM = 0, the value of the DLL register must be 3 or greater.

Table 177. UART Fractional Divider Register (FDR - address 0x4000 C028) bit description

Bit	Symbol	Description	Reset value
3:0	DIVADDVAL	Baud-rate generation pre-scaler divisor value. If this field is 0, fractional baud-rate generator will not impact the UARTn baudrate.	0
7:4	MULVAL	Baud-rate pre-scaler multiplier value. This field must be greater or equal 1 for UARTn to operate properly, regardless of whether the fractional baud-rate generator is used or not.	1
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	0

This register controls the clock pre-scaler for the baud rate generation. The reset value of the register keeps the fractional capabilities of UART disabled making sure that UART is fully software and hardware compatible with UARTs not equipped with this feature.

The UART baudrate can be calculated as (n = 1):

(5)

$$UART_{baudrate} = \frac{PCLK}{16 \times (256 \times U1DLM + U1DLL) \times \left(1 + \frac{DivAddVal}{MulVal}\right)}$$

Where UART_PCLK is the peripheral clock, DLM and DLL are the standard UART baud rate divider registers, and DIVADDVAL and MULVAL are UART fractional baudrate generator specific parameters.

The value of MULVAL and DIVADDVAL should comply to the following conditions:

1. $1 \leq MULVAL \leq 15$
2. $0 \leq DIVADDVAL < 15$
3. DIVADDVAL<MULVAL

The value of the FDR should not be modified while transmitting/receiving data or data may be lost or corrupted.

If the FDR register value does not comply to these two requests, then the fractional divider output is undefined. If DIVADDVAL is zero then the fractional divider is disabled, and the clock will not be divided.

10.5.12.1 Baudrate calculation

UART can operate with or without using the Fractional Divider. In real-life applications it is likely that the desired baudrate can be achieved using several different Fractional Divider settings. The following algorithm illustrates one way of finding a set of DLM, DLL, MULVAL, and DIVADDVAL values. Such set of parameters yields a baudrate with a relative error of less than 1.1% from the desired one.

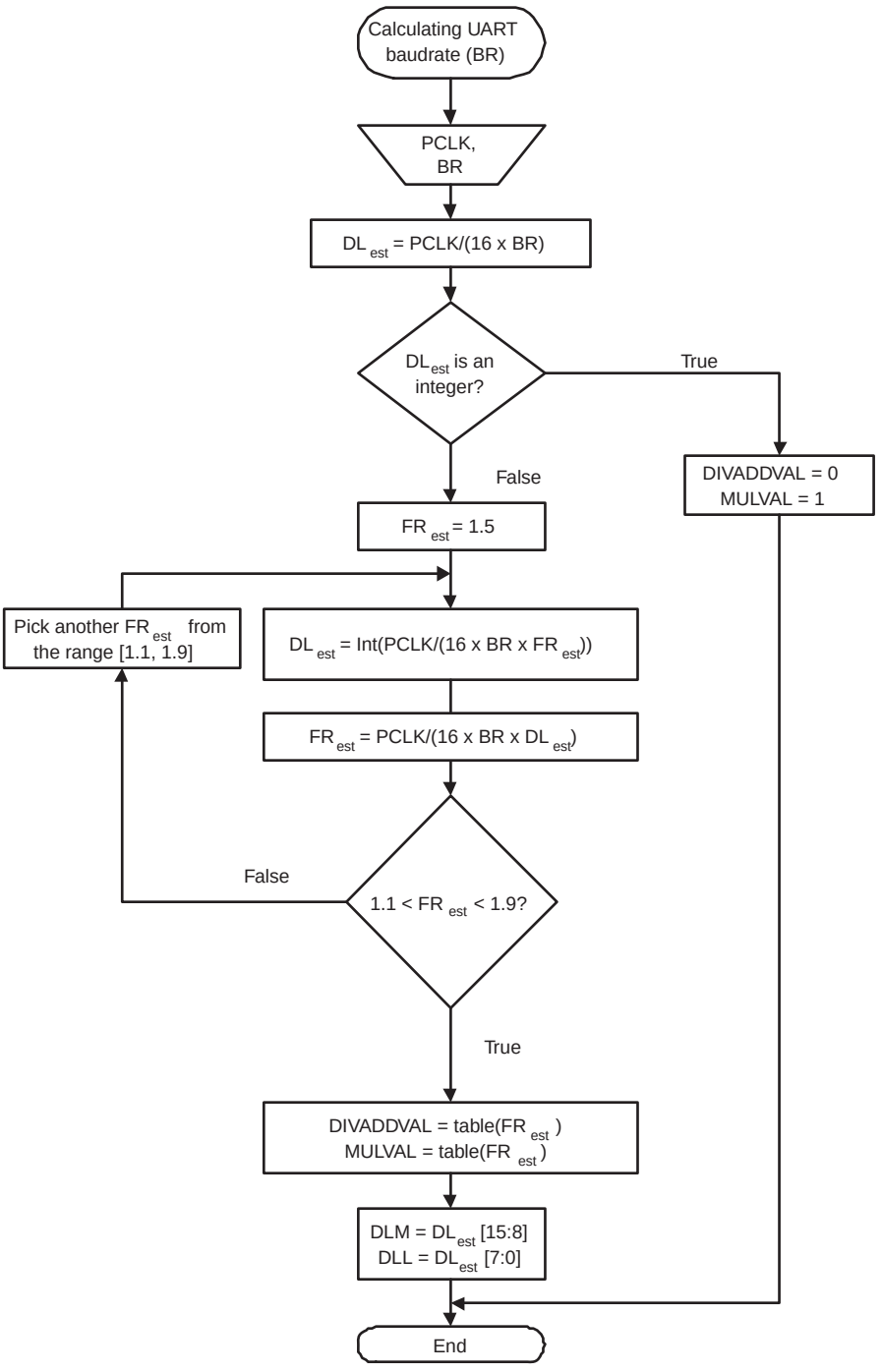


Fig 11. Algorithm for setting UART dividers

Table 178. Fractional Divider setting look-up table

FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal	FR	DivAddVal/ MulVal
1.000	0/1	1.250	1/4	1.500	1/2	1.750	3/4
1.067	1/15	1.267	4/15	1.533	8/15	1.769	10/13
1.071	1/14	1.273	3/11	1.538	7/13	1.778	7/9
1.077	1/13	1.286	2/7	1.545	6/11	1.786	11/14
1.083	1/12	1.300	3/10	1.556	5/9	1.800	4/5
1.091	1/11	1.308	4/13	1.571	4/7	1.818	9/11
1.100	1/10	1.333	1/3	1.583	7/12	1.833	5/6
1.111	1/9	1.357	5/14	1.600	3/5	1.846	11/13
1.125	1/8	1.364	4/11	1.615	8/13	1.857	6/7
1.133	2/15	1.375	3/8	1.625	5/8	1.867	13/15
1.143	1/7	1.385	5/13	1.636	7/11	1.875	7/8
1.154	2/13	1.400	2/5	1.643	9/14	1.889	8/9
1.167	1/6	1.417	5/12	1.667	2/3	1.900	9/10
1.182	2/11	1.429	3/7	1.692	9/13	1.909	10/11
1.200	1/5	1.444	4/9	1.700	7/10	1.917	11/12
1.214	3/14	1.455	5/11	1.714	5/7	1.923	12/13
1.222	2/9	1.462	6/13	1.727	8/11	1.929	13/14
1.231	3/13	1.467	7/15	1.733	11/15	1.933	14/15

10.5.12.1.1 Example 1: UART_PCLK = 14.7456 MHz, BR = 9600

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 14.7456 MHz / (16 x 9600) = 96. Since this DL_{est} is an integer number, DIVADDVAL = 0, MULVAL = 1, DLM = 0, and DLL = 96.

10.5.12.1.2 Example 2: UART_PCLK = 12 MHz, BR = 115200

According to the provided algorithm DL_{est} = PCLK/(16 x BR) = 12 MHz / (16 x 115200) = 6.51. This DL_{est} is not an integer number and the next step is to estimate the FR parameter. Using an initial estimate of FR_{est} = 1.5 a new DL_{est} = 4 is calculated and FR_{est} is recalculated as FR_{est} = 1.628. Since FRest = 1.628 is within the specified range of 1.1 and 1.9, DIVADDVAL and MULVAL values can be obtained from the attached look-up table.

The closest value for FRest = 1.628 in the look-up Table 178 is FR = 1.625. It is equivalent to DIVADDVAL = 5 and MULVAL = 8.

Based on these findings, the suggested UART setup would be: DLM = 0, DLL = 4, DIVADDVAL = 5, and MULVAL = 8. According to Equation 5 UART's is 115384. This rate has a relative error of 0.16% from the originally specified 115200.

10.5.13 UART Transmit Enable Register

In addition to being equipped with full hardware flow control (auto-cts and auto-rts mechanisms described above), TER enables implementation of software flow control, too. When TXEn =1, UART transmitter will keep sending data as long as they are available. As soon as TXEn becomes 0, UART transmission will stop.

Although Table 179 describes how to use TXEn bit in order to achieve hardware flow control, it is strongly suggested to let UART hardware implemented auto flow control features take care of this, and limit the scope of TXEn to software flow control.

TER enables implementation of software and hardware flow control. When TXEn =1, UART transmitter will keep sending data as long as they are available. As soon as TXEn becomes 0, UART transmission will stop.

Table 179 describes how to use TXEn bit in order to achieve software flow control.

Table 179. UART Transmit Enable Register (TER - address 0x4000 C030) bit description

Bit	Symbol	Description	Reset value
6:0	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
7	TXEN	When this bit is 1, as it is after a Reset, data written to the THR is output on the TXD pin as soon as any preceding data has been sent. If this bit cleared to 0 while a character is being sent, the transmission of that character is completed, but no further characters are sent until this bit is set again. In other words, a 0 in this bit blocks the transfer of characters from the THR or TX FIFO into the transmit shift register. Software can clear this bit when it receives an XOFF character (DC3). Software can set this bit again when it receives an XON (DC1) character.	1
31:8	-	Reserved	-

10.5.14 UART FIFO Level register

FIFOLVL register is a Read Only register that allows software to read the current FIFO level status. Both the transmit and receive FIFO levels are present in this register.

Table 180. UART FIFO Level register (FIFOLVL - address 0x4000 C058, Read Only) bit description

Bit	Symbol	Description	Reset value
3:0	RXFIFILVL	Reflects the current level of the UART receiver FIFO. 0 = empty, 0xF = FIFO full.	0x00
7:4	-	Reserved. The value read from a reserved bit is not defined.	-
11:8	TXFIFOLVL	Reflects the current level of the UART transmitter FIFO. 0 = empty, 0xF = FIFO full.	0x00
31:12	-	Reserved. The value read from a reserved bit is not defined.	NA

10.6 Architecture

The architecture of the UART is shown below in the block diagram.

The APB interface provides a communications link between the CPU or host and the UART.

The UART receiver block, RX, monitors the serial input line, RXD, for valid input. The UART RX Shift Register (RSR) accepts valid characters via RXD. After a valid character is assembled in the RSR, it is passed to the UART RX Buffer Register FIFO to await access by the CPU or host via the generic host interface.

The UART transmitter block, TX, accepts data written by the CPU or host and buffers the data in the UART TX Holding Register FIFO (THR). The UART TX Shift Register (TSR) reads the data stored in the THR and assembles the data to transmit via the serial output pin, TXD1.

The UART Baud Rate Generator block, BRG, generates the timing enables used by the UART TX block. The BRG clock input source is UART_PCLK. The main clock is divided down per the divisor specified in the DLL and DLM registers. This divided down clock is a 16x oversample clock, NBAUDOUT.

The interrupt interface contains registers IER and IIR. The interrupt interface receives several one clock wide enables from the TX and RX blocks.

Status information from the TX and RX is stored in the LSR. Control information for the TX and RX is stored in the LCR.

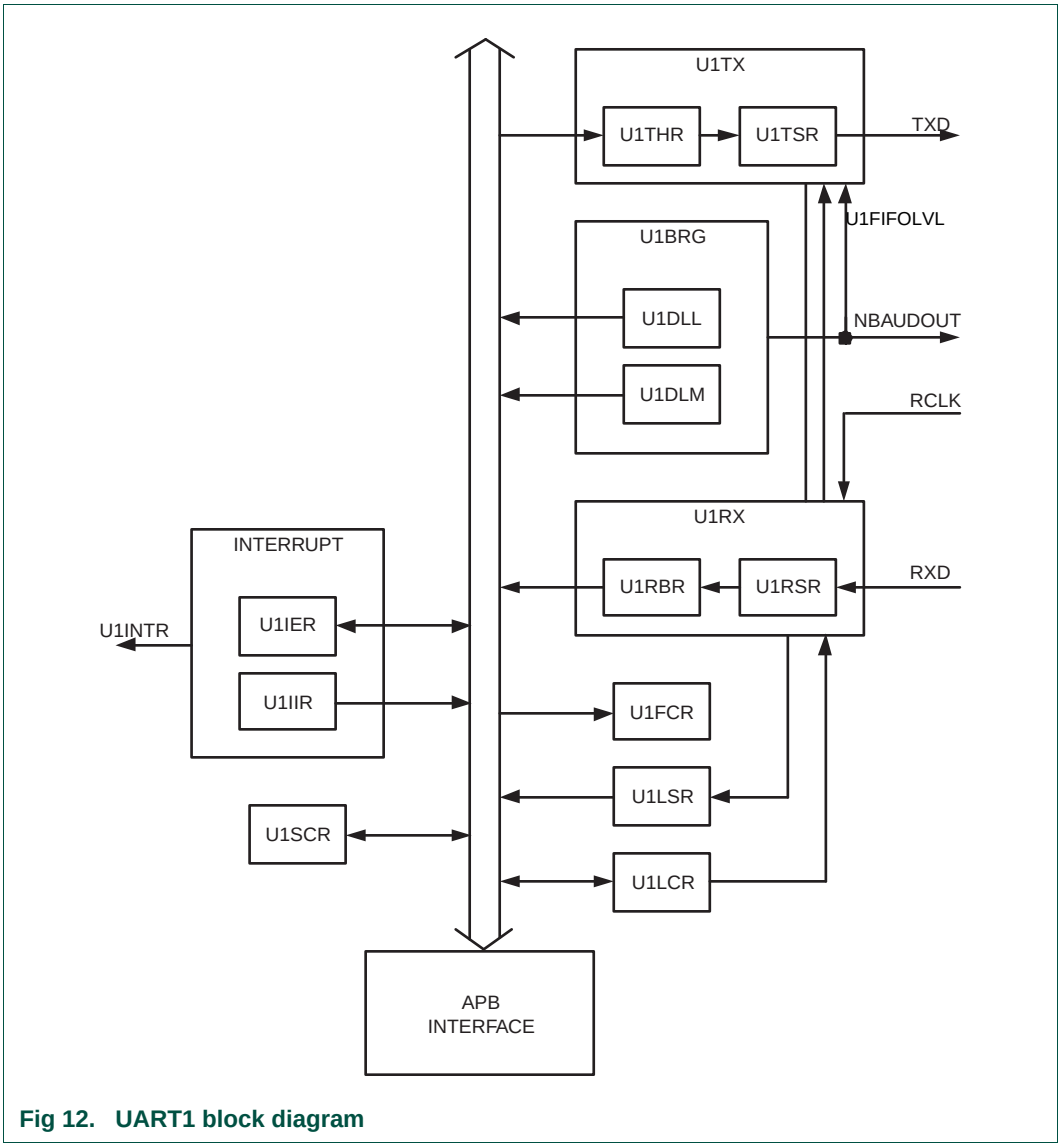


Fig 12. UART1 block diagram

11.1 How to read this chapter

The I^{2}C-bus controller is available on all LPC122x parts.

11.2 Basic configuration

The peripheral clock to the I2C block I2C_PCLK is provided by the system clock, which is controlled by the SYSAHBCLKDIV register (Table 21). The I2C block can be disabled through the System AHB clock control register bit 5 (Table 21) for power savings.

11.3 Features

- Standard I^{2}C-compliant bus interfaces may be configured as Master, Slave, or Master/Slave.
- Arbitration is handled between simultaneously transmitting masters without corruption of serial data on the bus.
- Programmable clock allows adjustment of I^{2}C transfer rates.
- Data transfer is bidirectional between masters and slaves.
- Serial clock synchronization allows devices with different bit rates to communicate via one serial bus.
- Serial clock synchronization is used as a handshake mechanism to suspend and resume serial transfer.
- Supports Fast-mode Plus.
- Optional recognition of up to four distinct slave addresses.
- Monitor mode allows observing all I^{2}C-bus traffic, regardless of slave address.
- I^{2}C-bus can be used for test and diagnostic purposes.
- The I^{2}C block contains a standard I^{2}C-compliant bus interface with two pins.

11.4 Applications

Interfaces to external I^{2}C standard parts, such as serial RAMs, LCDs, tone generators, other microcontrollers, etc.

11.5 Description

A typical I^{2}C-bus configuration is shown in Figure 13. Depending on the state of the direction bit (R/W), two types of data transfers are possible on the I^{2}C-bus:

- Data transfer from a master transmitter to a slave receiver. The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte.

- Data transfer from a slave transmitter to a master receiver. The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. Next follows the data bytes transmitted by the slave to the master. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a “not acknowledge” is returned. The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a Repeated START condition. Since a Repeated START condition is also the beginning of the next serial transfer, the I^2C bus will not be released.

The I^2C interface is byte oriented and has four operating modes: master transmitter mode, master receiver mode, slave transmitter mode and slave receiver mode.

The I^2C interface complies with the entire I^2C specification, supporting the ability to turn power off to the ARM Cortex-M0 without interfering with other devices on the same I^2C-bus.

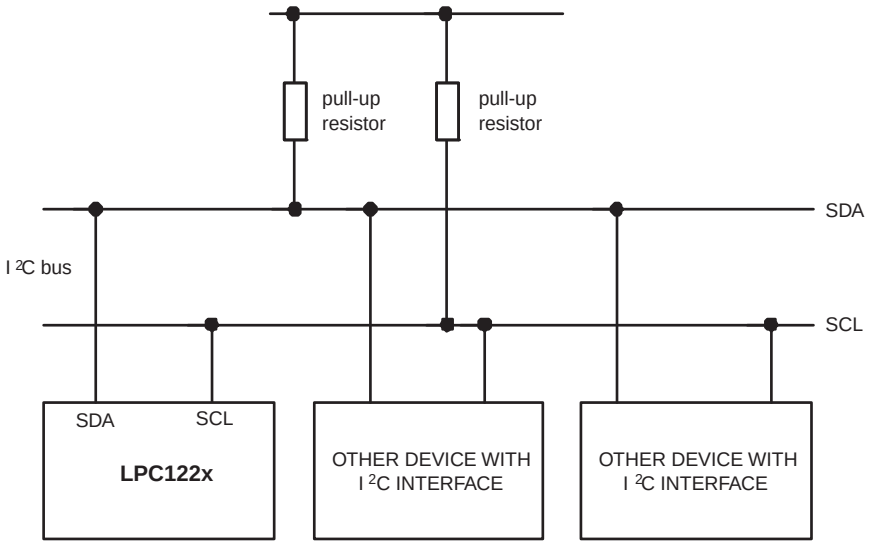


Fig 13. I^2C-bus configuration

11.5.1 I^2C Fast-mode Plus

Fast-Mode Plus supports a 1 Mbit/sec transfer rate to communicate with the I^2C products which NXP Semiconductors is now providing.

In order to use Fast-Mode Plus, the I^2C pins must be properly configured in the IOCONFIG register block, see Table 96 and Table 97. In Fast-mode Plus, rates above 400 kHz and up to 1 MHz may be selected, see Table 189.

11.6 Pin description

Table 181. I²C-bus pin description

Pin	Type	Description
SDA	Input/Output	I²C Serial Data
SCL	Input/Output	I²C Serial Clock

The I²C-bus pins must be configured through the IOCON_PIO0_10 (Table 96) and IOCON_PIO0_11 (Table 97) registers for standard/ Fast-mode or Fast-mode Plus. The I²C-bus pins are open-drain outputs and fully compatible with the I²C-bus specification.

11.7 Register description

Table 182. Register overview: I²C (base address 0x4000 0000)

Name	Access	Address offset	Description	Reset value[1]
CONSET	R/W	0x000	**I2C Control Set Register.** When a one is written to a bit of this register, the corresponding bit in the I²C control register is set. Writing a zero has no effect on the corresponding bit in the I²C control register.	0x00
STAT	RO	0x004	**I2C Status Register.** During I²C operation, this register provides detailed status codes that allow software to determine the next action needed.	0xF8
DAT	R/W	0x008	**I2C Data Register.** During master or slave transmit mode, data to be transmitted is written to this register. During master or slave receive mode, data that has been received may be read from this register.	0x00
ADR0	R/W	0x00C	**I2C Slave Address Register 0.** Contains the 7-bit slave address for operation of the I²C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	0x00
SCLH	R/W	0x010	**SCH Duty Cycle Register High Half Word.** Determines the high time of the I²C clock.	0x04
I2SCLL	R/W	0x014	**SCL Duty Cycle Register Low Half Word.** Determines the low time of the I²C clock. I2nSCLL and I2nSCLH together determine the clock frequency generated by an I²C master and certain times used in slave mode.	0x04
CONCLR	WO	0x018	**I2C Control Clear Register.** When a one is written to a bit of this register, the corresponding bit in the I²C control register is cleared. Writing a zero has no effect on the corresponding bit in the I²C control register.	NA
MMCTRL	R/W	0x01C	**Monitor mode control register.**	0x00
ADR1	R/W	0x020	**I2C Slave Address Register 1.** Contains the 7-bit slave address for operation of the I²C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	0x00
ADR2	R/W	0x024	**I2C Slave Address Register 2.** Contains the 7-bit slave address for operation of the I²C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	0x00

Table 182. Register overview: I²C (base address 0x4000 0000) *…continued*

Name	Access	Address offset	Description	Reset value[1]
ADR3	R/W	0x028	**I2C Slave Address Register 3.** Contains the 7-bit slave address for operation of the I²C interface in slave mode, and is not used in master mode. The least significant bit determines whether a slave responds to the General Call address.	0x00
DATA_ BUFFER	RO	0x02C	**Data buffer register.** The contents of the 8 MSBs of the I2DAT shift register will be transferred to the DATA_BUFFER automatically after every nine bits (8 bits of data plus ACK or NACK) has been received on the bus.	0x00
MASK0	R/W	0x030	**I2C Slave address mask register 0**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	0x00
MASK1	R/W	0x034	**I2C Slave address mask register 1**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	0x00
MASK2	R/W	0x038	**I2C Slave address mask register 2**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	0x00
MASK3	R/W	0x03C	**I2C Slave address mask register 3**. This mask register is associated with I2ADR0 to determine an address match. The mask register has no effect when comparing to the General Call address (‘0000000’).	0x00

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

11.7.1 I²C Control Set register (CONSET - 0x4000 0000)

The I2CONSET registers control setting of bits in the I2CON register that controls operation of the I²C interface. Writing a one to a bit of this register causes the corresponding bit in the I²C control register to be set. Writing a zero has no effect.

Table 183. I²C Control Set register (CONSET - address 0x4000 0000) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	AA	Assert acknowledge flag.	
3	SI	I²C interrupt flag.	0
4	STO	STOP flag.	0
5	STA	START flag.	0
6	I2EN	I²C interface enable.	0
7	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
31:8	-	Reserved	-

I2EN I²C Interface Enable. When I2EN is 1, the I²C interface is enabled. I2EN can be cleared by writing 1 to the I2ENC bit in the I2CONCLR register. When I2EN is 0, the I²C interface is disabled.

When I2EN is “0”, the SDA and SCL input signals are ignored, the I^2C block is in the “not addressed” slave state, and the STO bit is forced to “0”.

I2EN should not be used to temporarily release the I^2C-bus since, when I2EN is reset, the I^2C-bus status is lost. The AA flag should be used instead.

STA is the START flag. Setting this bit causes the I^2C interface to enter master mode and transmit a START condition or transmit a Repeated START condition if it is already in master mode.

When STA is 1 and the I^2C interface is not already in master mode, it enters master mode, checks the bus and generates a START condition if the bus is free. If the bus is not free, it waits for a STOP condition (which will free the bus) and generates a START condition after a delay of a half clock period of the internal clock generator. If the I^2C interface is already in master mode and data has been transmitted or received, it transmits a Repeated START condition. STA may be set at any time, including when the I^2C interface is in an addressed slave mode.

STA can be cleared by writing 1 to the STAC bit in the I2CONCLR register. When STA is 0, no START condition or Repeated START condition will be generated.

If STA and STO are both set, then a STOP condition is transmitted on the I^2C-bus if it the interface is in master mode, and transmits a START condition thereafter. If the I^2C interface is in slave mode, an internal STOP condition is generated, but is not transmitted on the bus.

STO is the STOP flag. Setting this bit causes the I^2C interface to transmit a STOP condition in master mode, or recover from an error condition in slave mode. When STO is 1 in master mode, a STOP condition is transmitted on the I^2C-bus. When the bus detects the STOP condition, STO is cleared automatically.

In slave mode, setting this bit can recover from an error condition. In this case, no STOP condition is transmitted to the bus. The hardware behaves as if a STOP condition has been received and it switches to “not addressed” slave receiver mode. The STO flag is cleared by hardware automatically.

SI is the I^2C Interrupt Flag. This bit is set when the I^2C state changes. However, entering state F8 does not set SI since there is nothing for an interrupt service routine to do in that case.

While SI is set, the low period of the serial clock on the SCL line is stretched, and the serial transfer is suspended. When SCL is HIGH, it is unaffected by the state of the SI flag. SI must be reset by software, by writing a 1 to the SIC bit in I2CONCLR register.

AA is the Assert Acknowledge Flag. When set to 1, an acknowledge (low level to SDA) will be returned during the acknowledge clock pulse on the SCL line on the following situations:

1. The address in the Slave Address Register has been received.
2. The General Call address has been received while the General Call bit (GC) in I2ADR is set.
3. A data byte has been received while the I^2C is in the master receiver mode.
4. A data byte has been received while the I^2C is in the addressed slave receiver mode

The AA bit can be cleared by writing 1 to the AAC bit in the I2CONCLR register. When AA is 0, a not acknowledge (HIGH level to SDA) will be returned during the acknowledge clock pulse on the SCL line on the following situations:

1. A data byte has been received while the I^2C is in the master receiver mode.
2. A data byte has been received while the I^2C is in the addressed slave receiver mode.

11.7.2 I^2C Status register (STAT - 0x4000 0004)

Each I^2C Status register reflects the condition of the corresponding I^2C interface. The I^2C Status register is Read-Only.

Table 184. I^2C Status register (STAT - 0x4000 0004) bit description

Bit	Symbol	Description	Reset value
2:0	-	These bits are unused and are always 0.	0
7:3	Status	These bits give the actual status information about the I^2C interface.	0x1F
31:8	-	Reserved	-

The three least significant bits are always 0. Taken as a byte, the status register contents represent a status code. There are 26 possible status codes. When the status code is 0xF8, there is no relevant information available and the SI bit is not set. All other 25 status codes correspond to defined I^2C states. When any of these states entered, the SI bit will be set. For a complete list of status codes, refer to tables from Table 200 to Table 203.

11.7.3 I^2C Data register (DAT - 0x4000 0008)

This register contains the data to be transmitted or the data just received. The CPU can read and write to this register only while it is not in the process of shifting a byte, when the SI bit is set. Data in I2DAT remains stable as long as the SI bit is set. Data in I2DAT is always shifted from right to left: the first bit to be transmitted is the MSB (bit 7), and after a byte has been received, the first bit of received data is located at the MSB of I2DAT.

Table 185. I^2C Data register (DAT - 0x4000 0008) bit description

Bit	Symbol	Description	Reset value
7:0	Data	This register holds data values that have been received or are to be transmitted.	0
31:8	-	Reserved	-

11.7.4 I^2C Slave Address register 0 (ADR0- 0x4000 000C)

This register is readable and writable and are only used when an I^2C interface is set to slave mode. In master mode, this register has no effect. The LSB of I2ADR is the General Call bit. When this bit is set, the General Call address (0x00) is recognized.

Any of these registers which contain the bit 00x will be disabled and will not match any address on the bus. This register will be cleared to this disabled state on reset.

Table 186. I²C Slave Address register 0 (ADR0- 0x4000 000C) bit description

Bit	Symbol	Description	Reset value
0	GC	General Call enable bit.	0
7:1	Address	The I²C device address for slave mode.	0x00
31:8	-	Reserved	-

11.7.5 I²C SCL HIGH and LOW duty cycle registers (SCLH - 0x4000 0010 and I2SCLL- 0x4000 0014)

Table 187. I²C SCL HIGH Duty Cycle register (SCLH - address 0x4000 0010) bit description

Bit	Symbol	Description	Reset value
15:0	SCLH	Count for SCL HIGH time period selection.	0x0004
31:16	-	Reserved	-

Table 188. I²C SCL Low duty cycle register (SCLL - 0x4000 0014) bit description

Bit	Symbol	Description	Reset value
15:0	SCLL	Count for SCL low time period selection.	0x0004
31:16	-	Reserved	-

11.7.5.1 Selecting the appropriate I²C data rate and duty cycle

Software must set values for the registers I2SCLH and I2SCLL to select the appropriate data rate and duty cycle. I2SCLH defines the number of I2C_PCLK cycles for the SCL HIGH time, I2SCLL defines the number of I2C_PCLK cycles for the SCL low time. The frequency is determined by the following formula (I2C_PCLK is the frequency of the peripheral bus APB):

(6)

$$I^2C_{bitfrequency} = \frac{I2CPCLK}{I2CSCLH + I2CSCLL}$$

The values for I2SCLL and I2SCLH must ensure that the data rate is in the appropriate I²C data rate range. Each register value must be greater than or equal to 4. Table 189 gives some examples of I²C-bus rates based on I2C_PCLK frequency and I2SCLL and I2SCLH values.

Table 189. I2SCLL + I2SCLH values for selected I2C_PCLK values

	I2C_PCLK (MHz)						
	6	8	10	12	16	20	30
I²C mode	SCLH + SCLL						
100 kHz (Standard)	60	80	100	120	160	200	300
400 kHz (Fast Mode)	15	20	25	30	40	50	75
1 MHz (Fast Mode Plus)	-	8	10	12	16	20	30

I2SCLL and I2SCLH values should not necessarily be the same. Software can set different duty cycles on SCL by setting these two registers. For example, the I^2C-bus specification defines the SCL low time and high time at different values for a Fast-mode and Fast-mode Plus I^2C.

11.7.6 I^2C Control Clear register (CONCLR - 0x4000 0018)

The I2CONCLR registers control clearing of bits in the I2CON register that controls operation of the I^2C interface. Writing a one to a bit of this register causes the corresponding bit in the I^2C control register to be cleared. Writing a zero has no effect.

Table 190. I^2C Control Clear register (CONCLR - 0x4000 0018) bit description

Bit	Symbol	Description	Reset value
1:0	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
2	AAC	Assert acknowledge Clear bit.	
3	SIC	I^2C interrupt Clear bit.	0
4	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
5	STAC	START flag Clear bit.	0
6	I2ENC	I^2C interface Disable bit.	0
7	-	Reserved. User software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
31:8	-	Reserved	-

AAC is the Assert Acknowledge Clear bit. Writing a 1 to this bit clears the AA bit in the I2CONSET register. Writing 0 has no effect.

SIC is the I^2C Interrupt Clear bit. Writing a 1 to this bit clears the SI bit in the I2CONSET register. Writing 0 has no effect.

STAC is the START flag Clear bit. Writing a 1 to this bit clears the STA bit in the I2CONSET register. Writing 0 has no effect.

I2ENC is the I^2C Interface Disable bit. Writing a 1 to this bit clears the I2EN bit in the I2CONSET register. Writing 0 has no effect.

11.7.7 I^2C Monitor mode control register

This register controls the Monitor mode which allows the I^2C module to monitor traffic on the I^2C bus without actually participating in traffic or interfering with the I^2C bus.

Table 191. I²C Monitor mode control register (MMCTRL - 0x4000 001C) bit description

Bit	Symbol	Value	Description	Reset value
0	MM_ENA		Monitor mode enable.	0
		0	Monitor mode disabled.	
		1	The I²C module will enter monitor mode. In this mode the SDA output will be forced high. This will prevent the I²C module from outputting data of any kind (including ACK) onto the I²C data bus. Depending on the state of the ENA_SCL bit, the output may be also forced high, preventing the module from having control over the I²C clock line.	
1	ENA_SCL		SCL output enable.	0
		0	When this bit is cleared to 0, the SCL output will be forced high when the module is in monitor mode. As described above, this will prevent the module from having any control over the I²C clock line.	
		1	When this bit is set, the I²C module may exercise the same control over the clock line that it would in normal operation. This means that, acting as a slave peripheral, the I²C module can stretch the clock line (hold it low) until it has had time to respond to an I²C interrupt. **Remark:** When the ENA_SCL bit is cleared and the I²C no longer has the ability to stall the bus, interrupt response time becomes important. To give the part more time to respond to an I²C interrupt under these conditions, a DATA _BUFFER register is used to hold received data for a full 9-bit word transmission time.	
2	MATCH_ALL		Select interrupt register match.	0
		0	When this bit is cleared, an interrupt will only be generated when a match occurs to one of the (up-to) four address registers described above. That is, the module will respond as a normal slave as far as address-recognition is concerned.	
		1	When this bit is set to 1 and the I²C is in monitor mode, an interrupt will be generated on ANY address received. This will enable the part to monitor all traffic on the bus.	
31:3	-	-	Reserved.	-

Remark: The ENA_SCL and MATCH_ALL bits have no effect if the MM_ENA is ‘0’ (i.e. if the module is NOT in monitor mode).

11.7.7.1 Interrupt in Monitor mode

All interrupts will occur as normal when the module is in monitor mode. This means that the first interrupt will occur when an address-match is detected (any address received if the MATCH_ALL bit is set, otherwise an address matching one of the four address registers).

Subsequent to an address-match detection, interrupts will be generated after each data byte is received for a slave-write transfer, or after each byte that the module “thinks” it has transmitted for a slave-read transfer. In this second case, the data register will actually contain data transmitted by some other slave on the bus which was actually addressed by the master.

Following all of these interrupts, the processor may read the data register to see what was actually transmitted on the bus.

11.7.7.2 Loss of arbitration in Monitor mode

In monitor mode, the I2C module will not be able to respond to a request for information by the bus master or issue an ACK). Some other slave on the bus will respond instead. This will most probably result in a lost-arbitration state as far as our module is concerned.

Software should be aware of the fact that the module is in monitor mode and should not respond to any loss of arbitration state that is detected. In addition, hardware may be designed into the module to block some/all loss of arbitration states from occurring if those state would either prevent a desired interrupt from occurring or cause an unwanted interrupt to occur. Whether any such hardware will be added is still to be determined.

11.7.8 I2C Slave Address registers (ADR[1, 2, 3]- 0x4000 00[20, 24, 28])

These registers are readable and writable and are only used when an I2C interface is set to slave mode. In master mode, this register has no effect. The LSB of I2ADR is the General Call bit. When this bit is set, the General Call address (0x00) is recognized.

Any of these registers which contain the bit 00x will be disabled and will not match any address on the bus. All four registers (including the I2ADR0 register) will be cleared to this disabled state on reset.

Table 192. I2C Slave Address registers (ADR1 - 0x4000 0020, ADR2 - 0x4000 0024, ADR3 - 0x4000 0028) bit description

Bit	Symbol	Description	Reset value
0	GC	General Call enable bit.	0
7:1	Address	The I2C device address for slave mode.	0x00
31:8	-	Reserved	-

11.7.9 I2C Data buffer register (DATA_BUFFER - 0x4000 002C)

In monitor mode, the I2C module may lose the ability to stretch the clock (stall the bus) if the ENA_SCL bit is not set. This means that the processor will have a limited amount of time to read the contents of the data received on the bus. If the processor reads the I2DAT shift register, as it ordinarily would, it could have only one bit-time to respond to the interrupt before the received data is overwritten by new data.

To give the processor more time to respond, a new 8-bit, read-only DATA_BUFFER register will be added. The contents of the 8 MSBs of the I2DAT shift register will be transferred to the DATA_BUFFER automatically after every nine bits (8 bits of data plus ACK or NACK) has been received on the bus. This means that the processor will have nine bit transmission times to respond to the interrupt and read the data before it is overwritten.

The processor will still have the ability to read I2DAT directly, as usual, and the behavior of I2DAT will not be altered in any way.

Although the DATA_BUFFER register is primarily intended for use in monitor mode with the ENA_SCL bit = ‘0’, it will be available for reading at any time under any mode of operation.

Table 193. I²C Data buffer register (DATA_BUFFER - 0x4000 002C) bit description

Bit	Symbol	Description	Reset value
7:0	Data	This register holds contents of the 8 MSBs of the I2DAT shift register.	0
31:8	-	Reserved	-

11.7.10 I²C Mask registers (MASK[0, 1, 2, 3] - 0x4000 00[30, 34, 38, 3C])

The four mask registers each contain seven active bits (7:1). Any bit in these registers which is set to '1' will cause an automatic compare on the corresponding bit of the received address when it is compared to the I2ADDRn register associated with that mask register. In other words, bits in an I2ADDRn register which are masked are not taken into account in determining an address match.

On reset, all mask register bits are cleared to '0'.

The mask register has no effect on comparison to the General Call address ("0000000").

Bits(31:8) and bit(0) of the mask registers are unused and should not be written to. These bits will always read back as zeros.

When an address-match interrupt occurs, the processor will have to read the data register (DAT) to determine what the received address was that actually caused the match.

Table 194. I²C Mask registers (MASK0 - 0x4000 0030, MASK1 - 0x4000 0034, MASK2 - 0x4000 0038, MASK3 - 0x4000 003C) bit description

Bit	Symbol	Description	Reset value
0	-	Reserved. User software should not write ones to reserved bits. This bit reads always back as 0.	0
7:1	MASK	Mask bits.	0x00
31:8	-	Reserved. User software should not write ones to reserved bits. These bits read always back as 0's.	0

11.8 I²C operating modes

In a given application, the I²C block may operate as a master, a slave, or both. In the slave mode, the I²C hardware looks for any one of its four slave addresses and the General Call address. If one of these addresses is detected, an interrupt is requested. If the processor wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave operation is not interrupted. If bus arbitration is lost in the master mode, the I²C block switches to the slave mode immediately and can detect any of its own configured slave addresses in the same serial transfer.

11.8.1 Master Transmitter mode

In this mode data is transmitted from master to slave. Before the master transmitter mode can be entered, the CONSET register must be initialized as shown in Table 195. I2EN must be set to 1 to enable the I²C function. If the AA bit is 0, the I²C interface will not acknowledge any address when another device is master of the bus, so it can not enter

slave mode. The STA, STO and SI bits must be 0. The SI bit is cleared by writing 1 to the SIC bit in the CONCLR register. THe STA bit should be cleared after writing the slave address.

Table 195. CONSET used to configure Master mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	0	-	-

The first byte transmitted contains the slave address of the receiving device (7 bits) and the data direction bit. In this mode the data direction bit (R/W) should be 0 which means Write. The first byte transmitted contains the slave address and Write bit. Data is transmitted 8 bits at a time. After each byte is transmitted, an acknowledge bit is received. START and STOP conditions are output to indicate the beginning and the end of a serial transfer.

The I^2C interface will enter master transmitter mode when software sets the STA bit. The I^2C logic will send the START condition as soon as the bus is free. After the START condition is transmitted, the SI bit is set, and the status code in the STAT register is 0x08. This status code is used to vector to a state service routine which will load the slave address and Write bit to the DAT register, and then clear the SI bit. SI is cleared by writing a 1 to the SIC bit in the CONCLR register.

When the slave address and R/W bit have been transmitted and an acknowledgment bit has been received, the SI bit is set again, and the possible status codes now are 0x18, 0x20, or 0x38 for the master mode, or 0x68, 0x78, or 0xB0 if the slave mode was enabled (by setting AA to 1). The appropriate actions to be taken for each of these status codes are shown in Table 200 to Table 203.

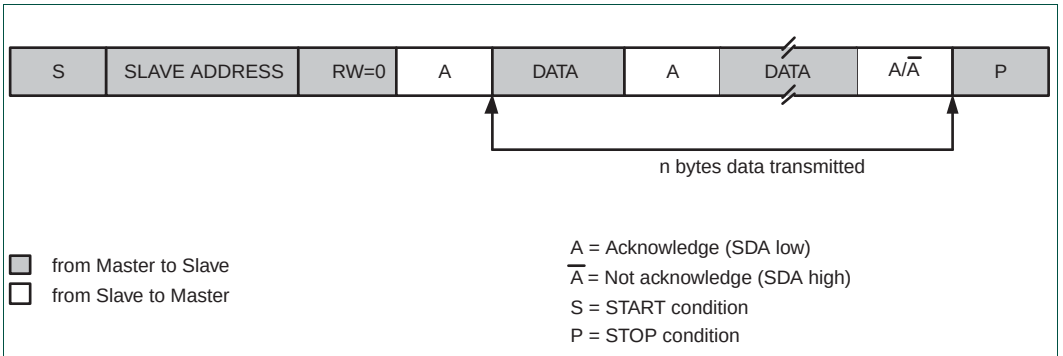


Fig 14. Format in the Master Transmitter mode

11.8.2 Master Receiver mode

In the master receiver mode, data is received from a slave transmitter. The transfer is initiated in the same way as in the master transmitter mode. When the START condition has been transmitted, the interrupt service routine must load the slave address and the data direction bit to the I^2C Data register (DAT), and then clear the SI bit. In this case, the data direction bit (R/W) should be 1 to indicate a read.

When the slave address and data direction bit have been transmitted and an acknowledge bit has been received, the SI bit is set, and the Status Register will show the status code. For master mode, the possible status codes are 0x40, 0x48, or 0x38. For slave mode, the possible status codes are 0x68, 0x78, or 0xB0. For details, refer to Table 201.

When the LPC122x needs to acknowledge a received byte, the AA bit needs to be set accordingly prior to clearing the SI bit and initiating the byte read. When the LPC122x needs to not acknowledge a received byte, the AA bit needs to be cleared prior to clearing the SI bit and initiating the byte read.

Note that the last received byte is always followed by a "Not Acknowledge" from the LPC122x so that the master can signal the slave that the reading sequence is finished and that it needs to issue a STOP or repeated START Command. Once the "Not Acknowledge has been sent and the SI bit is set, the LPC122x can send either a STOP (STO bit is set) or a repeated START (STA bit is set). Then the SI bit is cleared to initiate the requested operation.

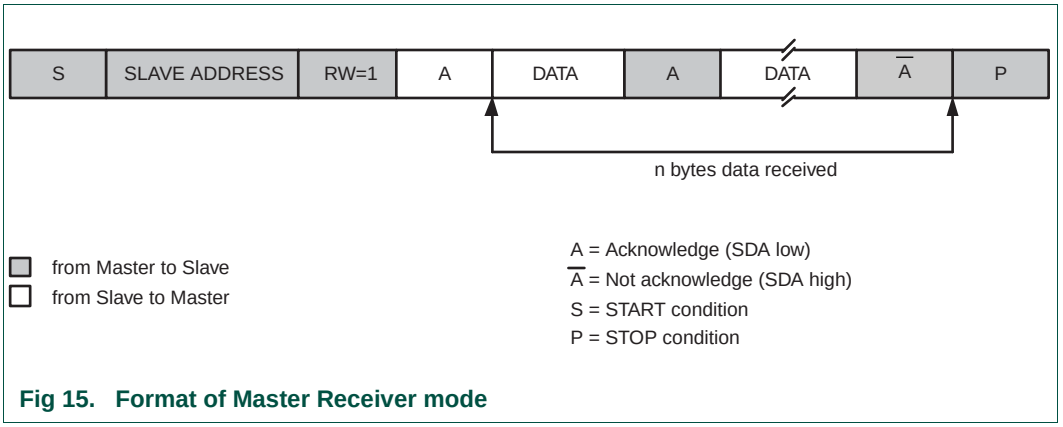


Fig 15. Format of Master Receiver mode

After a repeated START condition, I^2C may switch to the master transmitter mode.

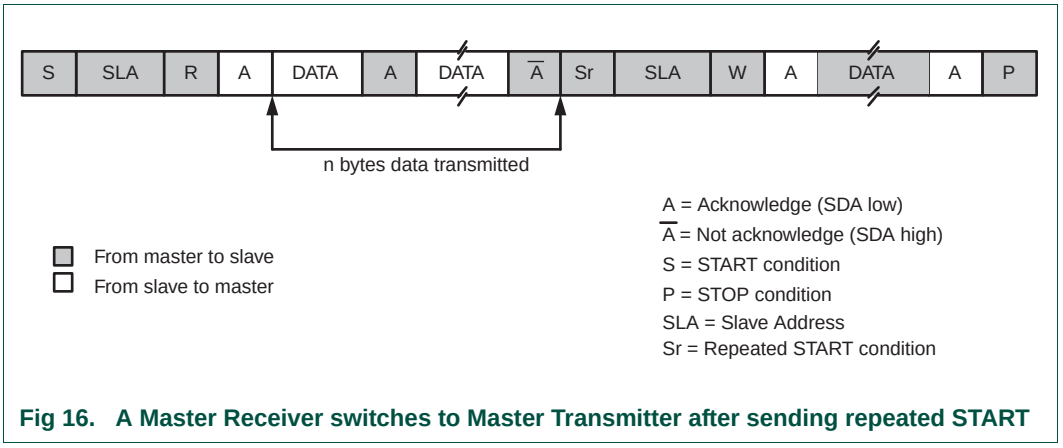


Fig 16. A Master Receiver switches to Master Transmitter after sending repeated START

11.8.3 Slave Receiver mode

In the slave receiver mode, data bytes are received from a master transmitter. To initialize the slave receiver mode, write any of the Slave Address registers (ADR0-3) and Slave Mask registers (MASK0-3) and write the I^2C Control Set register (CONSET) as shown in Table 196.

Table 196. CONSET used to configure Slave mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	1	-	-

I2EN must be set to 1 to enable the I^2C function. AA bit must be set to 1 to acknowledge any of its own slave addresses or the General Call address. The STA, STO and SI bits are set to 0.

After ADR and CONSET are initialized, the I^2C interface waits until it is addressed by its any of its own slave addresses or General Call address followed by the data direction bit. If the direction bit is 0 (W), it enters slave receiver mode. If the direction bit is 1 (R), it enters slave transmitter mode. After the address and direction bit have been received, the SI bit is set and a valid status code can be read from the Status register (STAT). Refer to Table 202 for the status codes and actions.

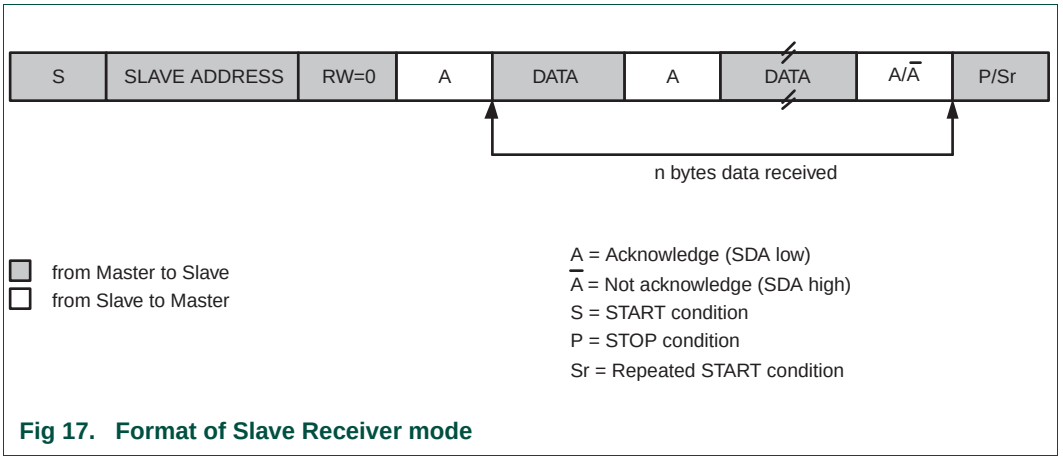


Fig 17. Format of Slave Receiver mode

11.8.4 Slave Transmitter mode

The first byte is received and handled as in the slave receiver mode. However, in this mode, the direction bit will be 1, indicating a read operation. Serial data is transmitted via SDA while the serial clock is input through SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer. In a given application, I^2C may operate as a master and as a slave. In the slave mode, the I^2C hardware looks for any of its own slave addresses and the General Call address. If one of these addresses is detected, an interrupt is requested. When the microcontrollers wishes to become the bus master, the hardware waits until the bus is free before the master mode is entered so that a possible slave action is not interrupted. If bus arbitration is lost in the master mode, the I^2C interface switches to the slave mode immediately and can detect any of its own slave addresses in the same serial transfer.

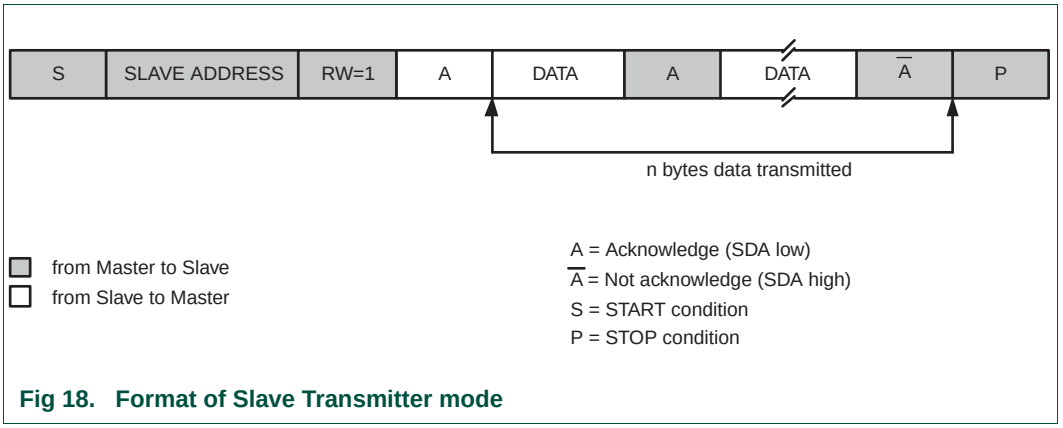


Fig 18. Format of Slave Transmitter mode

11.9 I²C implementation and operation

Figure 19 shows how the on-chip I²C-bus interface is implemented, and the following text describes the individual blocks.

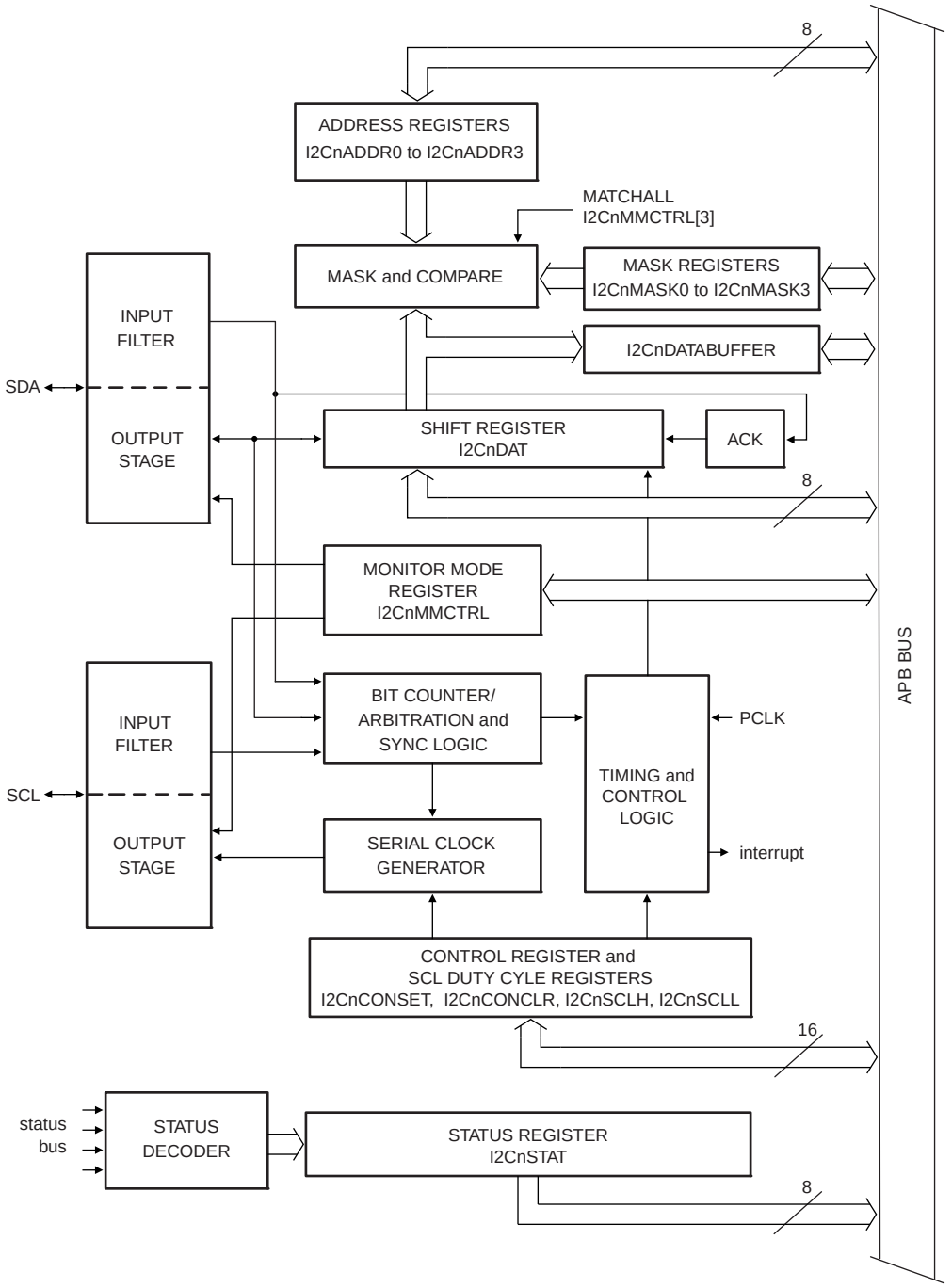


Fig 19. I²C serial interface block diagram

11.9.1 Input filters and output stages

Input signals are synchronized with the internal clock, and spikes shorter than three clocks are filtered out.

The output for I²C is a special pad designed to conform to the I²C specification.

11.9.2 Address Registers, ADR0 to ADR3

These registers may be loaded with the 7-bit slave address (7 most significant bits) to which the I^2C block will respond when programmed as a slave transmitter or receiver. The LSB (GC) is used to enable General Call address (0x00) recognition. When multiple slave addresses are enabled, the actual address received may be read from the DAT register at the state where the “own slave address” has just been received.

Remark: in the remainder of this chapter, when the phrase “own slave address” is used, it refers to any of the four configured slave addresses after address masking.

11.9.3 Address mask registers, MASK0 to MASK3

The four mask registers each contain seven active bits (7:1). Any bit in these registers which is set to ‘1’ will cause an automatic compare on the corresponding bit of the received address when it is compared to the ADRn register associated with that mask register. In other words, bits in an ADRn register which are masked are not taken into account in determining an address match.

When an address-match interrupt occurs, the processor will have to read the data register (DAT) to determine which received address actually caused the match.

11.9.4 Comparator

The comparator compares the received 7-bit slave address with any of the four configured slave addresses in ADR0 through ADR3 after masking. It also compares the first received 8-bit byte with the General Call address (0x00). If an a match is found, the appropriate status bits are set and an interrupt is requested.

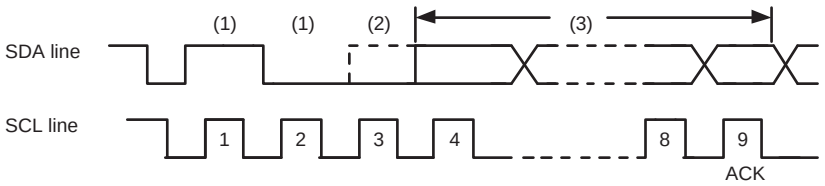
11.9.5 Shift register, DAT

This 8-bit register contains a byte of serial data to be transmitted or a byte which has just been received. Data in DAT is always shifted from right to left; the first bit to be transmitted is the MSB (bit 7) and, after a byte has been received, the first bit of received data is located at the MSB of DAT. While data is being shifted out, data on the bus is simultaneously being shifted in; DAT always contains the last byte present on the bus. Thus, in the event of lost arbitration, the transition from master transmitter to slave receiver is made with the correct data in DAT.

11.9.6 Arbitration and synchronization logic

In the master transmitter mode, the arbitration logic checks that every transmitted logic 1 actually appears as a logic 1 on the I^2C-bus. If another device on the bus overrules a logic 1 and pulls the SDA line low, arbitration is lost, and the I^2C block immediately changes from master transmitter to slave receiver. The I^2C block will continue to output clock pulses (on SCL) until transmission of the current serial byte is complete.

Arbitration may also be lost in the master receiver mode. Loss of arbitration in this mode can only occur while the I^2C block is returning a “not acknowledge: (logic 1) to the bus. Arbitration is lost when another device on the bus pulls this signal low. Since this can occur only at the end of a serial byte, the I^2C block generates no further clock pulses. Figure 20 shows the arbitration procedure.



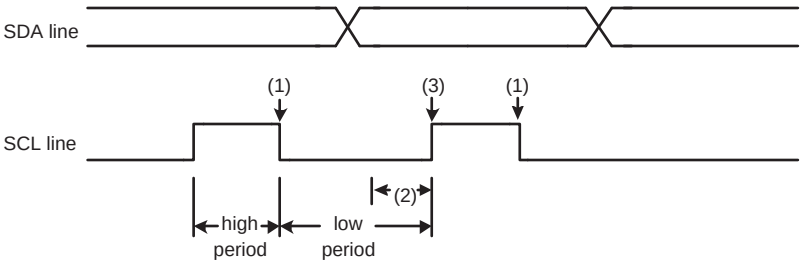
(1) Another device transmits serial data.

(2) Another device overrules a logic (dotted line) transmitted this I2C master by pulling the SDA line low. Arbitration is lost, and this I2C enters Slave Receiver mode.

(3) This I2C is in Slave Receiver mode but still generates clock pulses until the current byte has been transmitted. This I2C will not generate clock pulses for the next byte. Data on SDA originates from the new master once it has won arbitration.

Fig 20. Arbitration procedure

The synchronization logic will synchronize the serial clock generator with the clock pulses on the SCL line from another device. If two or more master devices generate clock pulses, the “mark” duration is determined by the device that generates the shortest “marks,” and the “space” duration is determined by the device that generates the longest “spaces”. Figure 21 shows the synchronization procedure.



(1) Another device pulls the SCL line low before this I2C has timed a complete high time. The other device effectively determines the (shorter) HIGH period.

(2) Another device continues to pull the SCL line low after this I2C has timed a complete low time and released SCL. The I2C clock generator is forced to wait until SCL goes HIGH. The other device effectively determines the (longer) LOW period.

(3) The SCL line is released , and the clock generator begins timing the HIGH time.

Fig 21. Serial clock synchronization

A slave may stretch the space duration to slow down the bus master. The space duration may also be stretched for handshaking purposes. This can be done after each bit or after a complete byte transfer. the I2C block will stretch the SCL space duration after a byte has been transmitted or received and the acknowledge bit has been transferred. The serial interrupt flag (SI) is set, and the stretching continues until the serial interrupt flag is cleared.

11.9.7 Serial clock generator

This programmable clock pulse generator provides the SCL clock pulses when the I2C block is in the master transmitter or master receiver mode. It is switched off when the I2C block is in a slave mode. The I2C output clock frequency and duty cycle is programmable

via the I2C Clock Control Registers. See the description of the CSCLL and CSCLH registers for details. The output clock pulses have a duty cycle as programmed unless the bus is synchronizing with other SCL clock sources as described above.

11.9.8 Timing and control

The timing and control logic generates the timing and control signals for serial byte handling. This logic block provides the shift pulses for DAT, enables the comparator, generates and detects START and STOP conditions, receives and transmits acknowledge bits, controls the master and slave modes, contains interrupt request logic, and monitors the I2C-bus status.

11.9.9 Control register, CONSET and CONCLR

The I2C control register contains bits used to control the following I2C block functions: start and restart of a serial transfer, termination of a serial transfer, bit rate, address recognition, and acknowledgment.

The contents of the I2C control register may be read as CONSET. Writing to CONSET will set bits in the I2C control register that correspond to ones in the value written. Conversely, writing to CONCLR will clear bits in the I2C control register that correspond to ones in the value written.

11.9.10 Status decoder and status register

The status decoder takes all of the internal status bits and compresses them into a 5-bit code. This code is unique for each I2C-bus status. The 5-bit code may be used to generate vector addresses for fast processing of the various service routines. Each service routine processes a particular bus status. There are 26 possible bus states if all four modes of the I2C block are used. The 5-bit status code is latched into the five most significant bits of the status register when the serial interrupt flag is set (by hardware) and remains stable until the interrupt flag is cleared by software. The three least significant bits of the status register are always zero. If the status code is used as a vector to service routines, then the routines are displaced by eight address locations. Eight bytes of code is sufficient for most of the service routines (see the software example in this section).

11.10 Details of I²C operating modes

The four operating modes are:

- Master Transmitter
- Master Receiver
- Slave Receiver
- Slave Transmitter

Data transfers in each mode of operation are shown in Figure 22, Figure 23, Figure 24, Figure 25, and Figure 26. Table 197 lists abbreviations used in these figures when describing the I²C operating modes.

Table 197. Abbreviations used to describe an I²C operation

Abbreviation	Explanation
S	START condition
SLA	7-bit slave address
R	Read bit (HIGH level at SDA)
W	Write bit (LOW level at SDA)
A	Acknowledge bit (LOW level at SDA)
$\overline{A}$	Not acknowledge bit (HIGH level at SDA)
Data	8-bit data byte
P	STOP condition
Sr	Repeated START condition

In Figure 22 to Figure 26, circles are used to indicate when the serial interrupt flag is set. The numbers in the circles show the status code held in the STAT register. At these points, a service routine must be executed to continue or complete the serial transfer. These service routines are not critical since the serial transfer is suspended until the serial interrupt flag is cleared by software.

When a serial interrupt routine is entered, the status code in STAT is used to branch to the appropriate service routine. For each status code, the required software action and details of the following serial transfer are given in tables from Table 200 to Table 204.

11.10.1 Master Transmitter mode

In the master transmitter mode, a number of data bytes are transmitted to a slave receiver (see Figure 22). Before the master transmitter mode can be entered, CON must be initialized as follows:

Table 198. CONSET used to initialize Master Transmitter mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	x	-	-

The I^2C rate must also be configured in the SCLL and SCLH registers. I2EN must be set to logic 1 to enable the I^2C block. If the AA bit is reset, the I^2C block will not acknowledge its own slave address or the General Call address in the event of another device becoming master of the bus. In other words, if AA is reset, the I^2C interface cannot enter a slave mode. STA, STO, and SI must be reset.

The master transmitter mode may now be entered by setting the STA bit. The I^2C logic will now test the I^2C-bus and generate a START condition as soon as the bus becomes free. When a START condition is transmitted, the serial interrupt flag (SI) is set, and the status code in the status register (STAT) will be 0x08. This status code is used by the interrupt service routine to enter the appropriate state service routine that loads DAT with the slave address and the data direction bit (SLA+W). The SI bit in CON must then be reset before the serial transfer can continue.

When the slave address and the direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in STAT are possible. There are 0x18, 0x20, or 0x38 for the master mode and also 0x68, 0x78, or 0xB0 if the slave mode was enabled (AA = logic 1). The appropriate action to be taken for each of these status codes is detailed in Table 200. After a repeated START condition (state 0x10). The I^2C block may switch to the master receiver mode by loading DAT with SLA+R).

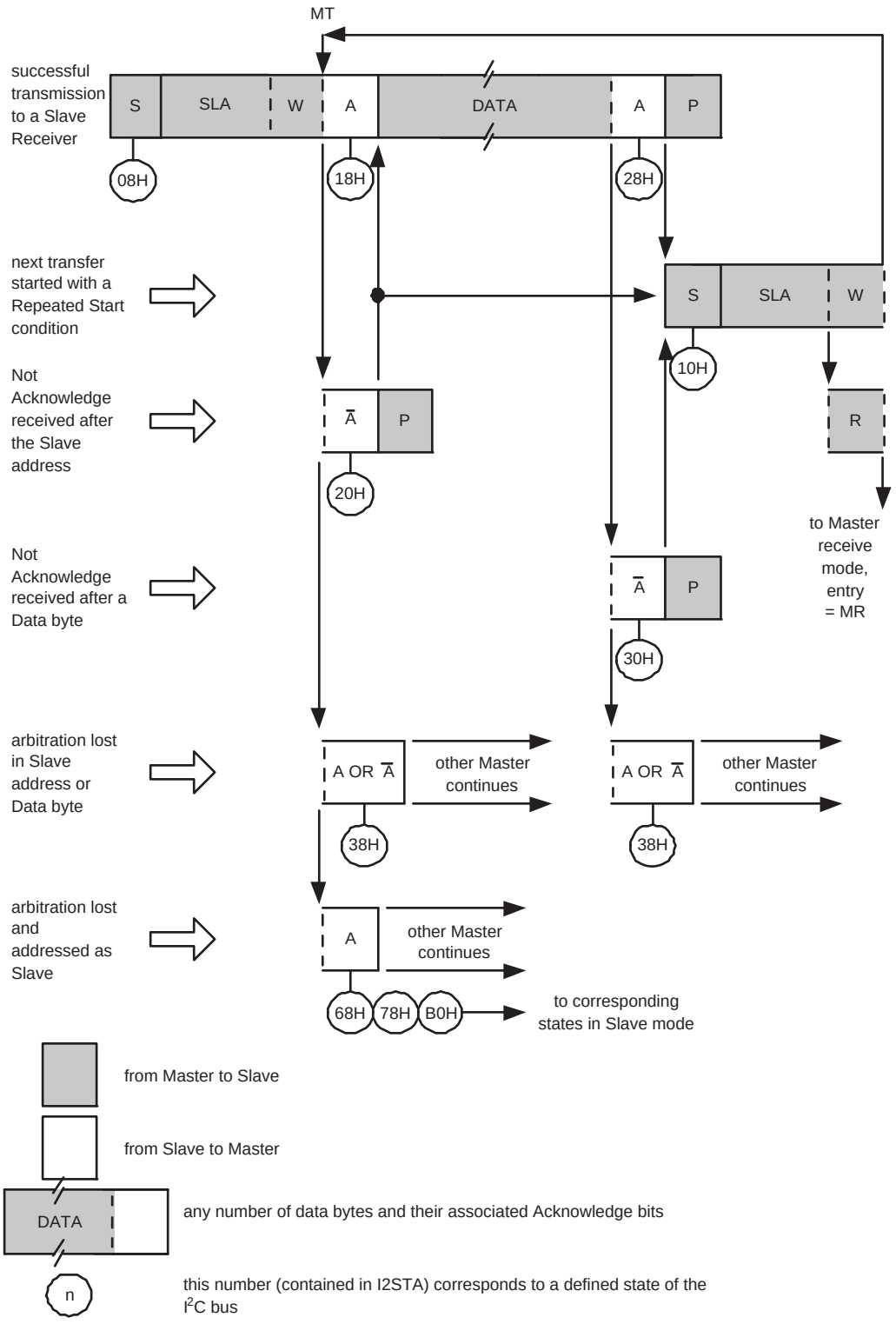


Fig 22. Format and states in the Master Transmitter mode

11.10.2 Master Receiver mode

In the master receiver mode, a number of data bytes are received from a slave transmitter (see Figure 23). The transfer is initialized as in the master transmitter mode. When the START condition has been transmitted, the interrupt service routine must load DAT with the 7-bit slave address and the data direction bit (SLA+R). The SI bit in CON must then be cleared before the serial transfer can continue.

When the slave address and the data direction bit have been transmitted and an acknowledgment bit has been received, the serial interrupt flag (SI) is set again, and a number of status codes in STAT are possible. These are 0x40, 0x48, or 0x38 for the master mode and also 0x68, 0x78, or 0xB0 if the slave mode was enabled (AA = 1). The appropriate action to be taken for each of these status codes is detailed in Table 201. After a repeated START condition (state 0x10), the I^2C block may switch to the master transmitter mode by loading DAT with SLA+W.

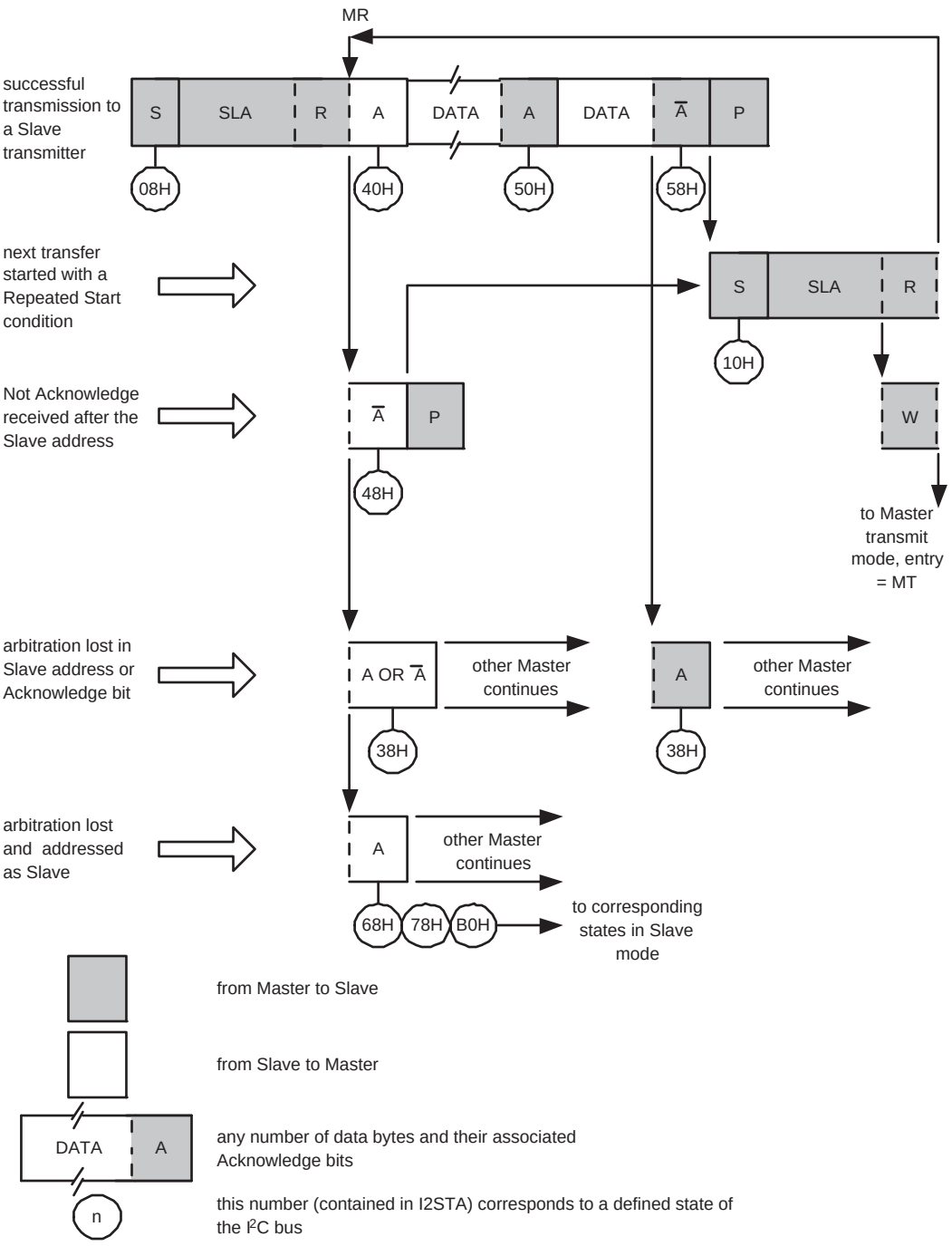


Fig 23. Format and states in the Master Receiver mode

11.10.3 Slave Receiver mode

In the slave receiver mode, a number of data bytes are received from a master transmitter (see Figure 24). To initiate the slave receiver mode, CON register, the ADR registers, and the MASK registers must be configured.

The values on the four ADR registers combined with the values on the four MASK registers determines which address(es) the I^2C block will respond to when slave functions are enabled. See sections Section 11.9.2, Section 11.9.3, Section 11.7.8, and Section 11.7.10 for details.

Table 199. CONSET used to initialize Slave Receiver mode

Bit	7	6	5	4	3	2	1	0
Symbol	-	I2EN	STA	STO	SI	AA	-	-
Value	-	1	0	0	0	1	-	-

The I^2C-bus rate settings do not affect the I^2C block in the slave mode. I2EN must be set to logic 1 to enable the I^2C block. The AA bit must be set to enable the I^2C block to acknowledge its own slave address or the General Call address. STA, STO, and SI must be reset.

When the ADR, MASK, and CON registers have been initialized, the I^2C block waits until it is addressed by its own slave address followed by the data direction bit which must be “0” (W) for the I^2C block to operate in the slave receiver mode. After its own slave address and the W bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from STAT. This status code is used to vector to a state service routine. The appropriate action to be taken for each of these status codes is detailed in Table 202. The slave receiver mode may also be entered if arbitration is lost while the I^2C block is in the master mode (see status 0x68 and 0x78).

If the AA bit is reset during a transfer, the I^2C block will return a not acknowledge (logic 1) to SDA after the next received data byte. While AA is reset, the I^2C block does not respond to its own slave address or a General Call address. However, the I^2C-bus is still monitored and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate the I^2C block from the I^2C-bus.

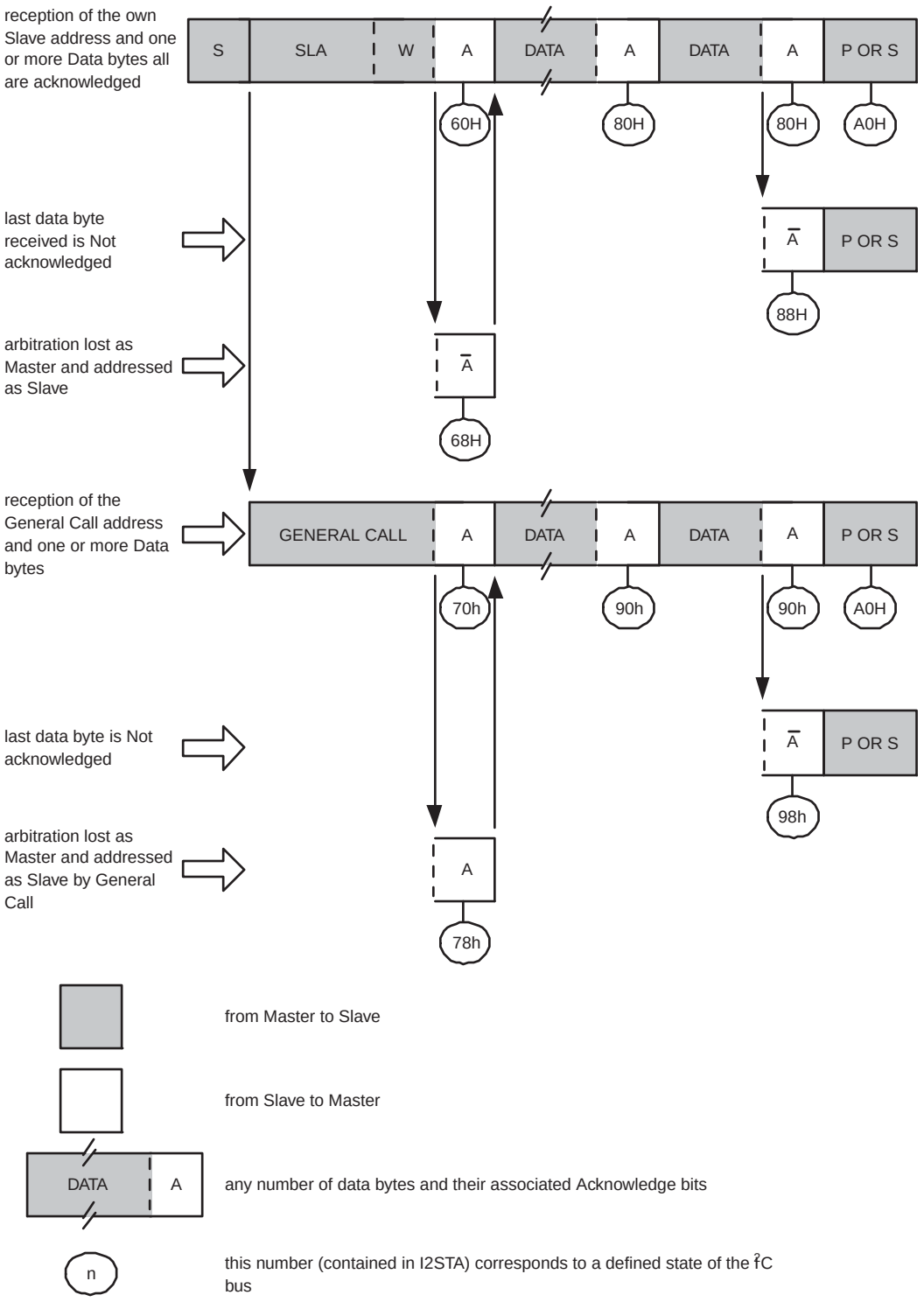


Fig 24. Format and states in the Slave Receiver mode

11.10.4 Slave Transmitter mode

In the slave transmitter mode, a number of data bytes are transmitted to a master receiver (see Figure 25). Data transfer is initialized as in the slave receiver mode. When ADR and CON have been initialized, the I^2C block waits until it is addressed by its own slave address followed by the data direction bit which must be “1” (R) for the I^2C block to operate in the slave transmitter mode. After its own slave address and the R bit have been received, the serial interrupt flag (SI) is set and a valid status code can be read from STAT. This status code is used to vector to a state service routine, and the appropriate action to be taken for each of these status codes is detailed in Table 203. The slave transmitter mode may also be entered if arbitration is lost while the I^2C block is in the master mode (see state 0xB0).

If the AA bit is reset during a transfer, the I^2C block will transmit the last byte of the transfer and enter state 0xC0 or 0xC8. The I^2C block is switched to the not addressed slave mode and will ignore the master receiver if it continues the transfer. Thus the master receiver receives all 1s as serial data. While AA is reset, the I^2C block does not respond to its own slave address or a General Call address. However, the I^2C-bus is still monitored, and address recognition may be resumed at any time by setting AA. This means that the AA bit may be used to temporarily isolate the I^2C block from the I^2C-bus.

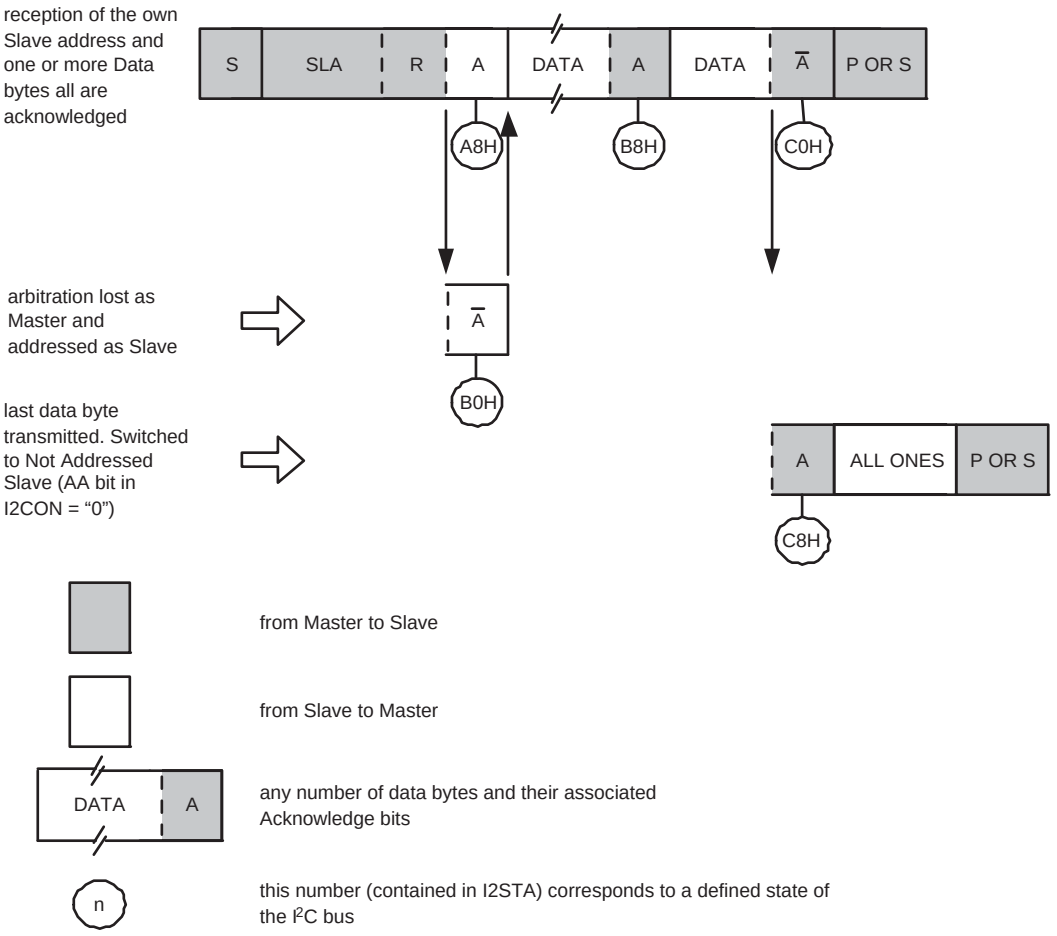


Fig 25. Format and states in the Slave Transmitter mode

11.10.5 Detailed state tables

The following tables show detailed state information for the four I^2C operating modes.

Table 200. Master Transmitter mode

STAT Status Code	Status of the I^2C-bus and hardware	Application software response					Next action taken by I^2C hardware
		To/From DAT	To CON				
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+W; clear STA	X	0	0	X	SLA+W will be transmitted; ACK bit will be received.
0x10	A repeated START condition has been transmitted.	Load SLA+W or	X	0	0	X	As above.
		Load SLA+R; Clear STA	X	0	0	X	SLA+W will be transmitted; the I^2C block will be switched to MST/REC mode.
0x18	SLA+W has been transmitted; ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No DAT action or	1	0	0	X	Repeated START will be transmitted.
		No DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x20	SLA+W has been transmitted; NOT ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No DAT action or	1	0	0	X	Repeated START will be transmitted.
		No DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x28	Data byte in DAT has been transmitted; ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No DAT action or	1	0	0	X	Repeated START will be transmitted.
		No DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x30	Data byte in DAT has been transmitted; NOT ACK has been received.	Load data byte or	0	0	0	X	Data byte will be transmitted; ACK bit will be received.
		No DAT action or	1	0	0	X	Repeated START will be transmitted.
		No DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x38	Arbitration lost in SLA+R/W or Data bytes.	No DAT action or	0	0	0	X	I^2C-bus will be released; not addressed slave will be entered.
		No DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.

Table 201. Master Receiver mode

STAT Status Code	Status of the I^2C-bus and hardware	Application software response					Next action taken by I^2C hardware
		To/From DAT	To CON				
			STA	STO	SI	AA	
0x08	A START condition has been transmitted.	Load SLA+R	X	0	0	X	SLA+R will be transmitted; ACK bit will be received.
0x10	A repeated START condition has been transmitted.	Load SLA+R or	X	0	0	X	As above.
		Load SLA+W	X	0	0	X	SLA+W will be transmitted; the I^2C block will be switched to MST/TRX mode.
0x38	Arbitration lost in NOT ACK bit.	No DAT action or	0	0	0	X	I^2C-bus will be released; the I^2C block will enter a slave mode.
		No DAT action	1	0	0	X	A START condition will be transmitted when the bus becomes free.
0x40	SLA+R has been transmitted; ACK has been received.	No DAT action or	0	0	0	0	Data byte will be received; NOT ACK bit will be returned.
		No DAT action	0	0	0	1	Data byte will be received; ACK bit will be returned.
0x48	SLA+R has been transmitted; NOT ACK has been received.	No DAT action or	1	0	0	X	Repeated START condition will be transmitted.
		No DAT action or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		No DAT action	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.
0x50	Data byte has been received; ACK has been returned.	Read data byte or	0	0	0	0	Data byte will be received; NOT ACK bit will be returned.
		Read data byte	0	0	0	1	Data byte will be received; ACK bit will be returned.
0x58	Data byte has been received; NOT ACK has been returned.	Read data byte or	1	0	0	X	Repeated START condition will be transmitted.
		Read data byte or	0	1	0	X	STOP condition will be transmitted; STO flag will be reset.
		Read data byte	1	1	0	X	STOP condition followed by a START condition will be transmitted; STO flag will be reset.

Table 202. Slave Receiver mode

STAT Status Code	Status of the I²C-bus and hardware	Application software response					Next action taken by I²C hardware
		To/From DAT	To CON				
			STA	STO	SI	AA	
0x60	Own SLA+W has been received; ACK has been returned.	No DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x68	Arbitration lost in SLA+R/W as master; Own SLA+W has been received, ACK returned.	No DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x70	General Call address (0x00) has been received; ACK has been returned.	No DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x78	Arbitration lost in SLA+R/W as master; General Call address has been received, ACK has been returned.	No DAT action or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		No DAT action	X	0	0	1	Data byte will be received and ACK will be returned.
0x80	Previously addressed with own SLA address; DATA has been received; ACK has been returned.	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		Read data byte	X	0	0	1	Data byte will be received and ACK will be returned.
0x88	Previously addressed with own SLA; DATA byte has been received; NOT ACK has been returned.	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		Read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if ADR[0] = logic 1.
		Read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		Read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0x90	Previously addressed with General Call; DATA byte has been received; ACK has been returned.	Read data byte or	X	0	0	0	Data byte will be received and NOT ACK will be returned.
		Read data byte	X	0	0	1	Data byte will be received and ACK will be returned.

Table 202. Slave Receiver mode

STAT Status Code	Status of the I^2C-bus and hardware	Application software response					Next action taken by I^2C hardware
		To/From DAT	To CON				
			STA	STO	SI	AA	
0x98	Previously addressed with General Call; DATA byte has been received; NOT ACK has been returned.	Read data byte or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		Read data byte or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if ADR[0] = logic 1.
		Read data byte or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		Read data byte	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0xA0	A STOP condition or repeated START condition has been received while still addressed as Slave Receiver or Slave Transmitter.	No STDAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		No STDAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if ADR[0] = logic 1.
		No STDAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		No STDAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.

Table 203. Slave Transmitter mode

STAT Status Code	Status of the I^2C-bus and hardware	Application software response					Next action taken by I^2C hardware
		To/From DAT	To CON				
			STA	STO	SI	AA	
0xA8	Own SLA+R has been received; ACK has been returned.	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK will be received.
0xB0	Arbitration lost in SLA+R/W as master; Own SLA+R has been received, ACK has been returned.	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received.
0xB8	Data byte in DAT has been transmitted; ACK has been received.	Load data byte or	X	0	0	0	Last data byte will be transmitted and ACK bit will be received.
		Load data byte	X	0	0	1	Data byte will be transmitted; ACK bit will be received.
0xC0	Data byte in DAT has been transmitted; NOT ACK has been received.	No DAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		No DAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if ADR[0] = logic 1.
		No DAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		No DAT action	1	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if ADR[0] = logic 1. A START condition will be transmitted when the bus becomes free.
0xC8	Last data byte in DAT has been transmitted (AA = 0); ACK has been received.	No DAT action or	0	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address.
		No DAT action or	0	0	0	1	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if ADR[0] = logic 1.
		No DAT action or	1	0	0	0	Switched to not addressed SLV mode; no recognition of own SLA or General Call address. A START condition will be transmitted when the bus becomes free.
		No DAT action	1	0	0	01	Switched to not addressed SLV mode; Own SLA will be recognized; General Call address will be recognized if ADR.0 = logic 1. A START condition will be transmitted when the bus becomes free.

11.10.6 Miscellaneous states

There are two STAT codes that do not correspond to a defined I^2C hardware state (see Table 204). These are discussed below.

11.10.6.1 STAT = 0xF8

This status code indicates that no relevant information is available because the serial interrupt flag, SI, is not yet set. This occurs between other states and when the I^2C block is not involved in a serial transfer.

11.10.6.2 STAT = 0x00

This status code indicates that a bus error has occurred during an I^2C serial transfer. A bus error is caused when a START or STOP condition occurs at an illegal position in the format frame. Examples of such illegal positions are during the serial transfer of an address byte, a data byte, or an acknowledge bit. A bus error may also be caused when external interference disturbs the internal I^2C block signals. When a bus error occurs, SI is set. To recover from a bus error, the STO flag must be set and SI must be cleared. This causes the I^2C block to enter the “not addressed” slave mode (a defined state) and to clear the STO flag (no other bits in CON are affected). The SDA and SCL lines are released (a STOP condition is not transmitted).

Table 204. Miscellaneous States

Status Code (CSTAT)	Status of the I^2C-bus and hardware	Application software response					Next action taken by I^2C hardware
		To/From DAT	To CON				
			STA	STO	SI	AA	
0xF8	No relevant state information available; SI = 0.	No DAT action	No CON action				Wait or proceed current transfer.
0x00	Bus error during MST or selected slave modes, due to an illegal START or STOP condition. State 0x00 can also occur when interference causes the I^2C block to enter an undefined state.	No DAT action	0	1	0	X	Only the internal hardware is affected in the MST or addressed SLV modes. In all cases, the bus is released and the I^2C block is switched to the not addressed SLV mode. STO is reset.

11.10.7 Some special cases

The I^2C hardware has facilities to handle the following special cases that may occur during a serial transfer:

- Simultaneous repeated START conditions from two masters
- Data transfer after loss of arbitration
- Forced access to the I^2C-bus
- I^2C-bus obstructed by a LOW level on SCL or SDA
- Bus error

11.10.7.1 Simultaneous repeated START conditions from two masters

A repeated START condition may be generated in the master transmitter or master receiver modes. A special case occurs if another master simultaneously generates a repeated START condition (see Figure 26). Until this occurs, arbitration is not lost by either master since they were both transmitting the same data.

If the I^2C hardware detects a repeated START condition on the I^2C-bus before generating a repeated START condition itself, it will release the bus, and no interrupt request is generated. If another master frees the bus by generating a STOP condition, the I^2C block will transmit a normal START condition (state 0x08), and a retry of the total serial data transfer can commence.

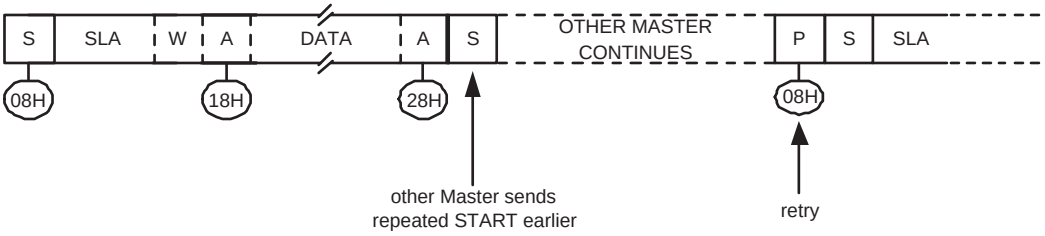


Fig 26. Simultaneous repeated START conditions from two masters

11.10.7.2 Data transfer after loss of arbitration

Arbitration may be lost in the master transmitter and master receiver modes (see Figure 20). Loss of arbitration is indicated by the following states in STAT; 0x38, 0x68, 0x78, and 0xB0 (see Figure 22 and Figure 23).

If the STA flag in CON is set by the routines which service these states, then, if the bus is free again, a START condition (state 0x08) is transmitted without intervention by the CPU, and a retry of the total serial transfer can commence.

11.10.7.3 Forced access to the I^2C-bus

In some applications, it may be possible for an uncontrolled source to cause a bus hang-up. In such situations, the problem may be caused by interference, temporary interruption of the bus or a temporary short-circuit between SDA and SCL.

If an uncontrolled source generates a superfluous START or masks a STOP condition, then the I^2C-bus stays busy indefinitely. If the STA flag is set and bus access is not obtained within a reasonable amount of time, then a forced access to the I^2C-bus is possible. This is achieved by setting the STO flag while the STA flag is still set. No STOP condition is transmitted. The I^2C hardware behaves as if a STOP condition was received and is able to transmit a START condition. The STO flag is cleared by hardware Figure 27.

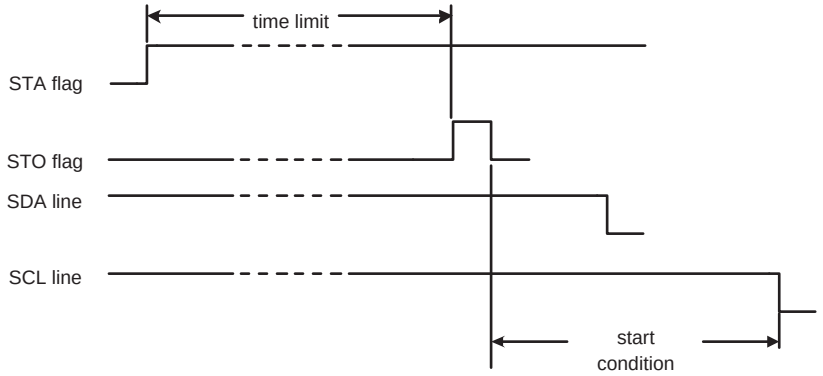
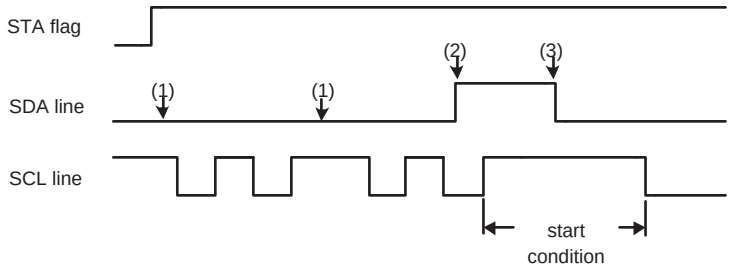


Fig 27. Forced access to a busy I²C-bus

11.10.7.4 I²C-bus obstructed by a LOW level on SCL or SDA

An I²C-bus hang-up can occur if either the SDA or SCL line is held LOW by any device on the bus. If the SCL line is obstructed (pulled LOW) by a device on the bus, no further serial transfer is possible, and the problem must be resolved by the device that is pulling the SCL bus line LOW.

Typically, the SDA line may be obstructed by another device on the bus that has become out of synchronization with the current bus master by either missing a clock, or by sensing a noise pulse as a clock. In this case, the problem can be solved by transmitting additional clock pulses on the SCL line Figure 28. The I²C interface does not include a dedicated time-out timer to detect an obstructed bus, but this can be implemented using another timer in the system. When detected, software can force clocks (up to 9 may be required) on SCL until SDA is released by the offending device. At that point, the slave may still be out of synchronization, so a START should be generated to insure that all I²C peripherals are synchronized.



(1) Unsuccessful attempt to send a START condition.

(2) SDA line is released.

(3) Successful attempt to send a START condition. State 08H is entered.

Fig 28. Recovering from a bus obstruction caused by a LOW level on SDA

11.10.7.5 Bus error

A bus error occurs when a START or STOP condition is detected at an illegal position in the format frame. Examples of illegal positions are during the serial transfer of an address byte, a data bit, or an acknowledge bit.

The I2C hardware only reacts to a bus error when it is involved in a serial transfer either as a master or an addressed slave. When a bus error is detected, the I2C block immediately switches to the not addressed slave mode, releases the SDA and SCL lines, sets the interrupt flag, and loads the status register with 0x00. This status code may be used to vector to a state service routine which either attempts the aborted serial transfer again or simply recovers from the error condition as shown in Table 204.

11.10.8 I2C state service routines

This section provides examples of operations that must be performed by various I2C state service routines. This includes:

- Initialization of the I2C block after a Reset.
- I2C Interrupt Service
- The 26 state service routines providing support for all four I2C operating modes.

11.10.8.1 Initialization

In the initialization example, the I2C block is enabled for both master and slave modes. For each mode, a buffer is used for transmission and reception. The initialization routine performs the following functions:

- The ADR registers and MASK registers are loaded with values to configure the part's own slave address(es) and the General Call bit (GC)
- The I2C interrupt enable and interrupt priority bits are set
- The slave mode is enabled by simultaneously setting the EN and AA bits in CON and the serial clock frequency (for master modes) is defined by loading the SCLH and SCLL registers. The master routines must be started in the main program.

The I2C hardware now begins checking the I2C-bus for its own slave address and General Call. If the General Call or the own slave address is detected, an interrupt is requested and STAT is loaded with the appropriate state information.

11.10.8.2 I2C interrupt service

When the I2C interrupt is entered, STAT contains a status code which identifies one of the 26 state services to be executed.

11.10.8.3 The state service routines

Each state routine is part of the I2C interrupt routine and handles one of the 26 states.

11.10.8.4 Adapting state services to an application

The state service examples show the typical actions that must be performed in response to the 26 I2C state codes. If one or more of the four I2C operating modes are not used, the associated state services can be omitted, as long as care is taken that the those states can never occur.

In an application, it may be desirable to implement some kind of time-out during I2C operations, in order to trap an inoperative bus or a lost service routine.

11.11 Software example

11.11.1 Initialization routine

Example to initialize I^2C Interface as a Slave and/or Master.

1. Load ADR with own Slave Address, enable General Call recognition if needed.
2. Enable I^2C interrupt.
3. Write 0x44 to CONSET to set the EN and AA bits, enabling Slave functions. For Master only functions, write 0x40 to CONSET.

11.11.2 Start Master Transmit function

Begin a Master Transmit operation by setting up the buffer, pointer, and data count, then initiating a START.

1. Initialize Master data counter.
2. Set up the Slave Address to which data will be transmitted, and add the Write bit.
3. Write 0x20 to CONSET to set the STA bit.
4. Set up data to be transmitted in Master Transmit buffer.
5. Initialize the Master data counter to match the length of the message being sent.
6. Exit

11.11.3 Start Master Receive function

Begin a Master Receive operation by setting up the buffer, pointer, and data count, then initiating a START.

1. Initialize Master data counter.
2. Set up the Slave Address to which data will be transmitted, and add the Read bit.
3. Write 0x20 to CONSET to set the STA bit.
4. Set up the Master Receive buffer.
5. Initialize the Master data counter to match the length of the message to be received.
6. Exit

11.11.4 I^2C interrupt routine

Determine the I^2C state and which state routine will be used to handle it.

1. Read the I^2C status from STA.
2. Use the status value to branch to one of 26 possible state routines.

11.11.5 Non mode specific states

11.11.5.1 State: 0x00

Bus Error. Enter not addressed Slave mode and release bus.

1. Write 0x14 to CONSET to set the STO and AA bits.

2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

11.11.5.2 Master States

State 08 and State 10 are for both Master Transmit and Master Receive modes. The R/W bit decides whether the next state is within Master Transmit mode or Master Receive mode.

11.11.5.3 State: 0x08

A START condition has been transmitted. The Slave Address + R/W bit will be transmitted, an ACK bit will be received.

1. Write Slave Address with R/W bit to DAT.
2. Write 0x04 to CONSET to set the AA bit.
3. Write 0x08 to CONCLR to clear the SI flag.
4. Set up Master Transmit mode data buffer.
5. Set up Master Receive mode data buffer.
6. Initialize Master data counter.
7. Exit

11.11.5.4 State: 0x10

A Repeated START condition has been transmitted. The Slave Address + R/W bit will be transmitted, an ACK bit will be received.

1. Write Slave Address with R/W bit to DAT.
2. Write 0x04 to CONSET to set the AA bit.
3. Write 0x08 to CONCLR to clear the SI flag.
4. Set up Master Transmit mode data buffer.
5. Set up Master Receive mode data buffer.
6. Initialize Master data counter.
7. Exit

11.11.6 Master Transmitter states

11.11.6.1 State: 0x18

Previous state was State 8 or State 10, Slave Address + Write has been transmitted, ACK has been received. The first data byte will be transmitted, an ACK bit will be received.

1. Load DAT with first data byte from Master Transmit buffer.
2. Write 0x04 to CONSET to set the AA bit.
3. Write 0x08 to CONCLR to clear the SI flag.
4. Increment Master Transmit buffer pointer.
5. Exit

11.11.6.2 State: 0x20

Slave Address + Write has been transmitted, NOT ACK has been received. A STOP condition will be transmitted.

1. Write 0x14 to CONSET to set the STO and AA bits.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

11.11.6.3 State: 0x28

Data has been transmitted, ACK has been received. If the transmitted data was the last data byte then transmit a STOP condition, otherwise transmit the next data byte.

1. Decrement the Master data counter, skip to step 5 if not the last data byte.
2. Write 0x14 to CONSET to set the STO and AA bits.
3. Write 0x08 to CONCLR to clear the SI flag.
4. Exit
5. Load DAT with next data byte from Master Transmit buffer.
6. Write 0x04 to CONSET to set the AA bit.
7. Write 0x08 to CONCLR to clear the SI flag.
8. Increment Master Transmit buffer pointer
9. Exit

11.11.6.4 State: 0x30

Data has been transmitted, NOT ACK received. A STOP condition will be transmitted.

1. Write 0x14 to CONSET to set the STO and AA bits.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

11.11.6.5 State: 0x38

Arbitration has been lost during Slave Address + Write or data. The bus has been released and not addressed Slave mode is entered. A new START condition will be transmitted when the bus is free again.

1. Write 0x24 to CONSET to set the STA and AA bits.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

11.11.7 Master Receive states

11.11.7.1 State: 0x40

Previous state was State 08 or State 10. Slave Address + Read has been transmitted, ACK has been received. Data will be received and ACK returned.

1. Write 0x04 to CONSET to set the AA bit.
2. Write 0x08 to CONCLR to clear the SI flag.

3. Exit

11.11.7.2 State: 0x48

Slave Address + Read has been transmitted, NOT ACK has been received. A STOP condition will be transmitted.

1. Write 0x14 to CONSET to set the STO and AA bits.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

11.11.7.3 State: 0x50

Data has been received, ACK has been returned. Data will be read from DAT. Additional data will be received. If this is the last data byte then NOT ACK will be returned, otherwise ACK will be returned.

1. Read data byte from DAT into Master Receive buffer.
2. Decrement the Master data counter, skip to step 5 if not the last data byte.
3. Write 0x0C to CONCLR to clear the SI flag and the AA bit.
4. Exit
5. Write 0x04 to CONSET to set the AA bit.
6. Write 0x08 to CONCLR to clear the SI flag.
7. Increment Master Receive buffer pointer
8. Exit

11.11.7.4 State: 0x58

Data has been received, NOT ACK has been returned. Data will be read from DAT. A STOP condition will be transmitted.

1. Read data byte from DAT into Master Receive buffer.
2. Write 0x14 to CONSET to set the STO and AA bits.
3. Write 0x08 to CONCLR to clear the SI flag.
4. Exit

11.11.8 Slave Receiver states

11.11.8.1 State: 0x60

Own Slave Address + Write has been received, ACK has been returned. Data will be received and ACK returned.

1. Write 0x04 to CONSET to set the AA bit.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

11.11.8.2 State: 0x68

Arbitration has been lost in Slave Address and R/W bit as bus Master. Own Slave Address + Write has been received, ACK has been returned. Data will be received and ACK will be returned. STA is set to restart Master mode after the bus is free again.

1. Write 0x24 to CONSET to set the STA and AA bits.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit.

11.11.8.3 State: 0x70

General call has been received, ACK has been returned. Data will be received and ACK returned.

1. Write 0x04 to CONSET to set the AA bit.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

11.11.8.4 State: 0x78

Arbitration has been lost in Slave Address + R/W bit as bus Master. General call has been received and ACK has been returned. Data will be received and ACK returned. STA is set to restart Master mode after the bus is free again.

1. Write 0x24 to CONSET to set the STA and AA bits.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Set up Slave Receive mode data buffer.
4. Initialize Slave data counter.
5. Exit

11.11.8.5 State: 0x80

Previously addressed with own Slave Address. Data has been received and ACK has been returned. Additional data will be read.

1. Read data byte from DAT into the Slave Receive buffer.
2. Decrement the Slave data counter, skip to step 5 if not the last data byte.
3. Write 0x0C to CONCLR to clear the SI flag and the AA bit.
4. Exit.
5. Write 0x04 to CONSET to set the AA bit.
6. Write 0x08 to CONCLR to clear the SI flag.
7. Increment Slave Receive buffer pointer.
8. Exit

11.11.8.6 State: 0x88

Previously addressed with own Slave Address. Data has been received and NOT ACK has been returned. Received data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to CONSET to set the AA bit.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

11.11.8.7 State: 0x90

Previously addressed with General Call. Data has been received, ACK has been returned. Received data will be saved. Only the first data byte will be received with ACK. Additional data will be received with NOT ACK.

1. Read data byte from DAT into the Slave Receive buffer.
2. Write 0x0C to CONCLR to clear the SI flag and the AA bit.
3. Exit

11.11.8.8 State: 0x98

Previously addressed with General Call. Data has been received, NOT ACK has been returned. Received data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to CONSET to set the AA bit.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

11.11.8.9 State: 0xA0

A STOP condition or Repeated START has been received, while still addressed as a Slave. Data will not be saved. Not addressed Slave mode is entered.

1. Write 0x04 to CONSET to set the AA bit.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

11.11.9 Slave Transmitter states

11.11.9.1 State: 0xA8

Own Slave Address + Read has been received, ACK has been returned. Data will be transmitted, ACK bit will be received.

1. Load DAT from Slave Transmit buffer with first data byte.
2. Write 0x04 to CONSET to set the AA bit.
3. Write 0x08 to CONCLR to clear the SI flag.
4. Set up Slave Transmit mode data buffer.
5. Increment Slave Transmit buffer pointer.
6. Exit

11.11.9.2 State: 0xB0

Arbitration lost in Slave Address and R/W bit as bus Master. Own Slave Address + Read has been received, ACK has been returned. Data will be transmitted, ACK bit will be received. STA is set to restart Master mode after the bus is free again.

1. Load DAT from Slave Transmit buffer with first data byte.
2. Write 0x24 to CONSET to set the STA and AA bits.
3. Write 0x08 to CONCLR to clear the SI flag.
4. Set up Slave Transmit mode data buffer.
5. Increment Slave Transmit buffer pointer.
6. Exit

11.11.9.3 State: 0xB8

Data has been transmitted, ACK has been received. Data will be transmitted, ACK bit will be received.

1. Load DAT from Slave Transmit buffer with data byte.
2. Write 0x04 to CONSET to set the AA bit.
3. Write 0x08 to CONCLR to clear the SI flag.
4. Increment Slave Transmit buffer pointer.
5. Exit

11.11.9.4 State: 0xC0

Data has been transmitted, NOT ACK has been received. Not addressed Slave mode is entered.

1. Write 0x04 to CONSET to set the AA bit.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit.

11.11.9.5 State: 0xC8

The last data byte has been transmitted, ACK has been received. Not addressed Slave mode is entered.

1. Write 0x04 to CONSET to set the AA bit.
2. Write 0x08 to CONCLR to clear the SI flag.
3. Exit

12.1 How to read this chapter

The SSP controller is available on all LPC122x parts.

12.2 Basic configuration

Clocks and power to the SSP are controlled by:

1. The SYSAHBCLKCTRL register (see Table 21).
2. The SSP_PCLK which is enabled in the SSP clock divider register (see Table 22). This clock is used by the SSP prescale divider.

The SSP_PCLK clock can be disabled in the SSP registers (see Table 22). The SSP block can be disabled through the System AHB clock control register bit 11 (see Table 21) for power savings.

12.3 Features

- Compatible with Motorola SPI, 4-wire TI SSI, and National Semiconductor Microwire buses
- Synchronous Serial Communication
- Master or slave operation
- Eight-frame FIFOs for both transmit and receive
- 4-bit to 16-bit frame

12.4 Description

The SSP is a Synchronous Serial Port (SSP) controller capable of operation on a SPI, 4-wire SSI, or Microwire bus. It can interact with multiple masters and slaves on the bus. Only a single master and a single slave can communicate on the bus during a given data transfer. Data transfers are in principle full duplex, with frames of 4 bit to 16 bit of data flowing from the master to the slave and from the slave to the master. In practice it is often the case that only one of these data flows carries meaningful data.

12.5 Pin description

Table 205. SSP pin descriptions

Pin Name	Type	Interface pin name/function			Pin Description
		SPI	SSI	Microwire	
SCK	I/O	SCK	CLK	SK	**Serial Clock.** SCK/CLK/SK is a clock signal used to synchronize the transfer of data. It is driven by the master and received by the slave. When SPI interface is used, the clock is programmable to be active-high or active-low, otherwise it is always active-high. SCK only switches during a data transfer. Any other time, the SSP interface either holds it in its inactive state or does not drive it (leaves it in high-impedance state).
SSEL	I/O	SSEL	FS	CS	**Frame Sync/Slave Select.** When the SSP interface is a bus master, it drives this signal to an active state before the start of serial data and then releases it to an inactive state after the data has been sent.The active state of this signal can be high or low depending upon the selected bus and mode. When the SSP interface is a bus slave, this signal qualifies the presence of data from the Master according to the protocol in use. When there is just one bus master and one bus slave, the Frame Sync or Slave Select signal from the Master can be connected directly to the slave's corresponding input. When there is more than one slave on the bus, further qualification of their Frame Select/Slave Select inputs will typically be necessary to prevent more than one slave from responding to a transfer.
MISO	I/O	MISO	DR(M) DX(S)	SI(M) SO(S)	**Master In Slave Out.** The MISO signal transfers serial data from the slave to the master. When the SSP is a slave, serial data is output on this signal. When the SSP is a master, it clocks in serial data from this signal. When the SSP is a slave and is not selected by FS/SSEL, it does not drive this signal (leaves it in high-impedance state).
MOSI	I/O	MOSI	DX(M) DR(S)	SO(M) SI(S)	**Master Out Slave In.** The MOSI signal transfers serial data from the master to the slave. When the SSP is a master, it outputs serial data on this signal. When the SSP is a slave, it clocks in serial data from this signal.

12.6 Register description

The register addresses of the SSP controller are shown in Table 206.

Table 206. Register overview: SSP (base address 0x4004 0000)

Name	Access	Address offset	Description	Reset value[1]
CR0	R/W	0x000	Control Register 0. Selects the serial clock rate, bus type, and data size.	0
CR1	R/W	0x004	Control Register 1. Selects master/slave and other modes.	0
DR	R/W	0x008	Data Register. Writes fill the transmit FIFO, and reads empty the receive FIFO.	0
SR	RO	0x00C	Status Register	0x0000 0003
CPSR	R/W	0x010	Clock Prescale Register	0
IMSC	R/W	0x014	Interrupt Mask Set and Clear Register	0
RIS	RO	0x018	Raw Interrupt Status Register	-
MIS	RO	0x01C	Masked Interrupt Status Register	0x0000 0008
ICR	WO	0x020	SSPICR Interrupt Clear Register	NA
DMACR	R/W	0x024	DMA Control Register	0

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

12.6.1 SSP Control Register 0

This register controls the basic operation of the SSP controller.

Table 207. SSP Control Register 0 (CR0 - address 0x4004 0000) bit description

Bit	Symbol	Value	Description	Reset value
3:0	DSS		Data Size Select. This field controls the number of bits transferred in each frame. Values 0000-0010 are not supported and should not be used.	0000
		0x3	4-bit transfer	
		0x4	5-bit transfer	
		0x5	6-bit transfer	
		0x6	7-bit transfer	
		0x7	8-bit transfer	
		0x8	9-bit transfer	
		0x9	10-bit transfer	
		0xA	11-bit transfer	
		0xB	12-bit transfer	
		0xC	13-bit transfer	
		0xD	14-bit transfer	
		0xE	15-bit transfer	
		0xF	16-bit transfer	
5:4	FRF		Frame Format.	00
		0x0	SPI	
		0x1	TI	
		0x2	Microwire	
		0x3	This combination is not supported and should not be used.	

Table 207. SSP Control Register 0 (CR0 - address 0x4004 0000) bit description

Bit	Symbol	Value	Description	Reset value
6	CPOL		Clock Out Polarity. This bit is only used in SPI mode.	0
		0	SSP controller maintains the bus clock low between frames.	
		1	SSP controller maintains the bus clock high between frames.	
7	CPHA		Clock Out Phase. This bit is only used in SPI mode.	0
		0	SSP controller captures serial data on the first clock transition of the frame, that is, the transition **away from** the inter-frame state of the clock line.	
		1	SSP controller captures serial data on the second clock transition of the frame, that is, the transition **back to** the inter-frame state of the clock line.	
15:8	SCR		Serial Clock Rate. The number of prescaler-output clocks per bit on the bus, minus one. Given that CPSDVSR is the prescale divider, and the APB clock PCLK clocks the prescaler, the bit frequency is PCLK / (CPSDVSR × [SCR+1]).	0x00
31:16	-		Reserved.	-

12.6.2 SSP Control Register 1

This register controls certain aspects of the operation of the SSP controller.

Table 208. SSP Control Register 1 (CR1 - address 0x4004 0004) bit description

Bit	Symbol	Value	Description	Reset value
0	LBM		Loop Back Mode.	0
		0	During normal operation.	
		1	Serial input is taken from the serial output (MOSI or MISO) rather than the serial input pin (MISO or MOSI respectively).	
1	SSE		SSP Enable.	0
		0	The SSP controller is disabled.	
		1	The SSP controller will interact with other devices on the serial bus. Software should write the appropriate control information to the other SSP registers and interrupt controller registers, before setting this bit.	
2	MS		Master/Slave Mode.This bit can only be written when the SSE bit is 0.	0
		0	The SSP controller acts as a master on the bus, driving the SCLK, MOSI, and SSEL lines and receiving the MISO line.	
		1	The SSP controller acts as a slave on the bus, driving MISO line and receiving SCLK, MOSI, and SSEL lines.	
3	SOD		Slave Output Disable. This bit is relevant only in slave mode (MS = 1). If it is 1, this blocks this SSP controller from driving the transmit data line (MISO).	0
31:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

12.6.3 SSP Data Register

Software can write data to be transmitted to this register, and read data that has been received.

Table 209. SSP Data Register (DR - address 0x4004 0008) bit description

Bit	Symbol	Description	Reset value
15:0	DATA	**Write:** software can write data to be sent in a future frame to this register whenever the TNF bit in the Status register is 1, indicating that the Tx FIFO is not full. If the Tx FIFO was previously empty and the SSP controller is not busy on the bus, transmission of the data will begin immediately. Otherwise the data written to this register will be sent as soon as all previous data has been sent (and received). If the data length is less than 16 bit, software must right-justify the data written to this register. **Read:** software can read data from this register whenever the RNE bit in the Status register is 1, indicating that the Rx FIFO is not empty. When software reads this register, the SSP controller returns data from the least recent frame in the Rx FIFO. If the data length is less than 16 bit, the data is right-justified in this field with higher order bits filled with 0s.	0x0000
31:16	-	Reserved	-

12.6.4 SSP Status Register

This read-only register reflects the current status of the SSP controller.

Table 210. SSP Status Register (SR - address 0x4004 000C bit description

Bit	Symbol	Description	Reset value
0	TFE	Transmit FIFO Empty. This bit is 1 is the Transmit FIFO is empty, 0 if not.	1
1	TNF	Transmit FIFO Not Full. This bit is 0 if the Tx FIFO is full, 1 if not.	1
2	RNE	Receive FIFO Not Empty. This bit is 0 if the Receive FIFO is empty, 1 if not.	0
3	RFF	Receive FIFO Full. This bit is 1 if the Receive FIFO is full, 0 if not.	0
4	BSY	Busy. This bit is 0 if the SSP controller is idle, or 1 if it is currently sending/receiving a frame and/or the Tx FIFO is not empty.	0
31:5	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

12.6.5 SSP Clock Prescale Register

This register controls the factor by which the Prescaler divides the SSP peripheral clock SSP_PCLK to yield the prescaler clock that is, in turn, divided by the SCR factor in CR0, to determine the bit clock.

Table 211. SSP Clock Prescale Register (CPSR - address 0x4004 0010) bit description

Bit	Symbol	Description	Reset value
7:0	CPSDVSR	This even value between 2 and 254, by which SSP_PCLK is divided to yield the prescaler output clock. Bit 0 always reads as 0.	0
31:8	-	Reserved	-

Important: the CPSR value must be properly initialized or the SSP controller will not be able to transmit data correctly.

In Slave mode, the SSP clock rate provided by the master must not exceed 1/12 of the SSP peripheral clock selected in Table 22. The content of the CPSR register is not relevant.

In master mode, $CPSDVSR_{min}$ = 2 or larger (even numbers only).

12.6.6 SSP Interrupt Mask Set/Clear Register (IMSC - 0x4004 0014)

This register controls whether each of the four possible interrupt conditions in the SSP controller are enabled. Note that ARM uses the word “masked” in the opposite sense from classic computer terminology, in which “masked” meant “disabled”. ARM uses the word “masked” to mean “enabled”. To avoid confusion we will not use the word “masked”.

Table 212. SSP Interrupt Mask Set/Clear register (IMSC - address 0x4004 0014) bit description

Bit	Symbol	Description	Reset value
0	RORIM	Software should set this bit to enable interrupt when a Receive Overrun occurs, that is, when the Rx FIFO is full and another frame is completely received. The ARM spec implies that the preceding frame data is overwritten by the new frame data when this occurs.	0
1	RTIM	Software should set this bit to enable interrupt when a Receive Time-out condition occurs. A Receive Time-out occurs when the Rx FIFO is not empty, and no has not been read for a “time-out period”. The time-out period is the same for master and slave modes and is determined by the SSP bit rate: 32 bits at PCLK / (CPSDVSR × [SCR+1]).	0
2	RXIM	Software should set this bit to enable interrupt when the Rx FIFO is at least half full.	0
3	TXIM	Software should set this bit to enable interrupt when the Tx FIFO is at least half empty.	0
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

12.6.7 SSP Raw Interrupt Status Register

This read-only register contains a 1 for each interrupt condition that is asserted, regardless of whether or not the interrupt is enabled in the IMSC.

Table 213. SSP Raw Interrupt Status register (RIS - address 0x4004 0018) bit description

Bit	Symbol	Description	Reset value
0	RORRIS	This bit is 1 if another frame was completely received while the RxFIFO was full. The ARM spec implies that the preceding frame data is overwritten by the new frame data when this occurs.	0
1	RTRIS	This bit is 1 if the Rx FIFO is not empty, and has not been read for a “time-out period”. The time-out period is the same for master and slave modes and is determined by the SSP bit rate: 32 bits at PCLK / (CPSDVSR × [SCR+1]).	0
2	RXRIS	This bit is 1 if the Rx FIFO is at least half full.	0
3	TXRIS	This bit is 1 if the Tx FIFO is at least half empty.	1
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

12.6.8 SSP Masked Interrupt Status Register

This read-only register contains a 1 for each interrupt condition that is asserted and enabled in the IMSC. When an SSP interrupt occurs, the interrupt service routine should read this register to determine the cause(s) of the interrupt.

Table 214. SSP Masked Interrupt Status register (MIS -address 0x4004 001C) bit description

Bit	Symbol	Description	Reset value
0	RORMIS	This bit is 1 if another frame was completely received while the RxFIFO was full, and this interrupt is enabled.	0
1	RTMIS	This bit is 1 if the Rx FIFO is not empty, has not been read for a “time-out period”, and this interrupt is enabled. The time-out period is the same for master and slave modes and is determined by the SSP bit rate: 32 bits at PCLK / (CPSDVSR × [SCR+1]).	0
2	RXMIS	This bit is 1 if the Rx FIFO is at least half full, and this interrupt is enabled.	0
3	TXMIS	This bit is 1 if the Tx FIFO is at least half empty, and this interrupt is enabled.	0
31:4	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

12.6.9 SSP Interrupt Clear Register

Software can write one or more one(s) to this write-only register, to clear the corresponding interrupt condition(s) in the SSP controller. Note that the other two interrupt conditions can be cleared by writing or reading the appropriate FIFO, or disabled by clearing the corresponding bit in IMSC.

Table 215. SSP interrupt Clear Register (ICR - address 0x4004 0020) bit description

Bit	Symbol	Description	Reset value
0	RORIC	Writing a 1 to this bit clears the “frame was received when RxFIFO was full” interrupt.	NA
1	RTIC	Writing a 1 to this bit clears the “Rx FIFO was not empty and has not been read for a time-out period” interrupt. The time-out period is the same for master and slave modes and is determined by the SSP bit rate: 32 bits at PCLK / (CPSDVSR × [SCR+1]).	NA
31:2	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

12.6.10 SSP DMA Control Register

The DMACR register is the DMA control register. It is a read/write register.

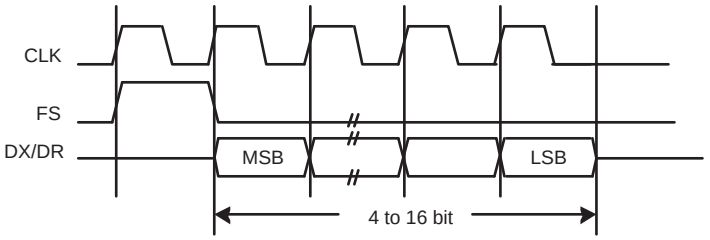
Table 216. SSP DMA Control Register (DMACR - address 0x4004 0024) bit description

Bit	Symbol	Value	Description	Reset value
0	RXDMAE		Receive DMA Enable	0
		0	Receive DMA disabled.	
		1	DMA for the receive FIFO is enabled.	
1	TXDMAE		Transmit DMA Enable	0
		0	Transmit DMA disabled.	
		1	DMA for the transmit FIFO is enabled.	
31:2	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

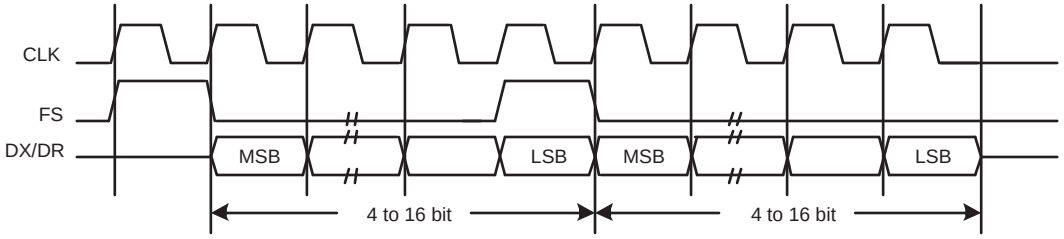
12.7 Functional description

12.7.1 Texas Instruments synchronous serial frame format

Figure 29 shows the 4-wire Texas Instruments synchronous serial frame format supported by the SSP module.



a. Single frame transfer



b. Continuous/back-to-back frames transfer

Fig 29. Texas Instruments Synchronous Serial Frame Format: a) Single and b) Continuous/back-to-back Two Frames Transfer

For device configured as a master in this mode, CLK and FS are forced LOW, and the transmit data line DX is in 3-state mode whenever the SSP is idle. Once the bottom entry of the transmit FIFO contains data, FS is pulsed HIGH for one CLK period. The value to be transmitted is also transferred from the transmit FIFO to the serial shift register of the transmit logic. On the next rising edge of CLK, the MSB of the 4-bit to 16-bit data frame is shifted out on the DX pin. Likewise, the MSB of the received data is shifted onto the DR pin by the off-chip serial slave device.

Both the SSP and the off-chip serial slave device then clock each data bit into their serial shifter on the falling edge of each CLK. The received data is transferred from the serial shifter to the receive FIFO on the first rising edge of CLK after the LSB has been latched.

12.7.2 SPI frame format

The SPI interface is a four-wire interface where the SSEL signal behaves as a slave select. The main feature of the SPI format is that the inactive state and phase of the SCK signal are programmable through the CPOL and CPHA bits within the SSPCR0 control register.

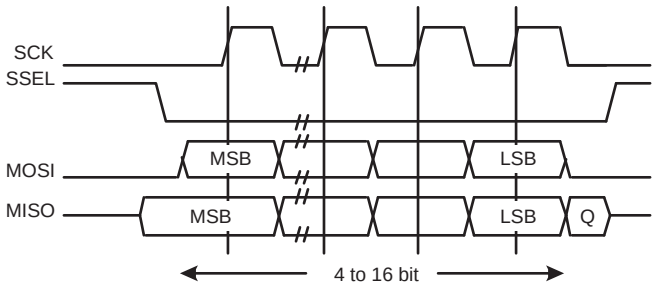
12.7.2.1 Clock Polarity (CPOL) and Phase (CPHA) control

When the CPOL clock polarity control bit is LOW, it produces a steady state low value on the SCK pin. If the CPOL clock polarity control bit is HIGH, a steady state high value is placed on the CLK pin when data is not being transferred.

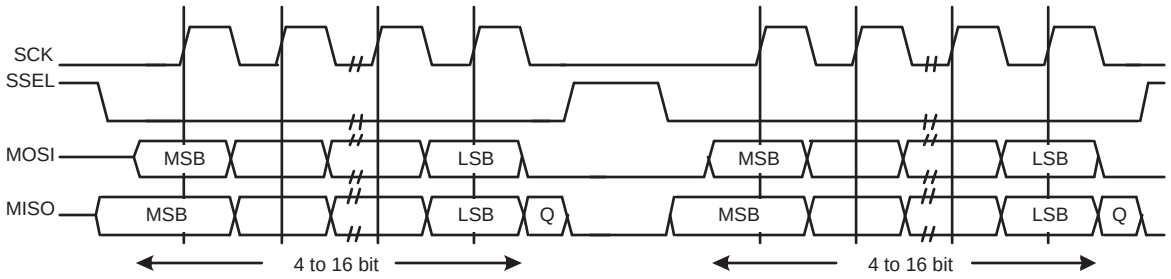
The CPHA control bit selects the clock edge that captures data and allows it to change state. It has the most impact on the first bit transmitted by either allowing or not allowing a clock transition before the first data capture edge. When the CPHA phase control bit is LOW, data is captured on the first clock edge transition. If the CPHA clock phase control bit is HIGH, data is captured on the second clock edge transition.

12.7.2.2 SPI format with CPOL=0,CPHA=0

Single and continuous transmission signal sequences for SPI format with CPOL = 0, CPHA = 0 are shown in Figure 30.



a. Single transfer with CPOL=0 and CPHA=0



b. Continuous transfer with CPOL=0 and CPHA=0

Fig 30. SPI frame format with CPOL=0 and CPHA=0 (a) Single and b) Continuous Transfer)

In this configuration, during idle periods:

- The CLK signal is forced LOW.
- SSEL is forced HIGH.
- The transmit MOSI/MISO pad is in high impedance.

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSEL master signal being driven LOW. This causes slave data to be enabled onto the MISO input line of the master. Master's MOSI is enabled.

One half SCK period later, valid master data is transferred to the MOSI pin. Now that both the master and slave data have been set, the SCK master clock pin goes HIGH after one further half SCK period.

The data is now captured on the rising and propagated on the falling edges of the SCK signal.

In the case of a single word transmission, after all bits of the data word have been transferred, the SSEL line is returned to its idle HIGH state one SCK period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSEL signal must be pulsed HIGH between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the CPHA bit is logic zero. Therefore the master device must raise the SSEL pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSEL pin is returned to its idle state one SCK period after the last bit has been captured.

12.7.2.3 SPI format with CPOL=0,CPHA=1

The transfer signal sequence for SPI format with CPOL = 0, CPHA = 1 is shown in Figure 31, which covers both single and continuous transfers.

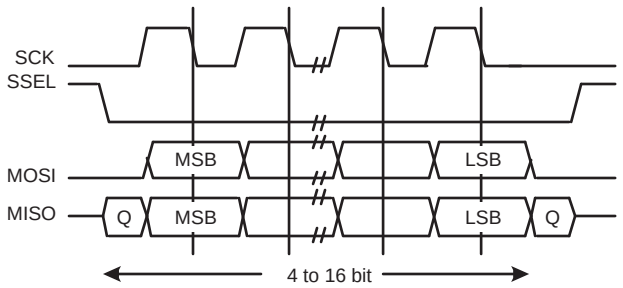


Fig 31. SPI frame format with CPOL=0 and CPHA=1

In this configuration, during idle periods:

- The CLK signal is forced LOW.
- SSEL is forced HIGH.
- The transmit MOSI/MISO pad is in high impedance.

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSEL master signal being driven LOW. Master's MOSI pin is enabled. After a further one half SCK period, both master and slave valid data is enabled onto their respective transmission lines. At the same time, the SCK is enabled with a rising edge transition.

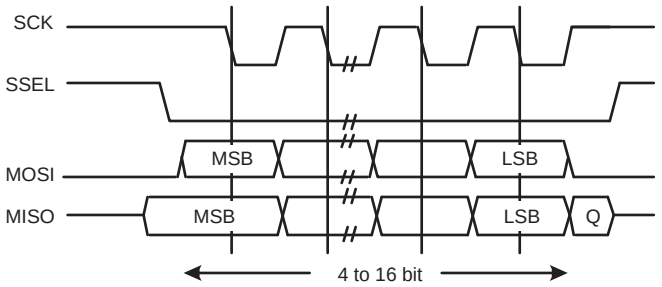
Data is then captured on the falling edges and propagated on the rising edges of the SCK signal.

In the case of a single word transfer, after all bits have been transferred, the SSEL line is returned to its idle HIGH state one SCK period after the last bit has been captured.

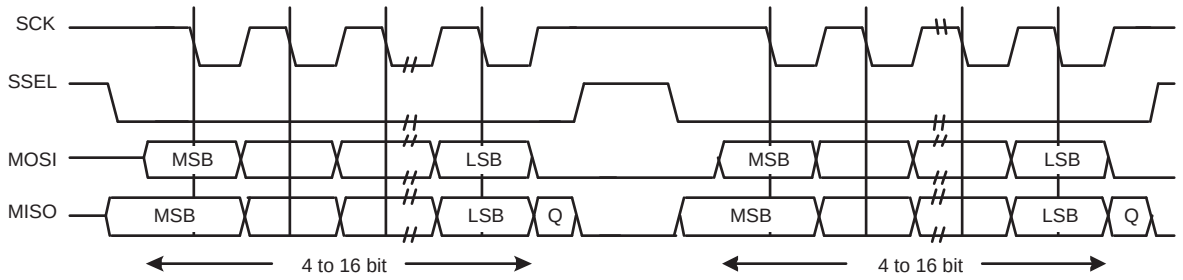
For continuous back-to-back transfers, the SSEL pin is held LOW between successive data words and termination is the same as that of the single word transfer.

12.7.2.4 SPI format with CPOL = 1,CPHA = 0

Single and continuous transmission signal sequences for SPI format with CPOL=1, CPHA=0 are shown in Figure 32.



a. Single transfer with CPOL=1 and CPHA=0



b. Continuous transfer with CPOL=1 and CPHA=0

Fig 32. SPI frame format with CPOL = 1 and CPHA = 0 (a) Single and b) Continuous Transfer)

In this configuration, during idle periods:

- The CLK signal is forced HIGH.
- SSEL is forced HIGH.
- The transmit MOSI/MISO pad is in high impedance.

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSEL master signal being driven LOW, which causes slave data to be immediately transferred onto the MISO line of the master. Master's MOSI pin is enabled.

One half period later, valid master data is transferred to the MOSI line. Now that both the master and slave data have been set, the SCK master clock pin becomes LOW after one further half SCK period. This means that data is captured on the falling edges and be propagated on the rising edges of the SCK signal.

In the case of a single word transmission, after all bits of the data word are transferred, the SSEL line is returned to its idle HIGH state one SCK period after the last bit has been captured.

However, in the case of continuous back-to-back transmissions, the SSEL signal must be pulsed HIGH between each data word transfer. This is because the slave select pin freezes the data in its serial peripheral register and does not allow it to be altered if the CPHA bit is logic zero. Therefore the master device must raise the SSEL pin of the slave device between each data transfer to enable the serial peripheral data write. On completion of the continuous transfer, the SSEL pin is returned to its idle state one SCK period after the last bit has been captured.

12.7.2.5 SPI format with CPOL = 1,CPHA = 1

The transfer signal sequence for SPI format with CPOL = 1, CPHA = 1 is shown in Figure 33, which covers both single and continuous transfers.

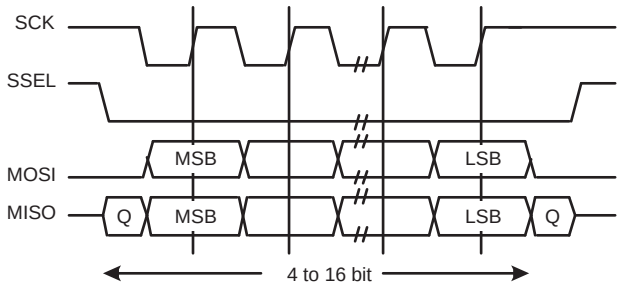


Fig 33. SPI Frame Format with CPOL = 1 and CPHA = 1

In this configuration, during idle periods:

- The CLK signal is forced HIGH.
- SSEL is forced HIGH.
- The transmit MOSI/MISO pad is in high impedance.

If the SSP is enabled and there is valid data within the transmit FIFO, the start of transmission is signified by the SSEL master signal being driven LOW. Master's MOSI is enabled. After a further one half SCK period, both master and slave data are enabled onto their respective transmission lines. At the same time, the SCK is enabled with a falling edge transition. Data is then captured on the rising edges and propagated on the falling edges of the SCK signal.

After all bits have been transferred, in the case of a single word transmission, the SSEL line is returned to its idle HIGH state one SCK period after the last bit has been captured. For continuous back-to-back transmissions, the SSEL pins remains in its active LOW state, until the final bit of the last word has been captured, and then returns to its idle state as described above. In general, for continuous back-to-back transfers the SSEL pin is held LOW between successive data words and termination is the same as that of the single word transfer.

12.7.3 Semiconductor Microwire frame format

Figure 34 shows the Microwire frame format for a single frame. Figure 35 shows the same format when back-to-back frames are transmitted.

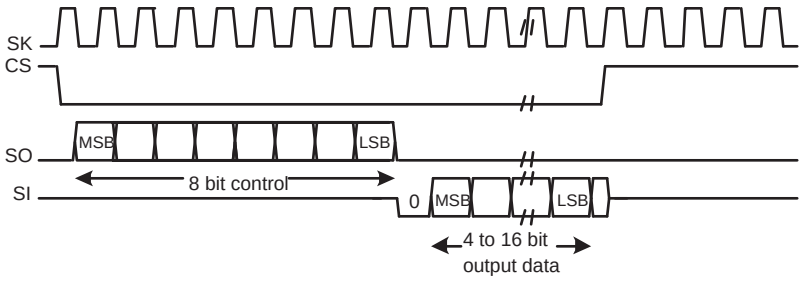


Fig 34. Microwire frame format (single transfer)

Microwire format is very similar to SPI format, except that transmission is half-duplex instead of full-duplex, using a master-slave message passing technique. Each serial transmission begins with an 8-bit control word that is transmitted from the SSP to the off-chip slave device. During this transmission, no incoming data is received by the SSP. After the message has been sent, the off-chip slave decodes it and, after waiting one serial clock after the last bit of the 8-bit control message has been sent, responds with the required data. The returned data is 4 to 16 bit in length, making the total frame length anywhere from 13 to 25 bit.

In this configuration, during idle periods:

- The SK signal is forced LOW.
- CS is forced HIGH.
- The transmit data line SO is arbitrarily forced LOW.

A transmission is triggered by writing a control byte to the transmit FIFO.The falling edge of CS causes the value contained in the bottom entry of the transmit FIFO to be transferred to the serial shift register of the transmit logic, and the MSB of the 8-bit control frame to be shifted out onto the SO pin. CS remains LOW for the duration of the frame transmission. The SI pin remains tristated during this transmission.

The off-chip serial slave device latches each control bit into its serial shifter on the rising edge of each SK. After the last bit is latched by the slave device, the control byte is decoded during a one clock wait-state, and the slave responds by transmitting data back to the SSP. Each bit is driven onto SI line on the falling edge of SK. The SSP in turn latches each bit on the rising edge of SK. At the end of the frame, for single transfers, the CS signal is pulled HIGH one clock period after the last bit has been latched in the receive serial shifter, that causes the data to be transferred to the receive FIFO.

Note: The off-chip slave device can tristate the receive line either on the falling edge of SK after the LSB has been latched by the receive shiftier, or when the CS pin goes HIGH.

For continuous transfers, data transmission begins and ends in the same manner as a single transfer. However, the CS line is continuously asserted (held LOW) and transmission of data occurs back to back. The control byte of the next frame follows directly after the LSB of the received data from the current frame. Each of the received values is transferred from the receive shifter on the falling edge SK, after the LSB of the frame has been latched into the SSP.

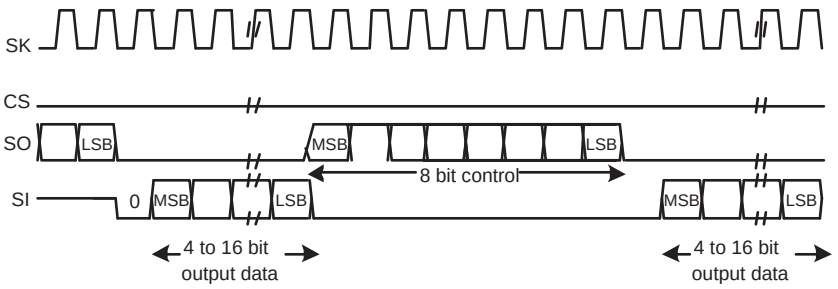


Fig 35. Microwire frame format (continuos transfers)

12.7.3.1 Setup and hold time requirements on CS with respect to SK in Microwire mode

In the Microwire mode, the SSP slave samples the first bit of receive data on the rising edge of SK after CS has gone LOW. Masters that drive a free-running SK must ensure that the CS signal has sufficient setup and hold margins with respect to the rising edge of SK.

Figure 36 illustrates these setup and hold time requirements. With respect to the SK rising edge on which the first bit of receive data is to be sampled by the SSP slave, CS must have a setup of at least two times the period of SK on which the SSP operates. With respect to the SK rising edge previous to this edge, CS must have a hold of at least one SK period.

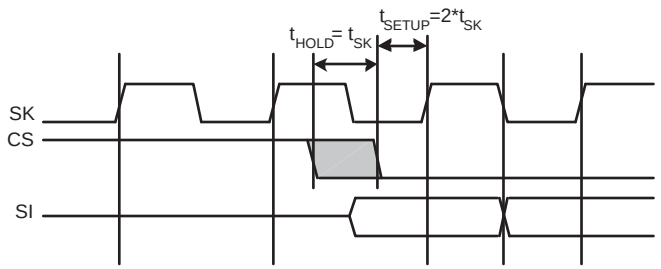


Fig 36. Microwire frame format setup and hold details

13.1 How to read this chapter

CT16B0/1 are available on all LPC122x parts.

13.2 Basic configuration

The peripheral clocks to the I6-bit timer counter blocks are provided by the system clock, which is controlled by the SYSAHBCLKDIV register (Table 20). The CT16B0/1 blocks can be disabled through the System AHB clock control register bits 7 and 8 (Table 21) for power savings.

13.3 Features

- Two 16-bit counter/timers with a programmable 16-bit prescaler.
- Counter or timer operation
- Two 16-bit capture channels that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.
- The timer and prescaler may be configured to be cleared on a designated capture event. This feature permits easy pulse-width measurement by clearing the timer on the leading edge of an input pulse and capturing the timer value on the trailing edge.
- Four 16-bit match registers that allow:
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- Two external outputs corresponding to match registers with the following capabilities:
  - Set LOW on match.
  - Set HIGH on match.
  - Toggle on match.
  - Do nothing on match.
- For each timer, up to four match registers can be configured as PWM allowing to use up to two match outputs as single edge controlled PWM outputs.
- Up to two match registers can be used to generate timed DMA requests.

13.4 Applications

- Interval Timer for counting internal events
- Pulse Width Demodulator via capture input
- Free running timer
- Pulse Width Modulator via match outputs

13.5 Description

Each Counter/timer is designed to count cycles of the peripheral clock (PCLK) or an externally supplied clock and can optionally generate interrupts or perform other actions at specified timer values based on four match registers. Each counter/timer also includes one capture input to trap the timer value when an input signal transitions, optionally generating an interrupt.

In PWM mode, two match registers can be used to provide a single-edge controlled PWM output on the match output pins. It is recommended to use the match registers that are not pinned out to control the PWM cycle length.

Remark: The 16-bit counter/timer0 (CT16B0) and the 16-bit counter/timer1 (CT16B1) are functionally identical except for the peripheral base address.

13.6 Pin description

Table 217 gives a brief summary of each of the counter/timer related pins.

Table 217. Counter/timer pin description

Pin	Type	Description
CT16B0_CAP[1:0] CT16B1_CAP[1:0]	Input	Capture Signal: A transition on a capture pin can be configured to load the Capture Register with the value in the counter/timer and optionally generate an interrupt. Counter/Timer block can select a capture signal as a clock source instead of the PCLK derived clock. For more details see Section 13.7.11.
CT16B0_MAT[1:0] CT16B1_MAT[1:0]	Output	External Match Outputs of CT16B0/1: When a match register of CT16B0/1 (MR1:0) equals the timer counter (TC), this output can either toggle, go LOW, go HIGH, or do nothing. The External Match Register (EMR) and the PWM Control Register (PWMCON) control the functionality of this output.

In addition, the level and edge outputs of the two comparators are internally connected to the remaining capture channels 2 and 3 of each of the 16-bit counter/timers (see Table 218). For details on the comparator outputs, see Section 18.7.5.

Table 218. Comparator connections to the 16-bit counter/timer

Comparator output	Timer capture input
Comparator 0, level output	CT16B0_CAP2
Comparator 0, edge output	CT16B0_CAP3
Comparator 1, level output	CT16B1_CAP2
Comparator 1, edge output	CT16B1_CAP3

13.7 Register description

The 16-bit counter/timer0 contains the registers shown in Table 219 and the 16-bit counter/timer1 contains the registers shown in Table 220. More detailed descriptions follow.

Table 219. Register overview: 16-bit counter/timer 0 CT16B0 (base address 0x4001 0000)

Name	Access	Address offset	Description	Reset value[1]
IR	R/W	0x000	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	0
TCR	R/W	0x004	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	0
TC	R/W	0x008	Timer Counter. The 16-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	0
PR	R/W	0x00C	Prescale Register. When the Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	0
PC	R/W	0x010	Prescale Counter. The 16-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	0
MCR	R/W	0x014	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	0
MR0	R/W	0x018	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	0
MR1	R/W	0x01C	Match Register 1. See MR0 description.	0
MR2	R/W	0x020	Match Register 2. See MR0 description.	0
MR3	R/W	0x024	Match Register 3. See MR0 description.	0
CCR	R/W	0x028	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	0
CR0	RO	0x02C	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CT16B0_CAP0 input.	0
CR1	RO	0x030	Capture Register 1. CR1 is loaded with the value of TC when there is an event on the CT16B0_CAP1 input.	0
CR2	RO	0x034	Capture Register 3. CR2 is loaded with the value of TC when there is an event on the input from the comparator.	0
CR3	RO	0x038	Capture Register 3. CR3 is loaded with the value of TC when there is an event on the input from the comparator.	0
EMR	R/W	0x03C	External Match Register. The EMR controls the match function and the external match pins CT16B0_MAT[1:0] and CT16B1_MAT[1:0].	0
-	-	0x040 - 0x06C	reserved	-
CTCR	R/W	0x070	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0
PWMC	R/W	0x074	PWM Control Register. The PWMCON enables PWM mode for the external match pins CT16B0_MAT[1:0] and CT16B1_MAT[1:0].	0

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

Table 220. Register overview: 16-bit counter/timer 1 CT16B1 (base address 0x4001 4000)

Name	Access	Address	Description	Reset value[1]
IR	R/W	0x000	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	0
TCR	R/W	0x004	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	0
TC	R/W	0x008	Timer Counter. The 16-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	0
PR	R/W	0x00C	Prescale Register. When the Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	0
PC	R/W	0x010	Prescale Counter. The 16-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	0
MCR	R/W	0x014	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	0
MR0	R/W	0x018	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	0
MR1	R/W	0x01C	Match Register 1. See MR0 description.	0
MR2	R/W	0x020	Match Register 2. See MR0 description.	0
MR3	R/W	0x024	Match Register 3. See MR0 description.	0
CCR	R/W	0x028	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	0
CR0	RO	0x02C	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CT16B1_CAP0 input.	0
CR1	RO	0x030	Capture Register 1. CR1 is loaded with the value of TC when there is an event on the CT16B1_CAP1 input.	0
CR2	RO	0x034	Capture Register 3. CR2 is loaded with the value of TC when there is an event on the input from the comparator.	0
CR3	RO	0x038	Capture Register 3. CR3 is loaded with the value of TC when there is an event on the input from the comparator.	0
EMR	R/W	0x03C	External Match Register. The EMR controls the match function and the external match pins CT16B0_MAT[1:0] and CT16B1_MAT[1:0].	0
-	-	0x040 - 0x06C	reserved	-
CTCR	R/W	0x070	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0
PWMC	R/W	0x074	PWM Control Register. The PWMCON enables PWM mode for the external match pins CT16B0_MAT[1:0] and CT16B1_MAT[1:0].	0

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

13.7.1 Interrupt Register

The Interrupt Register consists of four bits for the match interrupts and two bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be HIGH. Otherwise, the bit will be LOW. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect. Clearing an interrupt for timer match also clears any corresponding DMA request.

Table 221. Interrupt Register (IR, address 0x4001 0000 (CT16B0) and 0x4001 4000 (CT16B1)) bit description

Bit	Symbol	Description	Reset value
0	MR0INT	Interrupt flag for match channel 0.	0
1	MR1INT	Interrupt flag for match channel 1.	0
2	MR2INT	Interrupt flag for match channel 2.	0
3	MR3INT	Interrupt flag for match channel 3.	0
4	CR0INT	Interrupt flag for capture channel 0 event.	0
5	CR1INT	Interrupt flag for capture channel 1 event.	0
6	CR2INT	Interrupt flag for capture channel 2 event.	0
7	CR3INT	Interrupt flag for capture channel 3 event.	0
31:8	-	Reserved	-

13.7.2 Timer Control Register

The Timer Control Register (TCR) is used to control the operation of the counter/timer.

Table 222. Timer Control Register (TCR, address 0x4001 0004 (CT16B0) and 0x4001 4004 (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	CEN		Counter enable.	0
		0	The counters are disabled.	
		1	The Timer Counter and Prescale Counter are enabled for counting.	
1	CRST		Counter reset.	0
		0	Do nothing.	
		1	The Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until TCR[1] is returned to zero.	
31:2	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

13.7.3 Timer Counter

The 16-bit Timer Counter is incremented when the Prescale Counter reaches its terminal count. Unless it is reset before reaching its upper limit, the TC will count up to the value 0x0000 FFFF and then wrap back to the value 0x0000 0000. This event does not cause an interrupt, but a Match register can be used to detect an overflow if needed.

Table 223: Timer counter registers (TC, address 0x4001 0008 (CT16B0) and 0x4001 4008 (CT16B1)) bit description

Bit	Symbol	Description	Reset value
15:0	TC	Timer counter value.	0
31:16	-	Reserved.	-

13.7.4 Prescale Register

The 16-bit Prescale Register specifies the maximum value for the Prescale Counter.

Table 224: Prescale registers (PR, address 0x4001 000C (CT16B0) and 0x4001 400C (CT16B1)) bit description

Bit	Symbol	Description	Reset value
15:0	PCVAL	Prescale value.	0
31:16	-	Reserved.	-

13.7.5 Prescale Counter register

The 16-bit Prescale Counter controls division of PCLK by some constant value before it is applied to the Timer Counter. This allows control of the relationship between the resolution of the timer and the maximum time before the timer overflows. The Prescale Counter is incremented on every PCLK. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented, and the Prescale Counter is reset on the next PCLK. This causes the TC to increment on every PCLK when PR = 0, every 2 PCLKs when PR = 1, etc..

Table 225: Prescale counter registers (PC, address 0x4001 0010 (CT16B0) and 0x4001 4010 (CT16B1)) bit description

Bit	Symbol	Description	Reset value
15:0	PC	Prescale counter value.	0
31:16	-	Reserved.	-

13.7.6 Match Control Register

The Match Control Register is used to control what operations are performed when one of the Match Registers matches the Timer Counter. The function of each of the bits is shown in Table 226.

Table 226. Match Control Register (MCR, address 0x4001 0014 (CT16B0) and 0x4001 4014 (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	MR0I		Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC.	0
		1	Enabled	
		0	Disabled	

Table 226. Match Control Register (MCR, address 0x4001 0014 (CT16B0) and 0x4001 4014 (CT16B1)) bit description
…continued

Bit	Symbol	Value	Description	Reset value
1	MR0R		Reset on MR0: the TC will be reset if MR0 matches it.	0
		1	Enabled	
		0	Disabled	
2	MR0S		Stop on MR0: the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC.	0
		1	Enabled	
		0	Disabled	
3	MR1I		Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC.	0
		1	Enabled	
		0	Disabled	
4	MR1R		Reset on MR1: the TC will be reset if MR1 matches it.	0
		1	Enabled	
		0	Disabled	
5	MR1S		Stop on MR1: the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC.	0
		1	Enabled	
		0	Disabled	
6	MR2I		Interrupt on MR2: an interrupt is generated when MR2 matches the value in the TC.	0
		1	Enabled	
		0	Disabled	
7	MR2R		Reset on MR2: the TC will be reset if MR2 matches it.	0
		1	Enabled	
		0	Disabled	
8	MR2S		Stop on MR2: the TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC.	0
		1	Enabled	
		0	Disabled	
9	MR3I		Interrupt on MR3: an interrupt is generated when MR3 matches the value in the TC.	0
		1	Enabled	
		0	Disabled	
10	MR3R		Reset on MR3: the TC will be reset if MR3 matches it.	0
		1	Enabled	
		0	Disabled	
11	MR3S		Stop on MR3: the TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC.	0
		1	Enabled	
		0	Disabled	
31:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

13.7.7 Match Registers

The Match register values are continuously compared to the Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

Table 227: Match registers (MR0 to 3, addresses 0x4001 0018 to 24 (CT16B0) and 0x4001 4018 to 24 (CT16B1)) bit description

Bit	Symbol	Description	Reset value
15:0	MATCH	Timer counter match value.	0
31:16	-	Reserved.	-

13.7.8 Capture Control Register

The Capture Control Register is used to control whether the Capture Register is loaded with the value in the Counter/timer when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges. In the description below, "n" represents the Timer number, 0 or 1.

On the LPC122x, the capture channels 2 and 3 are connected to the edge and level outputs of the comparators (see Table 218). In the description below, “n” also represents the comparator number, 0 or 1. Comparator 0 is connected to CT16B0, and comparator 1 is connected to CT16B1.

Table 228. Capture Control Register (CCR, address 0x4001 0028 (CT16B0) and 0x4001 4028 (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	CAP0RE		Capture on CT16Bn_CAP0 rising edge: a sequence of 0 then 1 on CT16Bn_CAP0 will cause CR0 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
1	CAP0FE		Capture on CT16Bn_CAP0 falling edge: a sequence of 1 then 0 on CT16Bn_CAP0 will cause CR0 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
2	CAP0I		Interrupt on CT16Bn_CAP0 event: a CR0 load due to a CT16Bn_CAP0 event will generate an interrupt.	0
		1	Enabled.	
		0	Disabled.	
3	CAP1RE		Capture on CT16Bn_CAP1 rising edge: a sequence of 0 then 1 on CT16Bn_CAP1 will cause CR1 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	

Table 228. Capture Control Register (CCR, address 0x4001 0028 (CT16B0) and 0x4001 4028 (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
4	CAP1FE		Capture on CT16Bn_CAP1 falling edge: a sequence of 1 then 0 on CT16Bn_CAP1 will cause CR1 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
5	CAP1I		Interrupt on CT16Bn_CAP1 event: a CR1 load due to a CT16Bn_CAP1 event will generate an interrupt.	0
		1	Enabled.	
		0	Disabled.	
6	CAP2RE		Capture on comparator n level output - rising edge: a sequence of 0 then 1 on the comparator n output will cause CR2 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
7	CAP2FE		Capture on comparator n level output - falling edge: a sequence of 1 then 0 on comparator n output will cause CR2 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
8	CAP2I		Interrupt on comparator n level output event: a CR2 load due to a comparator 0 event will generate an interrupt.	0
		1	Enabled.	
		0	Disabled.	
9	CAP3RE		Capture on comparator n edge output - rising edge: a sequence of 0 then 1 on the comparator n output will cause CR3 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
10	CAP3FE		Capture on comparator n edge output - falling edge: a sequence of 1 then 0 on comparator n output will cause CR3 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
11	CAP3I		Interrupt on comparator n edge output event: a CR3 load due to a comparator n event will generate an interrupt.	0
		1	Enabled.	
		0	Disabled.	
31:12	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

13.7.9 Capture Registers

Each Capture register is associated with a device pin and may be loaded with the counter/timer value when a specified event occurs on that pin. The settings in the Capture Control Register register determine whether the capture function is enabled, and whether a capture event happens on the rising edge of the associated pin, the falling edge, or on both edges.

Table 229: Capture registers (CR0 to 3, addresses 0x4001 002C to 38 (CT16B0) and 0x4001 402C to 38 (CT16B1)) bit description

Bit	Symbol	Description	Reset value
15:0	CAP	Timer counter capture value.	0
31:16	-	Reserved.	-

13.7.10 External Match Register

The External Match Register provides both control and status of the external match pins CT16Bn_MAT[1:0].

Match events for Match 0 and Match 1 in each timer can cause a DMA request.

If the match outputs are configured as PWM output, the function of the external match registers is determined by the PWM rules (Section 13.7.13 “Rules for single edge controlled PWM outputs” on page 254).

Table 230. External Match Register (EMR, address 0x4001 003C (CT16B0) and 0x4001 403C (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	EM0		External Match 0. This bit reflects the state of output CT16B0_MAT0/CT16B1_MAT0, whether or not this output is connected to its pin. When a match occurs between the TC and MR0, this bit can either toggle, go LOW, go HIGH, or do nothing. Bits EMR[5:4] control the functionality of this output. This bit is driven to the CT16B0_MAT0/CT16B1_MAT0 pins if the match function is selected in the IOCON registers (0 = LOW, 1 = HIGH).	0
1	EM1		External Match 1. This bit reflects the state of output CT16B0_MAT1/CT16B1_MAT1, whether or not this output is connected to its pin. When a match occurs between the TC and MR1, this bit can either toggle, go LOW, go HIGH, or do nothing. Bits EMR[7:6] control the functionality of this output. This bit is driven to the CT16B0_MAT0/CT16B1_MAT0 pins if the match function is selected in the IOCON registers (0 = LOW, 1 = HIGH).	0
2	EM2		External Match 2. This bit reflects the state of match channel 2. When a match occurs between the TC and MR2, this bit can either toggle, go LOW, go HIGH, or do nothing. Bits EMR[9:8] control the functionality of this output.	0
3	EM3		External Match 3. This bit reflects the state of output of match channel 3. When a match occurs between the TC and MR3, this bit can either toggle, go LOW, go HIGH, or do nothing. Bits EMR[11:10] control the functionality of this output.	0
5:4	EMC0		External Match Control 0. Determines the functionality of External Match 0. Table 231 shows the encoding of these bits.	00
		0x0	Do Nothing.	
		0x1	Clear the corresponding External Match bit/output to 0 (CT16Bi_MAT0 pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT16Bi_MAT0 pin is HIGH if pinned out).	
		0x3	Toggle the corresponding External Match bit/output.	

Table 230. External Match Register (EMR, address 0x4001 003C (CT16B0) and 0x4001 403C (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
7:6	EMC1		External Match Control 1. Determines the functionality of External Match 1.	00
		0x0	Do Nothing.	
		0x1	Clear the corresponding External Match bit/output to 0 (CT16Bi_MAT1 pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT16Bi_MAT1 pin is HIGH if pinned out).	
		0x3	Toggle the corresponding External Match bit/output.	
9:8	EMC2		External Match Control 2. Determines the functionality of External Match 2.	00
		0x0	Do Nothing.	
		0x1	Clear the corresponding External Match bit/output to 0 (CT16Bi_MAT2 pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT16Bi_MAT2 pin is HIGH if pinned out).	
		0x3	Toggle the corresponding External Match bit/output.	
11:10	EMC3		External Match Control 3. Determines the functionality of External Match 3. Table 231 shows the encoding of these bits.	00
		0x0	Do Nothing.	
		0x1	Clear the corresponding External Match bit/output to 0 (CT16Bi_MAT3 pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT16Bi_MAT3 pin is HIGH if pinned out).	
		0x3	Toggle the corresponding External Match bit/output.	
31:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 231. External match control

EMR[11:10], EMR[9:8], EMR[7:6], or EMR[5:4]	Function
00	Do Nothing.
01	Clear the corresponding External Match bit/output to 0 (CT16Bn_MATm pin is LOW if pinned out).
10	Set the corresponding External Match bit/output to 1 (CT16Bn_MATm pin is HIGH if pinned out).
11	Toggle the corresponding External Match bit/output.

13.7.10.1 DMA operation

DMA requests are generated by 0 to 1 transitions of the External Match 0 bit of each timer. In order to have an effect, the GPDMA must be configured and the relevant timer DMA request selected as a DMA source. When a timer is initially set up to generate a DMA request, the request may already be asserted before a match condition occurs. An initial DMA request may be avoided by having software write a one to clear the timer interrupt flag. A DMA request will be automatically cleared via hardware by the DMA controller after servicing.

If the EMR bits are set to 10 or 11 for channel 0 (rising edge or toggle), a DMA request is generated even if the corresponding MR register is set to 0 because a match-on-zero condition exists. To disable any DMA requests, set the EMR bits for channel 0 to 00.

13.7.11 Count Control Register

The Count Control Register (CTCR) is used to select between Timer and Counter mode, and in Counter mode to select the pin and edge(s) for counting.

When Counter Mode is chosen as a mode of operation, the CAP input (selected by the CTCR bits 3:2) is sampled on every rising edge of the PCLK clock. After comparing two consecutive samples of this CAP input, one of the following four events is recognized: rising edge, falling edge, either of edges or no changes in the level of the selected CAP input. Only if the identified event occurs, and the event corresponds to the one selected by bits 1:0 in the CTCR register, will the Timer Counter register be incremented.

Effective processing of the externally supplied clock to the counter has some limitations. Since two successive rising edges of the PCLK clock are used to identify only one edge on the CAP selected input, the frequency of the CAP input can not exceed one half of the PCLK clock. Consequently, the duration of the HIGH/LOW levels on the same CAP input in this case can not be shorter than 1/PCLK.

Bits 7:4 of this register are also used to enable and configure the capture-clears-timer feature. This feature allows for a designated edge on a particular CAP input to reset the timer to all zeros. Using this mechanism to clear the timer on the leading edge of an input pulse and performing a capture on the trailing edge, permits direct pulse-width measurement using a single capture input without the need to perform a subtraction operation in software.

Table 232. Count Control Register (CTCR, address 0x4001 0070 (CT16B0) and 0x4001 4070 (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
1:0	CTM		Counter/Timer Mode. This field selects which rising PCLK edges can increment Timer's Prescale Counter (PC), or clear PC and increment Timer Counter (TC). **Remark:** If Counter mode is selected in the CTCR, bits 2:0 in the Capture Control Register (CCR) must be programmed as 000.	00
		0x0	Timer Mode: every rising PCLK edge	
		0x1	Counter Mode: TC is incremented on rising edges on the CAP input selected by bits 3:2.	
		0x2	Counter Mode: TC is incremented on falling edges on the CAP input selected by bits 3:2.	
		0x3	Counter Mode: TC is incremented on both edges on the CAP input selected by bits 3:2.	

Table 232. Count Control Register (CTCR, address 0x4001 0070 (CT16B0) and 0x4001 4070 (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
3:2	CIS		Count Input Select. In counter mode (when bits 1:0 in this register are not 00), these bits select which CAP pin or comparator output is sampled for clocking:	00
		0x0	CT16Bn_CAP0	
		0x1	CT16Bn_CAP1	
		0x2	Comparator n, level output	
		0x3	Comparator n, edge output	
4	ENCC		Setting this bit to 1 enables clearing of the timer and the prescaler when the capture-edge event specified in bits 7:5 occurs.	0
7:5	SELCC		When bit 4 is a 1, these bits select which capture input edge will cause the timer and prescaler to be cleared. These bits have no effect when bit 4 is low.	
		0x0	Rising Edge of CAP0 clears the timer (if bit 4 is set)	
		0x1	Falling Edge of CAP0 clears the timer (if bit 4 is set)	
		0x2	Rising Edge of CAP1 clears the timer (if bit 4 is set)	
		0x3	Falling Edge of CAP1 clears the timer (if bit 4 is set)	
		0x4	Rising Edge of CAP2 clears the timer (if bit 4 is set)	
		0x5	Falling Edge of CAP2 clears the timer (if bit 4 is set)	
		0x6	Rising Edge of CAP3 clears the timer (if bit 4 is set)	
		0x7	Falling Edge of CAP3 clears the timer (if bit 4 is set)	
31:8	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

13.7.12 PWM Control register

The PWM Control Register is used to configure the match outputs as PWM outputs. Each match output can be independently set to perform either as PWM output or as match output whose function is controlled by the External Match Register (EMR).

For each timer, a maximum of three single edge controlled PWM outputs can be selected on the CT16Bn_MAT[1:0] outputs. One additional match register determines the PWM cycle length. When a match occurs in any of the other match registers, the PWM output is set to HIGH. The timer is reset by the match register that is configured to set the PWM cycle length. When the timer is reset to zero, all currently HIGH match outputs configured as PWM outputs are cleared.

Table 233. PWM Control Register (PWMC, address 0x4001 0074 and 0x4001 4074 (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	PWMEN0		PWM mode enable for channel0.	0
		0	CT16Bi_MAT0 is controlled by EM0.	
		1	PWM mode is enabled for CT16Bi_MAT0.	

Table 233. PWM Control Register (PWMC, address 0x4001 0074 and 0x4001 4074 (CT16B1)) bit description

Bit	Symbol	Value	Description	Reset value
1	PWMEN1		PWM mode enable for channel1.	0
		0	CT16Bi_MAT01 is controlled by EM1.	
		1	PWM mode is enabled for CT16Bi_MAT1.	
2	PWMEN2		PWM mode enable for channel2.	0
		0	CT16Bi_MAT2 is controlled by EM2.	
		1	PWM mode is enabled for CT16Bi_MAT2.	
3	PWMEN3		PWM mode enable for channel3.	0
		0	CT16Bi_MAT3 is controlled by EM3.	
		1	PWM mode is enabled for CT16Bi_MAT3.	
31:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

13.7.13 Rules for single edge controlled PWM outputs

1. All single edge controlled PWM outputs go LOW at the beginning of each PWM cycle (timer is set to zero) unless their match value is equal to zero.
2. Each PWM output will go HIGH when its match value is reached. If no match occurs (i.e. the match value is greater than the PWM cycle length), the PWM output remains continuously LOW.
3. If a match value larger than the PWM cycle length is written to the match register, and the PWM signal is HIGH already, then the PWM signal will be cleared on the next start of the next PWM cycle.
4. If a match register contains the same value as the timer reset value (the PWM cycle length), then the PWM output will be reset to LOW on the next clock tick. Therefore, the PWM output will always consist of a one clock tick wide positive pulse with a period determined by the PWM cycle length (i.e. the timer reload value).
5. If a match register is set to zero, then the PWM output will go to HIGH the first time the timer goes back to zero and will stay HIGH continuously.

Note: When the match outputs are selected to perform as PWM outputs, the timer reset (MRnR) and timer stop (MRnS) bits in the Match Control Register MCR must be set to zero except for the match register setting the PWM cycle length. For this register, set the MRnR bit to one to enable the timer reset when the timer value matches the value of the corresponding match register.

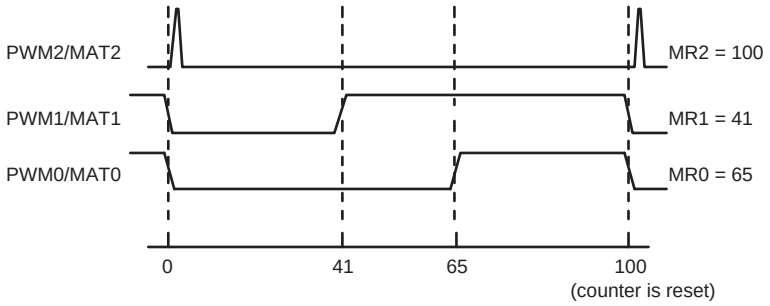


Fig 37. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.

13.8 Example timer operation

Figure 38 shows a timer configured to reset the count and generate an interrupt on match. The prescaler is set to 2 and the match register set to 6. At the end of the timer cycle where the match occurs, the timer count is reset. This gives a full length cycle to the match value. The interrupt indicating that a match occurred is generated in the next clock after the timer reached the match value.

Figure 39 shows a timer configured to stop and generate an interrupt on match. The prescaler is again set to 2 and the match register set to 6. In the next clock after the timer reaches the match value, the timer enable bit in TCR is cleared, and the interrupt indicating that a match occurred is generated.

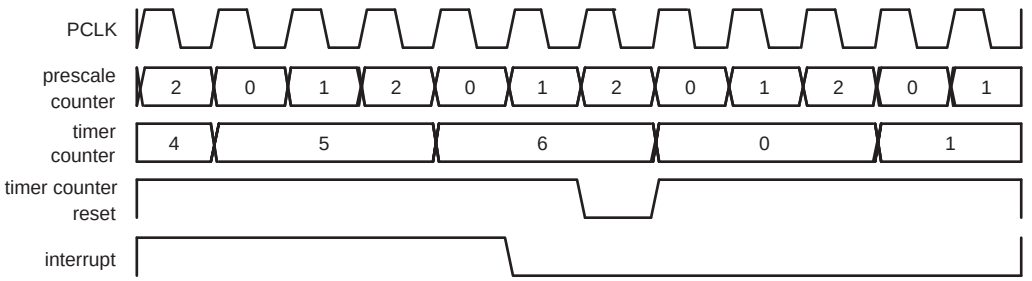


Fig 38. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

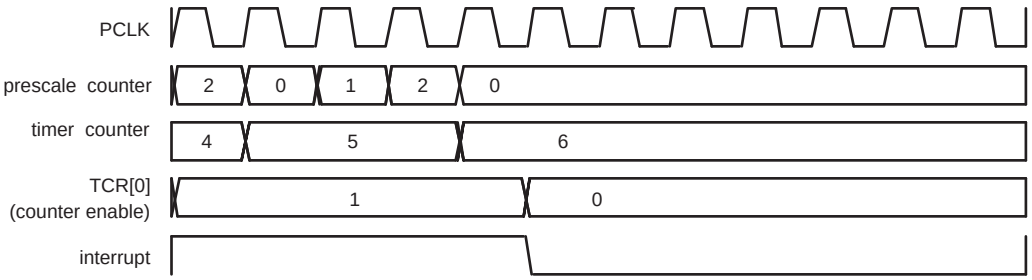


Fig 39. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

13.9 Architecture

The block diagram for counter/timer0 and counter/timer1 is shown in Figure 40.

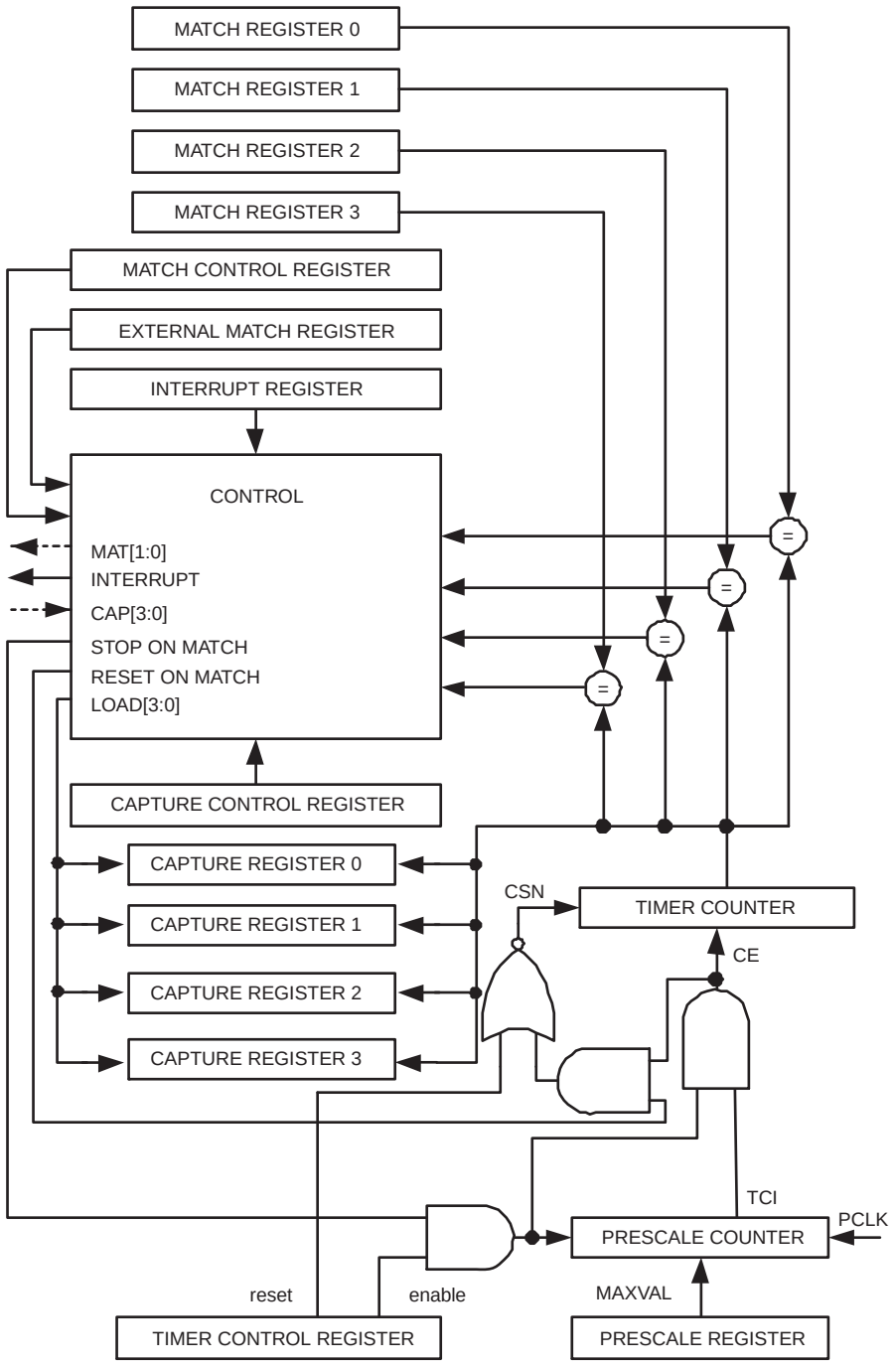


Fig 40. 16-bit counter/timer block diagram

14.1 How to read this chapter

CT32B0/1 are available on all LPC122x parts.

14.2 Basic configuration

The peripheral clocks to the 32-bit timer counter blocks are provided by the system clock, which is controlled by the SYSAHBCLKDIV register (Table 20). The CT32B0/1 blocks can be disabled through the System AHB clock control register bits 9 and 10 (Table 21) for power savings.

14.3 Features

- Two 32-bit counter/timers with a programmable 32-bit prescaler.
- Counter or timer operation.
- Four 32-bit capture channels that can take a snapshot of the timer value when an input signal transitions. A capture event may also optionally generate an interrupt.
- The timer and prescaler may be configured to be cleared on a designated capture event. This feature permits easy pulse-width measurement by clearing the timer on the leading edge of an input pulse and capturing the timer value on the trailing edge.
- Four 32-bit match registers that allow:
  - Continuous operation with optional interrupt generation on match.
  - Stop timer on match with optional interrupt generation.
  - Reset timer on match with optional interrupt generation.
- Four external outputs corresponding to match registers with the following capabilities:
  - Set LOW on match.
  - Set HIGH on match.
  - Toggle on match.
  - Do nothing on match.
- For each timer, up to four match registers can be configured as PWM allowing to use up to three match outputs as single edge controlled PWM outputs.
- Up to two match registers can be used to generate timed DMA requests.

14.4 Applications

- Interval timer for counting internal events
- Pulse Width Demodulator via capture input
- Free running timer
- Pulse Width Modulator via match outputs

14.5 General description

Each Counter/timer is designed to count cycles of the peripheral clock (PCLK) or an externally supplied clock and can optionally generate interrupts or perform other actions at specified timer values based on four match registers. Each counter/timer also includes one capture input to trap the timer value when an input signal transitions, optionally generating an interrupt.

In PWM mode, three match registers can be used to provide a single-edge controlled PWM output on the match output pins. One match register is used to control the PWM cycle length.

Remark: 32-bit counter/timer0 (CT32B0) and 32-bit counter/timer1 (CT32B1) are functionally identical except for their peripheral base addresses.

14.6 Pin description

Table 234 gives a brief summary of each of the counter/timer related pins.

Table 234. Counter/timer pin description

Pin	Type	Description
CT32B0_CAP[3:0] CT32B1_CAP[3:0]	Input	Capture Signals: A transition on a capture pin can be configured to load one of the Capture Registers with the value in the Timer Counter and optionally generate an interrupt. The counter/timer block can select a capture signal as a clock source instead of the PCLK derived clock. For more details see Section 14.7.11 “Count Control Register” on page 267.
CT32B0_MAT[3:0] CT32B1_MAT[3:0]	Output	External Match Output of CT32B0/1: When a match register MR3:0 equals the timer counter (TC), this output can either toggle, go LOW, go HIGH, or do nothing. The External Match Register (EMR) and the PWM Control register (PWMCON) control the functionality of this output.

14.7 Register description

32-bit counter/timer0 contains the registers shown in Table 235 and 32-bit counter/timer1 contains the registers shown in Table 236. More detailed descriptions follow.

Table 235. Register overview: 32-bit counter/timer 0 CT32B0 (base address 0x4001 8000)

Name	Access	Address offset	Description	Reset value[1]
IR	R/W	0x000	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	0
TCR	R/W	0x004	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	0
TC	R/W	0x008	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	0
PR	R/W	0x00C	Prescale Register. When the Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	0

Table 235. Register overview: 32-bit counter/timer 0 CT32B0 (base address 0x4001 8000) *…continued*

Name	Access	Address offset	Description	Reset value[1]
PC	R/W	0x010	Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	0
MCR	R/W	0x014	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	0
MR0	R/W	0x018	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	0
MR1	R/W	0x01C	Match Register 1. See MR0 description.	0
MR2	R/W	0x020	Match Register 2. See MR0 description.	0
MR3	R/W	0x024	Match Register 3. See MR0 description.	0
CCR	R/W	0x028	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	0
CR0	RO	0x02C	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CT32B0_CAP0 input.	0
CR1	RO	0x030	Capture Register 1. CR1 is loaded with the value of TC when there is an event on the CT32B0_CAP1 input.	0
CR2	RO	0x034	Capture Register 2. CR2 is loaded with the value of TC when there is an event on the CT32B0_CAP2 input.	0
CR3	RO	0x038	Capture Register 3. CR3 is loaded with the value of TC when there is an event on the CT32B3_CAP3 input.	0
EMR	R/W	0x03C	External Match Register. The EMR controls the match function and the external match pins CT32Bn_MAT[3:0].	0
-	-	0x040 - 0x06C	reserved	-
CTCR	R/W	0x070	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0
PWMC	R/W	0x074	PWM Control Register. The PWMCON enables PWM mode for the external match pins CT32Bn_MAT[3:0].	0

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

Table 236. Register overview: 32-bit counter/timer 1 CT32B1 (base address 0x4001 C000)

Name	Access	Address offset	Description	Reset value[1]
IR	R/W	0x000	Interrupt Register. The IR can be written to clear interrupts. The IR can be read to identify which of eight possible interrupt sources are pending.	0
TCR	R/W	0x004	Timer Control Register. The TCR is used to control the Timer Counter functions. The Timer Counter can be disabled or reset through the TCR.	0
TC	R/W	0x008	Timer Counter. The 32-bit TC is incremented every PR+1 cycles of PCLK. The TC is controlled through the TCR.	0

Table 236. Register overview: 32-bit counter/timer 1 CT32B1 (base address 0x4001 C000) *…continued*

Name	Access	Address offset	Description	Reset value[1]
PR	R/W	0x00C	Prescale Register. When the Prescale Counter (below) is equal to this value, the next clock increments the TC and clears the PC.	0
PC	R/W	0x010	Prescale Counter. The 32-bit PC is a counter which is incremented to the value stored in PR. When the value in PR is reached, the TC is incremented and the PC is cleared. The PC is observable and controllable through the bus interface.	0
MCR	R/W	0x014	Match Control Register. The MCR is used to control if an interrupt is generated and if the TC is reset when a Match occurs.	0
MR0	R/W	0x018	Match Register 0. MR0 can be enabled through the MCR to reset the TC, stop both the TC and PC, and/or generate an interrupt every time MR0 matches the TC.	0
MR1	R/W	0x01C	Match Register 1. See MR0 description.	0
MR2	R/W	0x020	Match Register 2. See MR0 description.	0
MR3	R/W	0x024	Match Register 3. See MR0 description.	0
CCR	R/W	0x028	Capture Control Register. The CCR controls which edges of the capture inputs are used to load the Capture Registers and whether or not an interrupt is generated when a capture takes place.	0
CR0	RO	0x02C	Capture Register 0. CR0 is loaded with the value of TC when there is an event on the CT32B1_CAP0 input.	0
CR1	RO	0x030	Capture Register 1. CR1 is loaded with the value of TC when there is an event on the CT32B1_CAP1 input.	0
CR2	RO	0x034	Capture Register 2. CR2 is loaded with the value of TC when there is an event on the CT32B1_CAP2 input.	0
CR3	RO	0x038	Capture Register 3. CR3 is loaded with the value of TC when there is an event on the CT32B1_CAP3 input.	0
EMR	R/W	0x03C	External Match Register. The EMR controls the match function and the external match pins CT32Bn_MAT[3:0].	0
-	-	0x040 - 0x06C	reserved	-
CTCR	R/W	0x070	Count Control Register. The CTCR selects between Timer and Counter mode, and in Counter mode selects the signal and edge(s) for counting.	0
PWMC	R/W	0x074	PWM Control Register. The PWMCON enables PWM mode for the external match pins CT32Bn_MAT[3:0].	0

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

14.7.1 Interrupt Register

The Interrupt Register consists of four bits for the match interrupts and four bits for the capture interrupts. If an interrupt is generated then the corresponding bit in the IR will be HIGH. Otherwise, the bit will be LOW. Writing a logic one to the corresponding IR bit will reset the interrupt. Writing a zero has no effect. Clearing an interrupt for timer match also clears any corresponding DMA request.

Table 237: Interrupt Register (IR, address 0x4001 8000 (CT32B0) and IR, address 0x4001 C000) bit description

Bit	Symbol	Description	Reset value
0	MR0INT	Interrupt flag for match channel 0.	0
1	MR1INT	Interrupt flag for match channel 1.	0
2	MR2INT	Interrupt flag for match channel 2.	0
3	MR3INT	Interrupt flag for match channel 3.	0
4	CR0INT	Interrupt flag for capture channel 0 event.	0
5	CR1INT	Interrupt flag for capture channel 1 event.	0
6	CR2INT	Interrupt flag for capture channel 2 event.	0
7	CR3INT	Interrupt flag for capture channel 3 event.	0
31:8	-	Reserved	-

14.7.2 Timer Control Register

The Timer Control Register (TCR) is used to control the operation of the counter/timer.

Table 238: Timer Control Register (TCR, address 0x4001 8004 (CT32B0) and 0x4001 C004 (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	CEN		Counter enable.	0
		0	The counters are disabled.	
		1	The Timer Counter and Prescale Counter are enabled for counting.	
1	CRST		Counter reset.	0
		0	Do nothing.	
		1	The Timer Counter and the Prescale Counter are synchronously reset on the next positive edge of PCLK. The counters remain reset until TCR[1] is returned to zero.	
31:2	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.7.3 Timer Counter registers

The 32-bit Timer Counter is incremented when the Prescale Counter reaches its terminal count. Unless it is reset before reaching its upper limit, the TC will count up through the value 0xFFFF FFFF and then wrap back to the value 0x0000 0000. This event does not cause an interrupt, but a Match register can be used to detect an overflow if needed.

Table 239: Timer counter registers (TC, address 0x4001 8008 (CT32B0) and 0x4001 C008 (CT32B1)) bit description

Bit	Symbol	Description	Reset value
31:0	TC	Timer counter value.	0

14.7.4 Prescale Register

The 32-bit Prescale Register specifies the maximum value for the Prescale Counter.

Table 240: Prescale registers (PR, address 0x4001 800C (CT32B0) and 0x4001 C00C (CT32B1)) bit description

Bit	Symbol	Description	Reset value
31:0	PCVAL	Prescaler value.	0

14.7.5 Prescale Counter Register

The 32-bit Prescale Counter controls division of PCLK by some constant value before it is applied to the Timer Counter. This allows control of the relationship between the resolution of the timer and the maximum time before the timer overflows. The Prescale Counter is incremented on every PCLK. When it reaches the value stored in the Prescale Register, the Timer Counter is incremented, and the Prescale Counter is reset on the next PCLK. This causes the TC to increment on every PCLK when PR = 0, every 2 PCLKs when PR = 1, etc.

Table 241: Prescale registers (PC, address 0x4001 8010 (CT32B0) and 0x4001 C010 (CT32B1)) bit description

Bit	Symbol	Description	Reset value
31:0	PC	Prescale counter value.	0

14.7.6 Match Control Register

The Match Control Register is used to control what operations are performed when one of the Match Registers matches the Timer Counter. The function of each of the bits is shown in Table 242.

Table 242: Match Control Register (MCR, address 0x4001 8014 (CT32B0) and 0x4001 C014 (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	MR0I		Interrupt on MR0: an interrupt is generated when MR0 matches the value in the TC.	0
		1	Enabled	
		0	Disabled	
1	MR0R		Reset on MR0: the TC will be reset if MR0 matches it.	0
		1	Enabled	
		0	Disabled	
2	MR0S		Stop on MR0: the TC and PC will be stopped and TCR[0] will be set to 0 if MR0 matches the TC.	0
		1	Enabled	
		0	Disabled	
3	MR1I		Interrupt on MR1: an interrupt is generated when MR1 matches the value in the TC.	0
		1	Enabled	
		0	Disabled	
4	MR1R		Reset on MR1: the TC will be reset if MR1 matches it.	0
		1	Enabled	
		0	Disabled	

Table 242: Match Control Register (MCR, address 0x4001 8014 (CT32B0) and 0x4001 C014 (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
5	MR1S		Stop on MR1: the TC and PC will be stopped and TCR[0] will be set to 0 if MR1 matches the TC.	0
		1	Enabled	
		0	Disabled	
6	MR2I		Interrupt on MR2: an interrupt is generated when MR2 matches the value in the TC.	0
		1	Enabled	
		0	Disabled	
7	MR2R		Reset on MR2: the TC will be reset if MR2 matches it.	0
		1	Enabled	
		0	Disabled	
8	MR2S		Stop on MR2: the TC and PC will be stopped and TCR[0] will be set to 0 if MR2 matches the TC.	0
		1	Enabled	
		0	Disabled	
9	MR3I		Interrupt on MR3: an interrupt is generated when MR3 matches the value in the TC.	0
		1	Enabled	
		0	Disabled	
10	MR3R		Reset on MR3: the TC will be reset if MR3 matches it.	0
		1	Enabled	
		0	Disabled	
11	MR3S		Stop on MR3: the TC and PC will be stopped and TCR[0] will be set to 0 if MR3 matches the TC.	0
		1	Enabled	
		0	Disabled	
31:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.7.7 Match Registers

The Match register values are continuously compared to the Timer Counter value. When the two values are equal, actions can be triggered automatically. The action possibilities are to generate an interrupt, reset the Timer Counter, or stop the timer. Actions are controlled by the settings in the MCR register.

Table 243: Match registers (MR0 to 3, addresses 0x4001 8018 to 24 (CT32B0) and 0x4001 C018 to 24 (CT32B1)) bit description

Bit	Symbol	Description	Reset value
31:0	MATCH	Timer counter match value.	0

14.7.8 Capture Control Register (CCR and CCR)

The Capture Control Register is used to control whether one of the four Capture Registers is loaded with the value in the Timer Counter when the capture event occurs, and whether an interrupt is generated by the capture event. Setting both the rising and falling bits at the same time is a valid configuration, resulting in a capture event for both edges. In the description below, “n” represents the Timer number, 0 or 1.

Table 244: Capture Control Register (CCR, address 0x4001 8028 (CT32B0) and 0x4001 C028 (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	CAP0RE	1	Capture on CT32Bn_CAP0 rising edge: a sequence of 0 then 1 on CT32Bn_CAP0 will cause CR0 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
1	CAP0FE	1	Capture on CT32Bn_CAP0 falling edge: a sequence of 1 then 0 on CT32Bn_CAP0 will cause CR0 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
2	CAP0I		Interrupt on CT32Bn_CAP0 event: a CR0 load due to a CT32Bn_CAP0 event will generate an interrupt.	0
		1	Enabled.	
		0	Disabled.	
3	CAP1RE		Capture on CT32Bn_CAP1 rising edge: a sequence of 0 then 1 on CT32Bn_CAP1 will cause CR1 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
4	CAP1FE		Capture on CT32Bn_CAP1 falling edge: a sequence of 1 then 0 on CT32Bn_CAP1 will cause CR1 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
5	CAP1I		Interrupt on CT32Bn_CAP1 event: a CR1 load due to a CT32Bn_CAP1 event will generate an interrupt.	0
		1	Enabled.	
		0	Disabled.	
6	CAP2RE		Capture on CT32Bn_CAP2 rising edge: a sequence of 0 then 1 on CT32Bn_CAP2 will cause CR2 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
7	CAP2FE		Capture on CT32Bn_CAP2 falling edge: a sequence of 1 then 0 on CT32Bn_CAP2 will cause CR2 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
8	CAP2I		Interrupt on CT32Bn_CAP2 event: a CR2 load due to a CT32Bn_CAP2 event will generate an interrupt.	0
		1	Enabled.	
		0	Disabled.	

Table 244: Capture Control Register (CCR, address 0x4001 8028 (CT32B0) and 0x4001 C028 (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
9	CAP3RE		Capture on CT32Bn_CAP3 rising edge: a sequence of 0 then 1 on CT32Bn_CAP3 will cause CR3 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
10	CAP3FE		Capture on CT32Bn_CAP3 falling edge: a sequence of 1 then 0 on CT32Bn_CAP3 will cause CR3 to be loaded with the contents of TC.	0
		1	Enabled.	
		0	Disabled.	
11	CAP3I		Interrupt on CT32Bn_CAP3 event: a CR3 load due to a CT32Bn_CAP3 event will generate an interrupt.	0
		1	Enabled.	
		0	Disabled.	
31:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.7.9 Capture Register

Each Capture register is associated with a device pin and may be loaded with the Timer Counter value when a specified event occurs on that pin. The settings in the Capture Control Register register determine whether the capture function is enabled, and whether a capture event happens on the rising edge of the associated pin, the falling edge, or on both edges.

Table 245: Capture registers (CR0 to 3, addresses 0x4001 802C to 38 (CT32B0) and 0x4001 C02C to 38 (CT32B1)) bit description

Bit	Symbol	Description	Reset value
31:0	CAP	Timer counter capture value.	0

14.7.10 External Match Register

The External Match Register provides both control and status of the external match pins CAP32Bn_MAT[3:0].

Match events for Match 0 and Match 1 in each timer can cause a DMA request.

If the match outputs are configured as PWM output, the function of the external match registers is determined by the PWM rules (Section 14.7.13 “Rules for single edge controlled PWM outputs” on page 270).

Table 246: External Match Register (EMR, address 0x4001 803C (CT32B0) and 0x4001 C03C (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	EM0		External Match 0. This bit reflects the state of output CT32Bn_MAT0, whether or not this output is connected to its pin. When a match occurs between the TC and MR0, this bit can either toggle, go LOW, go HIGH, or do nothing. Bits EMR[5:4] control the functionality of this output. This bit is driven to the CT32B0_MAT0/CT32B1_MAT0 pins if the match function is selected in the IOCON registers (0 = LOW, 1 = HIGH).	0
1	EM1		External Match 1. This bit reflects the state of output CT32Bn_MAT1, whether or not this output is connected to its pin. When a match occurs between the TC and MR1, this bit can either toggle, go LOW, go HIGH, or do nothing. Bits EMR[7:6] control the functionality of this output. This bit is driven to the CT32B0_MAT1/CT32B1_MAT1 pins if the match function is selected in the IOCON registers (0 = LOW, 1 = HIGH).	0
2	EM2		External Match 2. This bit reflects the state of output CT32Bn_MAT2, whether or not this output is connected to its pin. When a match occurs between the TC and MR2, this bit can either toggle, go LOW, go HIGH, or do nothing. Bits EMR[9:8] control the functionality of this output. This bit is driven to the CT32B0_MAT2/CT32B1_MAT2 pins if the match function is selected in the IOCON registers (0 = LOW, 1 = HIGH).	0
3	EM3		External Match 3. This bit reflects the state of output CT32Bn_MAT3, whether or not this output is connected to its pin. When a match occurs between the TC and MR3, this bit can either toggle, go LOW, go HIGH, or do nothing. Bits EMR[11:10] control the functionality of this output. This bit is driven to the CT32B3_MAT0/CT32B1_MAT3 pins if the match function is selected in the IOCON registers (0 = LOW, 1 = HIGH).	0
5:4	EMC0		External Match Control 0. Determines the functionality of External Match 0.	00
		0x0	Do Nothing.	
		0x1	Clear the corresponding External Match bit/output to 0 (CT32Bi_MAT0 pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT32Bi_MAT0 pin is HIGH if pinned out).	
		0x3	Toggle the corresponding External Match bit/output.	
7:6	EMC1		External Match Control 1. Determines the functionality of External Match 1.	00
		0x0	Do Nothing.	
		0x1	Clear the corresponding External Match bit/output to 0 (CT32Bi_MAT1 pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT32Bi_MAT1 pin is HIGH if pinned out).	
		0x3	Toggle the corresponding External Match bit/output.	
9:8	EMC2		External Match Control 2. Determines the functionality of External Match 2.	00
		0x0	Do Nothing.	
		0x1	Clear the corresponding External Match bit/output to 0 (CT32Bi_MAT2 pin is LOW if pinned out).	
		0x2	Set the corresponding External Match bit/output to 1 (CT32Bi_MAT2 pin is HIGH if pinned out).	
		0x3	Toggle the corresponding External Match bit/output.	
11:10	EMC3		External Match Control 3. Determines the functionality of External Match 3.	00
		0x0	Do Nothing.	
		0x1	Clear the corresponding External Match bit/output to 0 (CT32Bi_MAT3 pin is LOW if pinned out).	

Table 246: External Match Register (EMR, address 0x4001 803C (CT32B0) and 0x4001 C03C (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
		0x2	Set the corresponding External Match bit/output to 1 (CT32Bi_MAT3 pin is HIGH if pinned out).	
		0x3	Toggle the corresponding External Match bit/output.	
31:12	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

Table 247. External match control

EMR[11:10], EMR[9:8], EMR[7:6], or EMR[5:4]	Function
00	Do Nothing.
01	Clear the corresponding External Match bit/output to 0 (CT32Bn_MATm pin is LOW if pinned out).
10	Set the corresponding External Match bit/output to 1 (CT32Bn_MATm pin is HIGH if pinned out).
11	Toggle the corresponding External Match bit/output.

14.7.10.1 DMA operation

DMA requests are generated by 0 to 1 transitions of the External Match 0 and 1 bits of each timer. In order to have an effect, the GPDMA must be configured and the relevant timer DMA request selected as a DMA source. When a timer is initially set up to generate a DMA request, the request may already be asserted before a match condition occurs. An initial DMA request may be avoided by having software write a one to clear the timer interrupt flag. A DMA request will be automatically cleared via hardware by the DMA controller after servicing.

If the EMR bits are set to 10 or 11 for channels 0 or 1 (rising edge or toggle), a DMA request is generated even if the corresponding MR register is set to 0 because a match-on-zero condition exists. To disable any DMA requests, set the EMR bits for channels 0 and 1 to 00.

14.7.11 Count Control Register

The Count Control Register (CTCR) is used to select between Timer and Counter mode, and in Counter mode to select the pin and edge(s) for counting.

When Counter Mode is chosen as a mode of operation, the CAP input (selected by the CTCR bits 3:2) is sampled on every rising edge of the PCLK clock. After comparing two consecutive samples of this CAP input, one of the following four events is recognized: rising edge, falling edge, either of edges or no changes in the level of the selected CAP input. Only if the identified event occurs, and the event corresponds to the one selected by bits 1:0 in the CTCR register, will the Timer Counter register be incremented.

Effective processing of the externally supplied clock to the counter has some limitations. Since two successive rising edges of the PCLK clock are used to identify only one edge on the CAP selected input, the frequency of the CAP input can not exceed one half of the PCLK clock. Consequently, duration of the HIGH/LOWLOW levels on the same CAP input in this case can not be shorter than 1/PCLK.

Bits 7:4 of this register are also used to enable and configure the capture-clears-timer feature. This feature allows for a designated edge on a particular CAP input to reset the timer to all zeros. Using this mechanism to clear the timer on the leading edge of an input pulse and performing a capture on the trailing edge, permits direct pulse-width measurement using a single capture input without the need to perform a subtraction operation in software.

Table 248: Count Control Register (CTCR, address 0x4001 8070 (CT32B0) and 0x4001 C070 (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
1:0	CTM		Counter/Timer Mode. This field selects which rising PCLK edges can increment Timer's Prescale Counter (PC), or clear PC and increment Timer Counter (TC). **Remark:** If Counter mode is selected in the CTCR, bits 2:0 in the Capture Control Register (CCR) must be programmed as 000.	00
		0x0	Timer Mode: every rising PCLK edge	
		0x1	Counter Mode: TC is incremented on rising edges on the CAP input selected by bits 3:2.	
		0x2	Counter Mode: TC is incremented on falling edges on the CAP input selected by bits 3:2.	
		0x3	Counter Mode: TC is incremented on both edges on the CAP input selected by bits 3:2.	
3:2	CIS		Count Input Select. In counter mode (when bits 1:0 in this register are not 00), these bits select which CAP pin or comparator output is sampled for clocking. **Remark:** If Counter mode is selected in the CTCR, the 3 bits for that input in the Capture Control Register (CCR) must be programmed as 000.	00
		0x0	CT32Bn_CAP0	
		0x1	CT32Bn_CAP1	
		0x2	CT32Bn_CAP2	
		0x3	CT32Bn_CAP3	
4	ENCC		Setting this bit to 1 enables clearing of the timer and the prescaler when the capture-edge event specified in bits 7:5 occurs.	0

Table 248: Count Control Register (CTCR, address 0x4001 8070 (CT32B0) and 0x4001 C070 (CT32B1)) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
7:5	SElCC		When bit 4 is a 1, these bits select which capture input edge will cause the timer and prescaler to be cleared. These bits have no effect when bit 4 is low.	
		0x0	Rising Edge of CAP0 clears the timer (if bit 4 is set)	
		0x1	Falling Edge of CAP0 clears the timer (if bit 4 is set)	
		0x2	Rising Edge of CAP1 clears the timer (if bit 4 is set)	
		0x3	Falling Edge of CAP1 clears the timer (if bit 4 is set)	
		0x4	Rising Edge of CAP2 clears the timer (if bit 4 is set)	
		0x5	Falling Edge of CAP2 clears the timer (if bit 4 is set)	
		0x6	Rising Edge of CAP3 clears the timer (if bit 4 is set)	
		0x7	Falling Edge of CAP3 clears the timer (if bit 4 is set)	
31:8	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

14.7.12 PWM Control Register

The PWM Control Register is used to configure the match outputs as PWM outputs. Each match output can be independently set to perform either as PWM output or as match output whose function is controlled by the External Match Register (EMR).

For each timer, a maximum of three single edge controlled PWM outputs can be selected on the MATn.2:0 outputs. One additional match register determines the PWM cycle length. When a match occurs in any of the other match registers, the PWM output is set to HIGH. The timer is reset by the match register that is configured to set the PWM cycle length. When the timer is reset to zero, all currently HIGH match outputs configured as PWM outputs are cleared.

Table 249: PWM Control Register (PWMC, 0x4001 8074 (CT32B0) and 0x4001 C074 (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
0	PWMEN0		PWM mode enable for channel0.	0
		0	CT32Bi_MAT0 is controlled by EM0.	
		1	PWM mode is enabled for CT32Bi_MAT0.	
1	PWMEN1		PWM mode enable for channel1.	0
		0	CT32Bi_MAT01 is controlled by EM1.	
		1	PWM mode is enabled for CT32Bi_MAT1.	
2	PWMEN2		PWM mode enable for channel2.	0
		0	CT32Bi_MAT2 is controlled by EM2.	
		1	PWM mode is enabled for CT32Bi_MAT2.	

Table 249: PWM Control Register (PWMC, 0x4001 8074 (CT32B0) and 0x4001 C074 (CT32B1)) bit description

Bit	Symbol	Value	Description	Reset value
3	PWMEN3		PWM mode enable for channel3. **Note:** It is recommended to use match channel 3 to set the PWM cycle.	0
		0	CT32Bi_MAT3 is controlled by EM3.	
		1	PWM mode is enabled for CT132Bi_MAT3.	
31:4	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA

14.7.13 Rules for single edge controlled PWM outputs

1. All single edge controlled PWM outputs go LOW at the beginning of each PWM cycle (timer is set to zero) unless their match value is equal to zero.
2. Each PWM output will go HIGH when its match value is reached. If no match occurs (i.e. the match value is greater than the PWM cycle length), the PWM output remains continuously LOW.
3. If a match value larger than the PWM cycle length is written to the match register, and the PWM signal is HIGH already, then the PWM signal will be cleared with the start of the next PWM cycle.
4. If a match register contains the same value as the timer reset value (the PWM cycle length), then the PWM output will be reset to LOW on the next clock tick after the timer reaches the match value. Therefore, the PWM output will always consist of a one clock tick wide positive pulse with a period determined by the PWM cycle length (i.e. the timer reload value).
5. If a match register is set to zero, then the PWM output will go to HIGH the first time the timer goes back to zero and will stay HIGH continuously.

Note: When the match outputs are selected to perform as PWM outputs, the timer reset (MRnR) and timer stop (MRnS) bits in the Match Control Register MCR must be set to zero except for the match register setting the PWM cycle length. For this register, set the MRnR bit to one to enable the timer reset when the timer value matches the value of the corresponding match register.

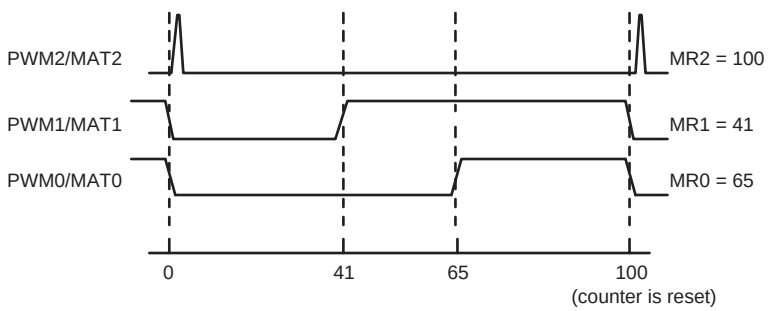


Fig 41. Sample PWM waveforms with a PWM cycle length of 100 (selected by MR3) and MAT3:0 enabled as PWM outputs by the PWCON register.

14.8 Example timer operation

Figure 42 shows a timer configured to reset the count and generate an interrupt on match. The prescaler is set to 2 and the match register set to 6. At the end of the timer cycle where the match occurs, the timer count is reset. This gives a full length cycle to the match value. The interrupt indicating that a match occurred is generated in the next clock after the timer reached the match value.

Figure 43 shows a timer configured to stop and generate an interrupt on match. The prescaler is again set to 2 and the match register set to 6. In the next clock after the timer reaches the match value, the timer enable bit in TCR is cleared, and the interrupt indicating that a match occurred is generated.

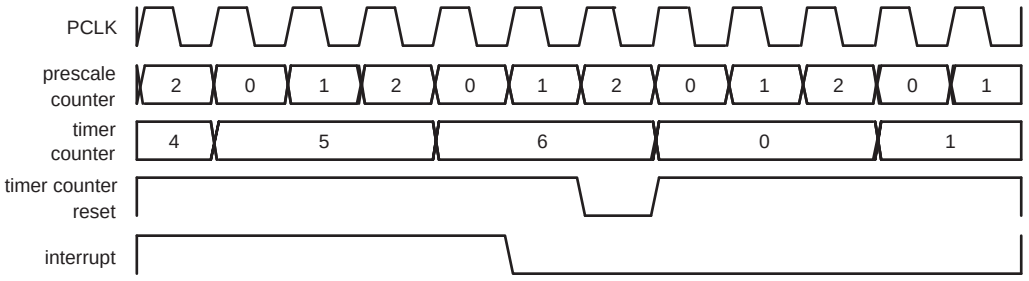


Fig 42. A timer cycle in which PR=2, MRx=6, and both interrupt and reset on match are enabled

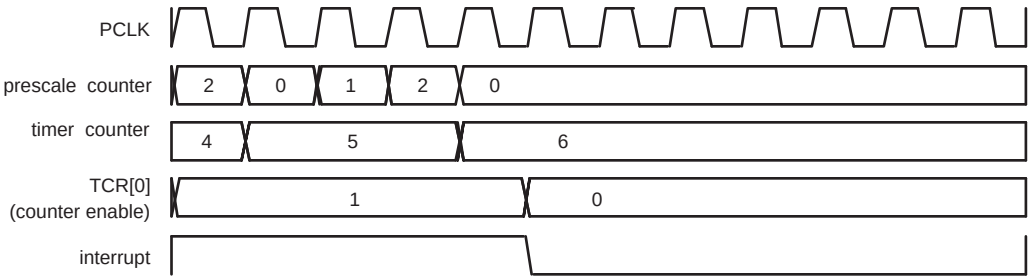


Fig 43. A timer cycle in which PR=2, MRx=6, and both interrupt and stop on match are enabled

14.9 Architecture

The block diagram for 32-bit counter/timer0 and 32-bit counter/timer1 is shown in Figure 44.

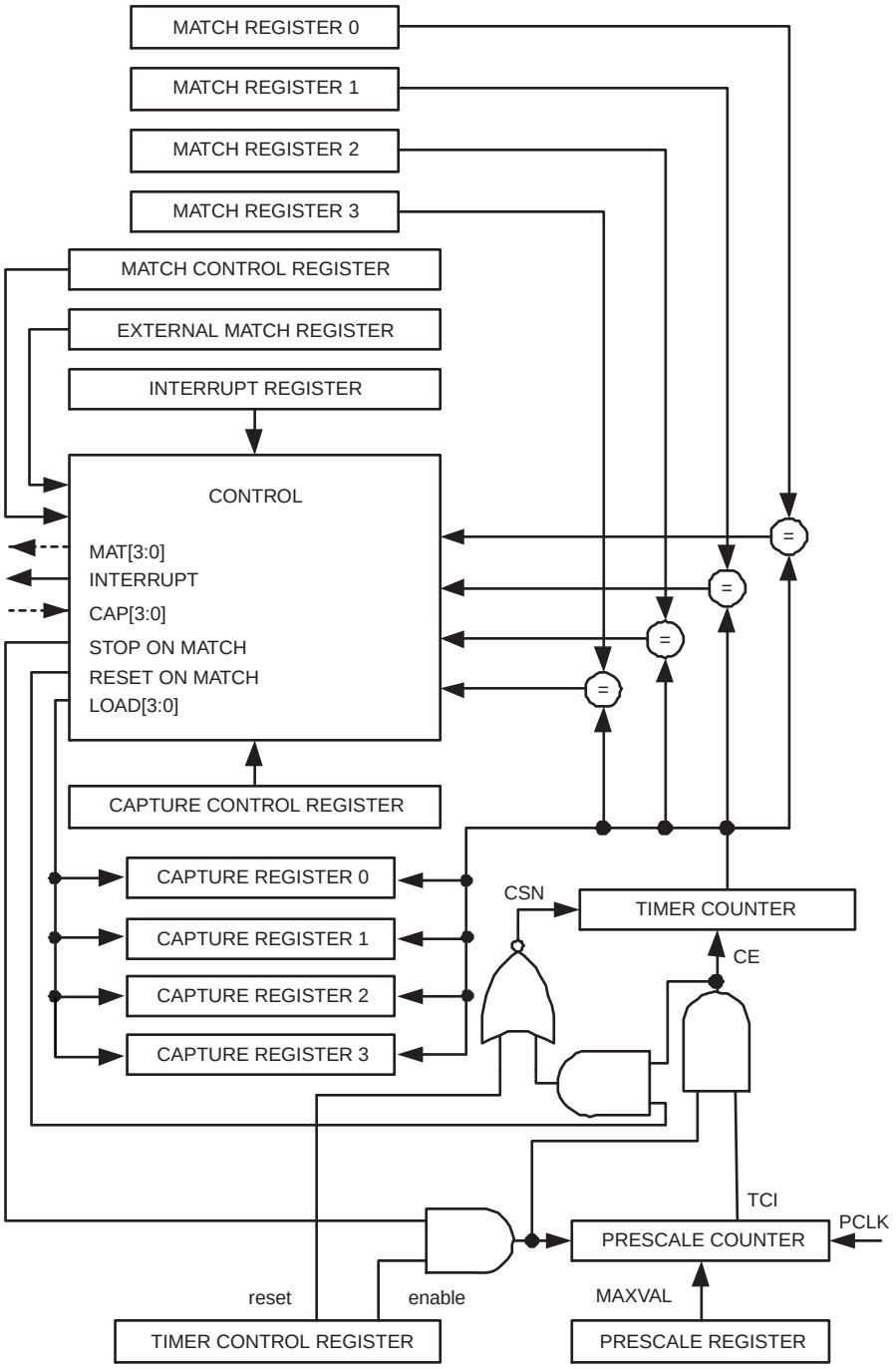


Fig 44. 32-bit counter/timer block diagram

15.1 How to read this chapter

The system tick timer (SysTick timer) is part of the ARM Cortex-M0 core and is identical for all LPC122x parts.

15.2 Basic configuration

The system tick timer is configured using the following registers:

1. Pins: The system tick timer uses no external pins.
2. Power: The system tick timer is enabled through the SysTick control register (Section 15.5.1). The system tick timer clock is fixed to half the frequency of the system clock.

15.3 Features

- Simple 24-bit timer.
- Uses dedicated exception vector.
- Clocked internally by the system clock.

15.4 General description

The block diagram of the SysTick timer is shown below in the Figure 45.

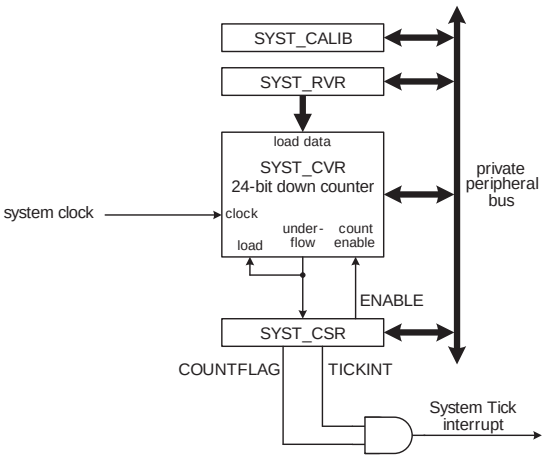


Fig 45. System tick timer block diagram

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 the *Cortex-M0 User Guide* for details.

15.5 Register description

The systick timer registers are located on the ARM Cortex-M0 private peripheral bus (see Figure 65), and are part of the ARM Cortex-M0 core peripherals. For details, see Section 25.5.4.

Table 250. 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	0x1F

[1] Reset Value reflects the data stored in used bits only. It does not include content of reserved bits.

15.5.1 System Timer Control and status register

The SYST_CSR register contains control information for the SysTick timer and provides a status flag. This register is part of the ARM Cortex-M0 core system timer register block. For a bit description of this register, see Section 25.5.4.

This register determines the clock source for the system tick timer.

Table 251. 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	Reserved	0

Table 251. SysTick Timer Control and status register (SYST_CSR - 0xE000 E010) bit description

Bit	Symbol	Description	Reset value
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

15.5.2 System Timer Reload value register

The SYST_RVR register is set to the value that will be loaded into the SysTick timer whenever it counts down to zero. This register is loaded by software as part of timer initialization. The SYST_CALIB register may be read and used as the value for SYST_RVR register if the CPU is running at the frequency intended for use with the SYST_CALIB value.

Table 252. 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

15.5.3 System Timer Current value register

The SYST_CVR register returns the current count from the System Tick counter when it is read by software.

Table 253. 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

15.5.4 System Timer Calibration value register (SYST_CALIB - 0xE000 E01C)

The value of the SYST_CALIB register is driven by the value of the SYSTCKCAL register in the system configuration block (see Section 4.5.26).

Table 254. System Timer Calibration value register (SYST_CALIB - 0xE000 E01C) bit description

Bit	Symbol	Value	Description	Reset value
23:0	TENMS		See Table 397.	0x1F
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 Table 397.	0
31	NOREF		See Table 397.	0

15.6 Functional description

The SysTick timer is a 24-bit timer that counts down to zero and generates an interrupt. The intent is to provide a fixed 10 millisecond time interval between interrupts. The SysTick timer is clocked from the CPU clock (the system clock, see Figure 3) or from the reference clock, which is fixed to half the frequency of the CPU clock. In order to generate recurring interrupts at a specific interval, the SYST_RVR register must be initialized with the correct value for the desired interval. A default value is provided in the SYST_CALIB register and may be changed by software. The default value gives a 10 millisecond interrupt rate if the CPU clock is set to <tbd> MHz.

15.7 Example timer calculations

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 LPC122x system clock set to 30 MHz.

Example (system clock = 30 MHz)

The system tick clock = system clock = 30 MHz.

RELOAD = (system tick clock frequency $\times$ 10 ms) -1 = (30 MHz $\times$ 10 ms) -1 = 300000 -1 = 299999 = 0x000493DF.

16.1 How to read this chapter

The RTC is available on all LPC122x parts.

16.2 Basic configuration

The RTC clock is controlled by the SYSCFG (Table 58) register in the PMU and can be selected from RTC internal oscillator or the main clock. Table 255 shows which clock options are available depending on the power mode of the LPC122x (see also Figure 46).

Table 255. RTC clock options

Clock	Options	Reference	Active mode	Sleep mode	Deep-sleep mode	Deep power-down mode
RTC internal oscillator	1 Hz (default), 1 kHz, delayed 1 Hz	Table 58	yes	yes	yes	yes
Main clock	Watchdog oscillator	Table 18	yes	yes	yes	no
	IRC	Table 18	yes	yes	no	no
	PLL	Table 18	yes	yes	no	no
	Input to PLL	Table 18	yes	yes	no	no

Remark: The input clock must not be changed while the RTC is running.

The RTC is powered as follows:

- The analog block of the RTC is always on as long as $V_{DD(3V3)}$ is at a valid level.
- The RTC register interface is enabled by default in the SYSAHBCLKCTRL register (Table 21).
- The RTC is enabled for creating a RTC match event using the CR register (Table 260).

16.3 Features

- Dedicated 32 kHz ultra low power oscillator.
- Uses 1 Hz clock, delayed 1 Hz clock, 1 kHz clock, or peripheral RTC clock as inputs.
- 32-bit counter.
- Programmable 32-bit match/compare register.
- Software maskable interrupt when counter and match registers are identical.
- RTC can wake up the device from Deep-sleep and Deep power-down modes.

16.4 General description

The RTC and the dedicated 32 kHz oscillator $V_{DD(3V3)}$ operate in an independent power domain, so that the RTC count and match values are maintained during reset and all low power modes as long as $V_{DD(3V3)}$ is present. The low-power analog portion including the internal counter of the RTC and the RTC oscillator are continuously running as soon as the operating voltage $V_{DD(3V3)}$ reaches a valid level. Once enabled in the CR register, the RTC can create a match event and interrupt. For the RTC clock connections, see Figure 46.

The RTC uses a 32-bit counter and a 32-bit match register. The interrupt is software maskable.

For details, see Section 16.6.

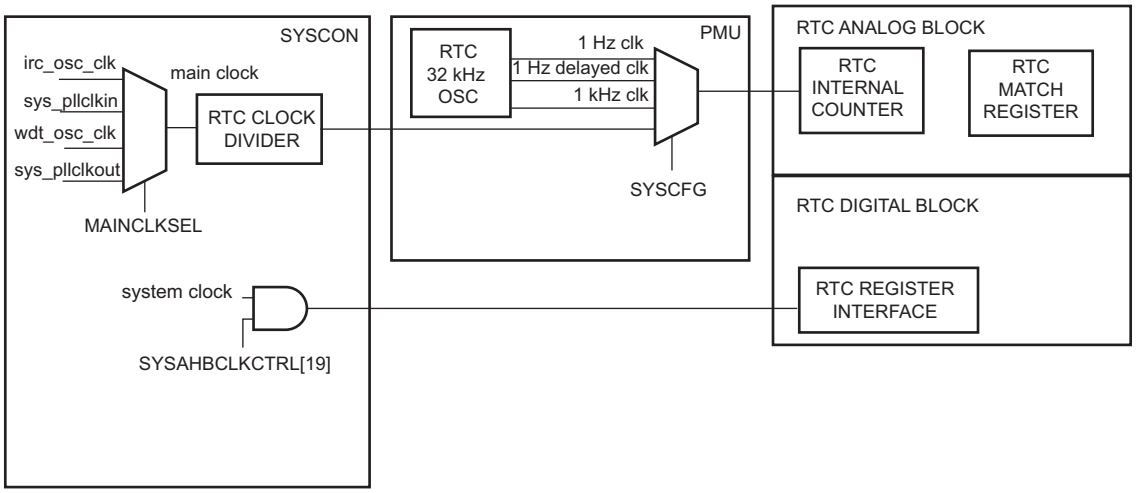


Fig 46. RTC connections to the PMU and SYSCON blocks

16.5 Register description

Table 256. Register overview: RTC (base address 0x4005 0000)

Name	Access	Address offset	Description	Reset value
DR	R	0x000	Data register	0x00
MR	R/W	0x004	Match register	0x00
LR	R/W	0x008	Load register	0x00
CR	R/W	0x00C	Control register	0x00
ICSC	R/W	0x010	Interrupt control set/clear register	0x00
RIS	R	0x014	Raw interrupt status register	0x00
MIS	R	0x018	Masked interrupt status register	0x00
ICR	W	0x01C	Interrupt clear register	0x00

16.5.1 RTC data register

This register returns the current value of the RTC counter. Note that after the RTC is enabled, a synchronization time period is in effect during which this register does not return a valid value even though the internal RTC counter counts correctly (see Section 16.6.1).

Table 257. RTC data register (DR - address 0x4005 0000) bit description

Bit	Symbol	Description	Reset value
31:0	DATA	Returns the current RTC value.	0x00

16.5.2 RTC match register

The RTC match register can be written to at any time during the initialization of the RTC and at run timer. A match event occurs when the current RTC value is identical to the content of the match register.

If the match event occurs while reset is asserted, no interrupt is created on the event and RTC misses the match event unless the match condition is tested in software after the dead-time has expired. For details, see Section 16.6.1.

Table 258. RTC match register (MR - address 0x4005 0004) bit description

Bit	Symbol	Description	Reset value
31:0	MATCH	RTC match register value.	0x00

16.5.3 RTC load register

The RTC load register allows to load or update the internal RTC counter value with any value at run-time. Writing to the load register updates the offset to the internal RTC counter.

Table 259. RTC load register (LR - address 0x4005 0008) bit description

Bit	Symbol	Description	Reset value
31:0	LOAD	RTC load register value.	0x00

16.5.4 RTC control register

This register is a R/W register. Reads return the status of the RTC. Writes enable or disable the RTC. Once the RTC is enabled, any writes to bit 0 of this register will have no effect until after a system reset.

Table 260. RTC control register (CR - address 0x4005 000C) bit description

Bit	Symbol	Value	Description	Reset value
0	RTCSTART		Enables the RTC. Once the RTC is enabled through this bit, any writes to this bit have no effect on the RTC until a power on reset (POR).	0x0
		0	RTC disabled.	
		1	RTC enabled.	
31:1	-		Reserved. Read undefined. These bits should be written as zeros.	-

16.5.5 RTC interrupt control set/clear register

This register is a R/W register and controls the masking of the interrupt generated by the RTC. Writing sets or clears the mask. Reading this register returns the current value of the mask on the RTC interrupt.

Table 261. RTC interrupt mask register (ICSC - address 0x4005 0010) bit description

Bit	Symbol	Value	Description	Reset value
0	RTCIC		Interrupt control register. A read returns the current value of the RTC control register.	0x0
		0	Writing 0 masks the interrupt.	
		1	Writing 1 enables the interrupt.	
31:1	-		Reserved. Read as zero. Do not modify these bits.	0x0

16.5.6 RTC interrupt status register

This register is a RO register. Reading this register gives the current raw status value of the corresponding interrupt prior to masking. A write has no effect.

Table 262. RTC interrupt status register (RIS - address 0x4005 0014) bit description

Bit	Symbol	Description	Reset value
0	RTCRIS	Raw interrupt event flag register. A read returns the state of the raw interrupt event flag.	0x0
31:1	-	Reserved. Read as zero.	0x0

16.5.7 RTC masked interrupt status register

This register is a RO register. Reading this register gives the current masked status value of the corresponding interrupt. A write has no effect.

Table 263. RTC masked interrupt status register (MIS - address 0x4005 0018) bit description

Bit	Symbol	Description	Reset value
0	RTCMIS	Masked interrupt register status. A read returns the masked interrupt status as controlled by the ICR register.	0x0
31:1	-	Reserved. Read as zero.	0x0

16.5.8 RTC interrupt clear register

This register is a WO register. Writing one clears the corresponding interrupt. Writing zero has no effect.

Table 264. RTC interrupt clear register (ICR - address 0x4005 001C) bit description

Bit	Symbol	Description	Reset value
0	RTCICR	Raw interrupt event flag clear register. Writing one clears the interrupt event flag. Writing 0 has no effect.	0x0
31:1	-	Reserved. Write as zero.	0x0

16.6 Functional description

16.6.1 RTC start-up behavior

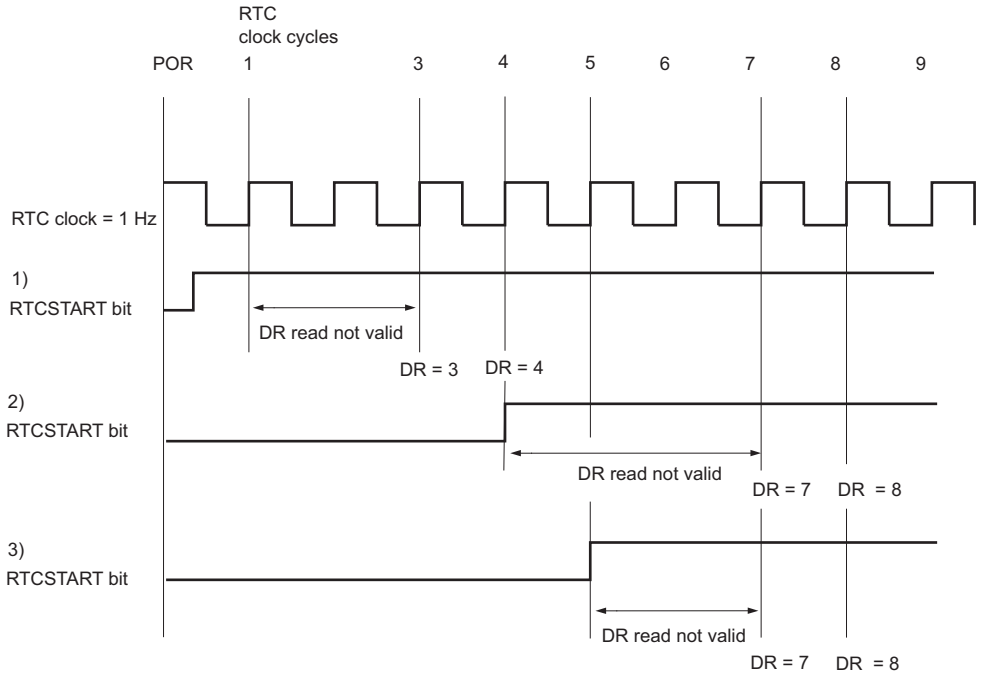
The internal counter of the RTC starts counting from 0 at POR at the default clock rate (1 Hz, see Table 58), and the RTC internal counter value is reset only by a POR. The RTC continues to count up during all other reset events (BOD, WWDT reset, software reset, RESET pin) and during Deep-power down mode as long as the chip remains powered.

However after POR, the contents of the DR register is undefined starting from the time the RTC is enabled in the CR register for a synchronization time period of up to three RTC clock cycles. Note that the internal counter does not lose time even though the DR register does not yield a valid result.

The synchronization time period for reading the DR register is two clock cycles if the RTC is enabled at an odd number of the internal RTC count and three clock cycles if the RTC is enabled at an even number of the internal RTC count.

Figure 47 shows examples of the delay introduced between the enabling the RTC in the CR register and reading a valid count from the DR register.

Remark: It is recommended to enable and initialize the RTC by software during the first clock cycle of the RTC clock after POR (example 1 in Figure 47).



(1) RTC enabled immediately after POR.

(2) RTC enable delayed for an even count of the internal RTC clock cycles.

(3) RTC enable delayed for an odd count of the internal RTC clock cycles.

Fig 47. RTC DR register read after reset

16.6.2 RTC match interrupt

A match event occurs when the match value is identical to the content of the load register. The match value is only reset during POR. If the RTC is in reset when the RTC match and count values are equal, no interrupt is created. In this case, software must check after the reset has been released if any match event has occurred.

16.6.3 Using the RTC in Deep-sleep or Deep power-down modes

The RTC can be configured to wake up the chip from Deep-sleep or Deep power-down modes when the RTC interrupt is raised. For details, see Section 4.8.3 and Section 4.9.2.

Always select one of the outputs of the RTC oscillator as RTC clock input if the RTC is used to keep time in Deep power-down mode because the main clock is not available. In Deep-sleep mode, the RTC clock should be either the watchdog oscillator or should be derived from the RTC oscillator.

Remark: To obtain a valid RTC value after waking up from Deep power-down, first perform a “dummy” read on the RTC. The next read contains the updated RTC value.

17.1 How to read this chapter

The WWDT is available on all LPC122x parts.

17.2 Basic configuration

The watchdog timer is automatically enabled by the boot ROM after reset, and the user must continue the feed sequences or disable the watchdog timer to avoid unintended reset of the device.

17.3 Features

- Internally resets chip if not reloaded during the programmable time-out period.
- 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.
- Programmable 24-bit timer with internal fixed prescaler.
- “Safe” watchdog operation. Once enabled, requires a hardware reset or a Watchdog reset to be disabled.
- The Watchdog clock (WDCLK) source can be selected from the Internal RC oscillator (IRC) or the WatchDog oscillator, see Table 13. This gives a wide range of potential timing choices for Watchdog operation under different power reduction conditions. For increased reliability, it also provides the ability to run the Watchdog timer from an entirely internal source that is not dependent on an external crystal and its associated components and wiring.
- Incorrect/Incomplete feed sequence causes reset/interrupt if enabled.
- Flag to indicate Watchdog reset.
- 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.
- The watchdog reload value can optionally be protected such that it can only be changed after the “warning interrupt” time is reached.
- The Watchdog timer can be configured to run in Deep-sleep mode.
- Debug mode.
- IEC-60335 Class B certified.

17.4 Description

The purpose of the Watchdog Timer is to reset the microcontroller within a programmable time window if the microcontroller enters an erroneous state. When enabled, a watchdog event will be generated if the user program fails to feed (or reload) the Watchdog within a predetermined amount of time. The watchdog event will cause a chip reset if configured to do so.

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-four prescaler and a 24-bit counter which decrements every clock cycle. 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 TC register.
- Setup the Watchdog timer operating mode in MOD register.
- Set a value for the watchdog window time in WINDOW register if windowed operation is required.
- Set a value for the watchdog warning interrupt in the WARNINT register if a warning interrupt is required.
- 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 in the case of 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 Clocking and power control

The watchdog timer block uses two clocks: PCLK and WDCLK. PCLK is used for the APB accesses to the watchdog registers and is derived from the system clock (see Figure 3). The WDCLK is used for the watchdog timer counting and is derived from the wdt_clk in Figure 3. Two clocks can be used as a clock source for wdt_clk clock: the IRC and the watchdog oscillator. The clock source is selected in the WDCLKSEL register (Table 271), but note that the clock source may be locked by software through the MODE register.

There is some synchronization logic between these two clock domains. 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 WDCLK clock cycles, the synchronization logic will first lock the value of the counter on WDCLK and then synchronize it with the PCLK for reading as the TV register by the CPU.

The watchdog oscillator can be powered down in the PDRUNCFG register (Section 4.5.40) if it is not used - unless bit 5 in the MOD register is set (Table 266). The clock to the watchdog register block (PCLK) can be disabled in the AHBCLKCTRL register (Table 21) for power savings.

17.6 Watchdog lock features

The watchdog timer operation can be locked in several ways to ensure that the WDT is always running. The lock features are enabled by a one-time write to the corresponding lock register bit and can only be reversed by a chip reset.

The following lock mechanisms can be applied:

- Lock the enable/disable state of the WDT and simultaneously whether the watchdog triggers an interrupt or a reset (Table 266).
- Lock the switching of clock sources. This lock mechanism prevents changing to a clock source that is powered down (Table 271).
- Lock the power control to any WDT clock source in the PDRUNCFG, PDSLEEPCFG, PDAWAKECFG registers (Table 266).
- Lock updating the WDT reload value (Table 266).
- Lock entering Deep power-down mode (Table 266).

Remark: The lock features must be used with caution.

- Ensure that the WDT clock source is selected to be powered on in all three power configuration registers PDSLEEPCFG, PDRUNCFG, and PDAWAKECFG before locking power control and the clock source select.
- The watchdog oscillator must be turned on before locking power control if the WDT is used in Deep-sleep mode.

17.7 Register description

Table 265. Register overview: Watchdog timer (base address 0x4000 4000)

Name	Access	Address offset	Description	Reset value[1]
MOD	R/W	0x000	Watchdog mode register. This register contains the basic mode and status of the Watchdog Timer.	0x0000 0003
TC	R/W	0x004	Watchdog timer constant register. This register determines the time-out value.	0x0000 FFFF
FEED	WO	0x008	Watchdog feed sequence register. Writing 0xAA followed by 0x55 to this register reloads the Watchdog timer with the value contained in TC.	NA
TV	RO	0x00C	Watchdog timer value register. This register reads out the current value of the Watchdog timer.	0xFF
CLKSEL	R/W	0x010	Watchdog clock source selection register.	0
WARNINT	R/W	0x014	Watchdog Warning Interrupt compare value.	0
WINDOW	R/W	0x018	Watchdog Window compare value.	0xFF FFFF

[1] Reset value reflects the data stored in used bits only. It does not include reserved bits content.

17.7.1 Watchdog Mode register

The MOD register controls the operation of the Watchdog as per the combination of WDEN and RESET bits.

Watchdog reset or interrupt will occur any time the watchdog is running and has an operating clock source. If a watchdog interrupt occurs in Sleep mode, it will wake up the device.

Table 266. Watchdog Mode register (MOD - 0x4000 4000) bit description

Bit	Symbol	Value	Description	Reset value
0	WDEN		Watchdog enable bit. The WDEN bit can be locked from subsequent writes by the WDLOCKEN bit.	1
		0	The watchdog timer is stopped.	
		1	The watchdog timer is running. The watchdog timer is automatically enabled at reset without requiring a valid feed sequence. Any subsequent writes to this bit require a valid feed sequence before the change can take effect.	
1	WDRESET		Watchdog reset enable bit. This bit can be changed at any time. The WDRESET bit is set by an external reset or a Watchdog timer reset. The WDRESET bit can be locked from subsequent writes by the WDLOCKEN bit.	1
		0	A watchdog time-out will cause an interrupt.	
		1	A watchdog timeout will cause a chip reset.	
2	WDTOF		Watchdog time-out flag. The Watchdog time-out flag is set when the Watchdog times out, when a feed error occurs, or when WDPROTECT =1 and an attempt is made to write to the WDTC register. This flag is cleared by software writing a 0 to this bit. Causes a chip reset if WDRESET = 1.	0 (Only after external reset)

Table 266. Watchdog Mode register (MOD - 0x4000 4000) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
3	WDINT		The Watchdog interrupt flag is set when the Watchdog counter reaches the value specified by WDWARNINT. This flag is cleared when any reset occurs, and is cleared by software by writing a 1 to this bit.	0
4	WDPROTECT		Watchdog update mode. This bit is Set Only. Once the WDPROTECT bit is set it can not be cleared by software. The WDPROTECT bit is cleared by an external reset or a Watchdog timer reset.	0
		0	The watchdog timer constant value (WDTC) can be changed at any time.	
		1	The watchdog timer constant value (WDTC) can be changed only after the counter is below the value of WARNINT and WINDOW.	
5	WDLOCKCLK		Watchdog clock lock bit. This bit is cleared on reset and can subsequently be written to only once to set it. Once this bit has been set, it can only be cleared through resetting the chip.	0
		0	The watchdog clock (WDCLK) can be disabled at any time.	
		1	Setting this bit disables any writes to the bit or bits that control the power to the currently selected watchdog clock source in the power configuration registers PDRUNCFG, PDSLEEPCFG, and PDAWAKECFG. Other bits in the power configuration registers are not affected. Setting the WDLOCKCLK bit ensures that the WDT always has a valid clock source for WDCLK provided that the watchdog oscillator and/or the IRC are powered. **Remark:** Before setting the WDLOCKCLK bit, the user must enable either the watchdog oscillator or the IRC or both in all three power configuration registers in order to keep the selected clock source running in Active, Sleep, or Deep-sleep modes. Once the WDLOCKCLK bit is set, the watchdog clock source cannot be switched off or on. **If the WDT is to be running in Deep-sleep mode, the watchdog oscillator must be enabled in the PDSLEEPCFG register before setting the WDLOCKCLK bit** (see Table 48 “Deep-sleep configuration register (PDSLEEPCFG, address 0x4004 8230) bit description”).	
6	WDLOCKDP		Deep power-down disable bit. This bit is cleared on reset and can subsequently be written to only once to set it. Once this bit has been set, it can only be cleared through resetting the chip.	0
		0	Deep power-down mode can be entered at any time.	
		1	The DPDEN bit in the PMU (see Table 56 “Power control register (PCON, address 0x4003 8000) bit description”) cannot be set to 1.	

Table 266. Watchdog Mode register (MOD - 0x4000 4000) bit description *…continued*

Bit	Symbol	Value	Description	Reset value
7	WDLOCKEN		Watchdog enable and reset lockout bit. This bit is cleared on reset and can subsequently be written to only once to set it. Once this bit has been set, it can only be cleared through resetting the chip.	
		0	The WDEN and WDRESET bits can be written to by software any time to enable or disable watchdog operation.	0
		1	If this bit is set to one, all subsequent writes to the WDEN and WDRESET bits will be blocked. The watchdog will be permanently disabled or enabled depending on the state of the WDEN bit when the WDLOCK bit was set. The reset behavior is affected as follows: • If the watchdog is enabled and the WDRESET is bit set to one before setting the WDLOCKEN bit, a watchdog trigger always causes a reset and this behavior cannot be overwritten by software. • If the watchdog is enabled and the WDRESET bit is set to zero before setting the WDLOCKEN bit, a watchdog trigger always causes an interrupt and this behavior cannot be overwritten by software.	
31:8	-		Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

Table 267. 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. The watchdog counter reaching zero will reset the microcontroller. Other causes for a watchdog reset are: A watchdog feed or changing the WDTC value (if the WDPROTECT bit is set in the MOD register) before reaching the value of WDWINDOW.

17.7.2 Watchdog Timer Constant register

The TC register determines the time-out value. Every time a feed sequence occurs, the TC content is reloaded into the Watchdog timer. This is pre-loaded with the value 0x00 FFFF upon reset. Writing values below 0xFF will cause 0xFF to be loaded into the TC. Thus the minimum time-out interval is $T_{WDCLK} \times 256 \times 4$.

If the WDPROTECT bit in MOD = 1, an attempt to change the value of TC before the watchdog counter is below the values of WARNINT and WINDOW will cause a watchdog reset and set the WDTOF flag.

Table 268. Watchdog Timer Constant register (TC - 0x4000 4004) bit description

Bit	Symbol	Description	Reset value
23:0	WDTC	Watchdog time-out interval.	0x00 FFFF
31:24	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

17.7.3 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.

Interrupts should be disabled during the feed sequence. An abort condition will occur if an interrupt happens during the feed sequence.

Table 269. Watchdog Feed register (FEED - 0x4000 4008) bit description

Bit	Symbol	Description	Reset value
7:0	Feed	Feed value should be 0xAA followed by 0x55.	-
31:8	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	-

17.7.4 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 270. Watchdog Timer Value register (TV - 0x4000 400C) 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

17.7.5 Watchdog Timer Clock Source Selection Register

This register allows selecting the clock source for the Watchdog timer. The clock source selection bits can be locked by software using bit 31 of this register, so that they cannot be modified. In addition, changes to the clock source are ignored if not both the watchdog oscillator and the IRC are powered in the PDRUNCFG register. This prevents the user from switching to a non-existing clock source.

If the WDT is running in Deep-sleep mode, the watchdog oscillator must be selected as clock source.

On reset, the clock source selection bits are always unlocked.

Table 271: Watchdog Timer Clock Source Selection register (CLKSEL - address 0x4000 4010) bit description

Bit	Symbol	Value	Description	Reset value
1:0	WDSEL		These bits select the clock source for the Watchdog timer as described below. **Warning:** Improper setting of this value may result in incorrect operation of the Watchdog timer, which could adversely affect system operation. If the WDLOCK bit in this register is set, the WDSEL bits cannot be modified. **Remark:** Writes to the WDSEL bits are ignored if the corresponding clock source is powered down in the PDRUNCFG register (Table 50).	00
		0x0	Selects the Internal RC oscillator as the Watchdog clock source (default). In active mode only.	
		0x1	Selects the watchdog oscillator as the Watchdog clock source. Must be selected if the WDT is running in Deep-sleep mode.	
		0x2	Reserved. Do not use.	
		0x3	Reserved. Do not use.	
30:2	-	-	Reserved, user software should not write ones to reserved bits. The value read from a reserved bit is not defined.	NA
31	WDLOCK		Lock Watchdog clock source.	0
		0	This bit is set to 0 on any reset. It cannot be cleared by software.	
		1	Software can set this bit to 1 at any time. Once WDLOCK is set, the bits of this register cannot be modified.	

17.7.6 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 WDWARNINT, an interrupt will be generated after the subsequent WDCLK.

A match of the watchdog timer counter to WDWARNINT 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 set to 0, the interrupt will occur at the same time as the watchdog event.

Table 272. Watchdog Timer Warning Interrupt register (WARNINT - 0x4000 4014) 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

17.7.7 Watchdog Timer Window register

The WDWINDOW register determines the highest WDTV value allowed when a watchdog feed is performed. If a feed valid sequence completes prior to WDTV reaching the value in WDWINDOW, a watchdog event will occur.

WDWINDOW resets to the maximum possible WDTV value, so windowing is not in effect. Values of WDWINDOW below 0x100 will make it impossible to ever feed the watchdog successfully.

Table 273. Watchdog Timer Window register (WINDOW - 0x4000 4018) 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

17.8 Block diagram

The block diagram of the Watchdog is shown below in the Figure 48. The synchronization logic (PCLK - WDCLK) is not shown in the block diagram.

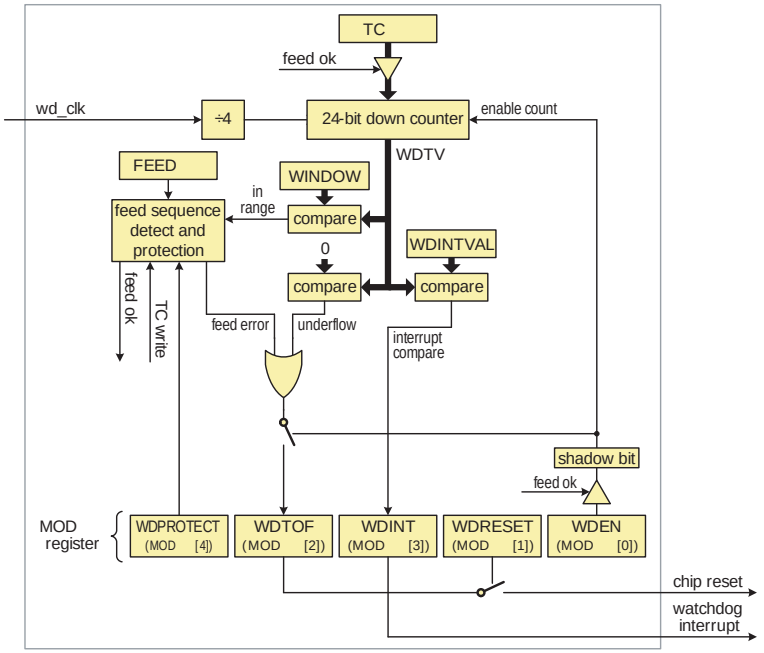


Fig 48. Watchdog block diagram

17.9 Watchdog timing examples

The following figures illustrate several aspects of Watchdog Timer operation is shown below in Figure 49.

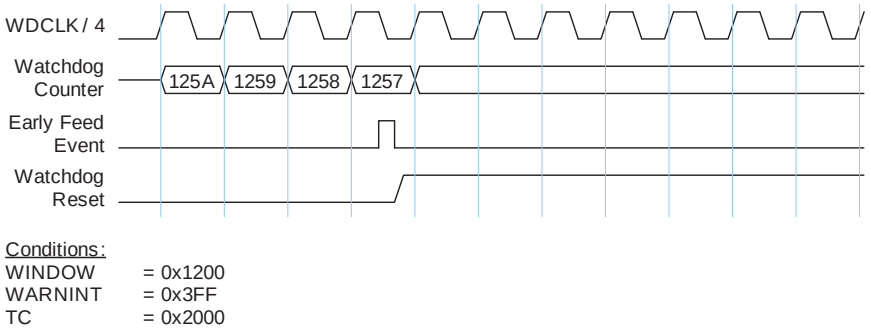


Fig 49. Early Watchdog Feed with Windowed Mode Enabled

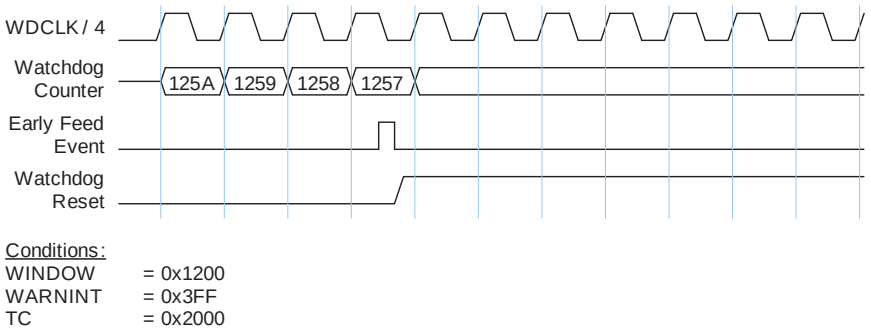


Fig 50. Correct Watchdog Feed with Windowed Mode Enabled

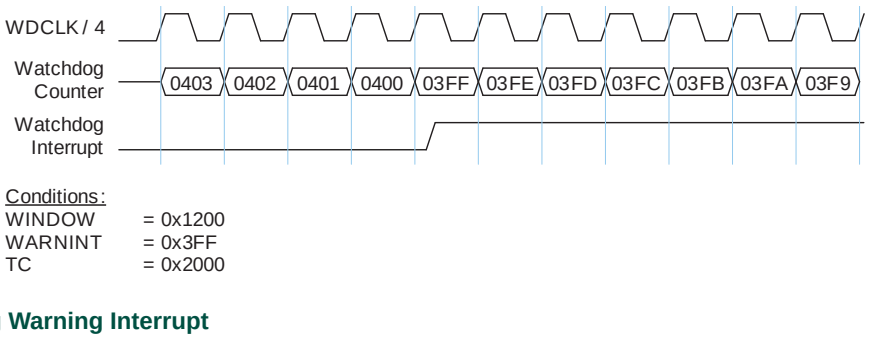


Fig 51. Watchdog Warning Interrupt

18.1 How to read this chapter

The comparator is available on all LPC122x parts.

18.2 Basic configuration

The peripheral clock to the comparator block is provided by the system clock, which is controlled by the SYSAHBCLKDIV register (Table 21). The comparator block can be disabled through the System AHB clock control register bit 20 (Table 21) and the PDRUNCFG register bit 15 (Table 50) for power savings.

Remark: The BOD must be enabled in the PDRUNCFG register (Table 50) in order to use the comparator. If the comparator is used to wake up the part from Deep-sleep mode, the BOD and the comparator must be powered in the PDSLEEPCFG register by overwriting the recommended settings for this register.

18.3 Features

- Up to six selectable external sources per comparator; fully configurable on either positive or negative comparator input channels.
- BOD 0.9 V internal reference voltage selectable on both comparators; configurable on either positive or negative comparator input channels.
- 32-stage voltage ladder internal reference voltage selectable on both comparators; configurable on either positive or negative comparator input channels.
- Voltage ladder source voltage is selectable from an external pin or the 3.3 V voltage rail if the external power source is not available.
- Voltage ladder can be separately powered down for applications only requiring the comparator function.
- Relaxation oscillator circuitry output, for a feedback 555 style timer application.
- Individual comparator interrupts connected to I/O pins, common interrupt connected to NVIC.
- Edge and level comparator outputs connect to two timers allowing edge tick counting while a level match has been asserted.

18.4 General description

Two embedded comparators are incorporated on-chip to compare the voltage levels on external pins or against internal voltages. Up to six voltages on external pins and two internal reference voltages are selectable on each comparator. Additionally, four of the external input voltages can be selected to drive an input common on both comparators in case identical voltages are required on both comparators (see Figure 52).

18.5 Pin description

Table 274. Comparator pin description

Pin	Type	Description
ACMP0_I[3:0]	input	Comparator 0 input sources (ACMP0_I[0] and ACMP0_I[1] can be programmed for Comparator 1 inputs)
ACMP1_I[3:0]	input	Comparator 1 input sources (ACMP1_I[0] and ACMP1_I[1] can be programmed for Comparator 0 inputs)
ACMP0_O	output	Comparator 0 output
ACMP1_O	output	Comparator 1 output
VREF_CMP	input	External reference voltage source for 32-stage Voltage Ladder
ROSC	output	Relaxation oscillator output, intended for 555 timer style applications

18.6 Register description

Table 275. Register overview: Comparator (base address 0x4005 4000)

Name	Access	Address offset	Description	Reset value
CMP	R/W	0x00	Comparator control register	0
VLAD	R/W	0x04	Voltage ladder register	0

18.6.1 Comparator control register

This register enables the comparators, configures the interrupts, and controls the input multiplexer to both comparators.

Table 276. Comparator control register (CMP, address 0x4005 4000) bit description

Bit	Symbol	Value	Description	Reset value
0	CMP0_EN		Comparator 0 enable.	0
		0	Comparator 0 disabled.	
		1	Comparator 0 enabled.	
1	CMP1_EN		Comparator 1 enable.	0
		0	Comparator 1 disabled.	
		1	Comparator 1 enabled.	
2	CMPIS		Select interrupt source.	0
		0	Edge triggered.	
		1	Level triggered.	
3	CMPIEV		Select edge triggered interrupt to be active on either high or low transitions.	0
		0	Interrupt active on falling edges.	
		1	Interrupt active on rising edges.	
4	CMPBE		Select edge triggered interrupt to be active on both edges.	0
		0	Interrupt not active on both edges.	
		1	Interrupt active on both edges.	

Table 276. Comparator control register (CMP, address 0x4005 4000) bit description

Bit	Symbol	Value	Description	Reset value
5	CMPSR		Select orientation of the two comparators in the relaxation oscillator circuit.	0
6	CMPSA0		Select async/sync output of the comparator 0.	0
		0	The comparator output is used directly.	
		1	The comparator output is synchronized with the bus clock for output to other modules.	
7	CMPSA1		Select async/sync output of the comparator 1.	0
		0	The comparator output is used directly.	
		1	The comparator output is synchronized with the bus clock for output to other modules.	
10:8	CMP0_VP_CTRL		Selection of comparator 0, positive voltage (VP) input channel.	000
		0x0	Voltage ladder output	
		0x1	ACMP0_I0	
		0x2	ACMP0_I1	
		0x3	ACMP0_I2	
		0x4	ACMP0_I3	
		0x5	ACMP1_I0	
		0x6	ACMP1_I1	
		0x7	BOD 0.9 V band gap	
13:11	CMP0_VM_CTRL		Selection of comparator 0, negative voltage (VM) input channel.	000
		0x0	Voltage ladder output	
		0x1	ACMP0_I0	
		0x2	ACMP0_I1	
		0x3	ACMP0_I2	
		0x4	ACMP0_I3	
		0x5	ACMP1_I0	
		0x6	ACMP1_I1	
		0x7	BOD 0.9 V band gap	
16:14	CMP1_VP_CTRL		Selection of comparator 1, positive voltage (VP) input channel.	000
		0x0	Voltage ladder output	
		0x1	ACMP1_I0	
		0x2	ACMP1_I1	
		0x3	ACMP1_I2	
		0x4	ACMP1_I3	
		0x5	ACMP0_I0	
		0x6	ACMP0_I1	
		0x7	BOD 0.9 V band gap	

Table 276. Comparator control register (CMP, address 0x4005 4000) bit description

Bit	Symbol	Value	Description	Reset value
19:17	CMP1_VM_CTRL		Selection of comparator 1, negative voltage (VM) input channel.	000
		0x0	Voltage ladder output	
		0x1	ACMP1_I0	
		0x2	ACMP1_I1	
		0x3	ACMP1_I2	
		0x4	ACMP1_I3	
		0x5	ACMP0_I0	
		0x6	ACMP0_I1	
		0x7	BOD 0.9 V band gap	
20	INTCLR		Interrupt clear bit. Setting this bit to one clears the comparator interrupt.	
21	CMP0STAT		Comparator 0 status. This bit reflects the state of the comparator 0 output.	
22	CMP1STAT		Comparator 1 status. This bit reflects the state of the comparator 1 output.	
31:23	-		Reserved.	0

18.6.2 Voltage ladder register

This register enables the voltage ladder for the comparator reference input. The reference input VREF_CMP is programmable in 32 levels from V_{SS} to VREF_CMP = 3.3 V.

Table 277. Voltage ladder register (VLAD, address 0x4005 4004) bit description

Bit	Symbol	Value	Description	Reset value
0	VLADEN		Voltage ladder enable	0
		0	Voltage ladder disabled.	
		1	Voltage ladder enabled.	
5:1	VSEL		Voltage ladder value. The reference voltage Vref depends on the setting of bit 6 in this register (either $V_{DD(3V3)}$ or voltage on pin VREF_CMP): 0000 = V_{SS} 0001 = 1 × Vref/31 0010 = 2 × Vref/31 ... 11111 = Vref	0
6	VLADREF		Voltage reference select	1
		0	VREF_CMP pin	
		1	$V_{DD(3V3)}$ pin	
31:7	-		Unused	0

18.7 Functional description

18.7.1 Input multiplexer

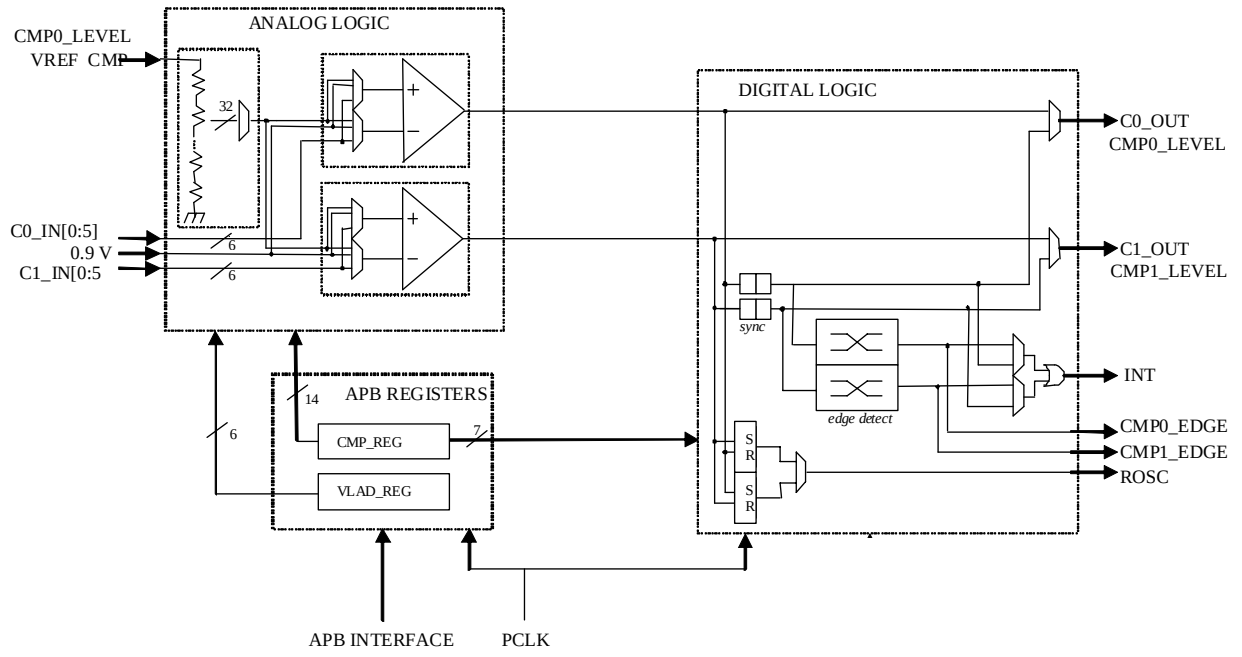


Fig 52. Comparator block diagram

The two comparators each have 8 inputs multiplexed separately to both their positive and negative inputs. The multiplexers for each comparator (both positive and negative inputs) are all controlled by the comparator register CMP (see Figure 52 and Table 276).

Bit 0 of the positive and negative inputs on each comparator can be selected to derive from a programmable voltage ladder output.

Bit 7 of the positive and negative inputs on each comparator can be selected that drives from the on-chip bandgap voltage.

The remaining 6 comparator inputs for the positive and negative sides of the comparators come from external chip IO pins. However due to a limitation on availability, only four IO pins are dedicated for each comparator. The two remaining inputs for each comparator can be selected from alternate comparators inputs if desired; ACMP0_I0 and ACMP0_I1 can be selected for ACMP1_I4 and ACMP1_I5 input sources respectively, similarly ACMP1_I0 and ACMP1_I1 can selected for ACMP0_I4 and ACMP0_I5 input sources respectively (see Figure 53)

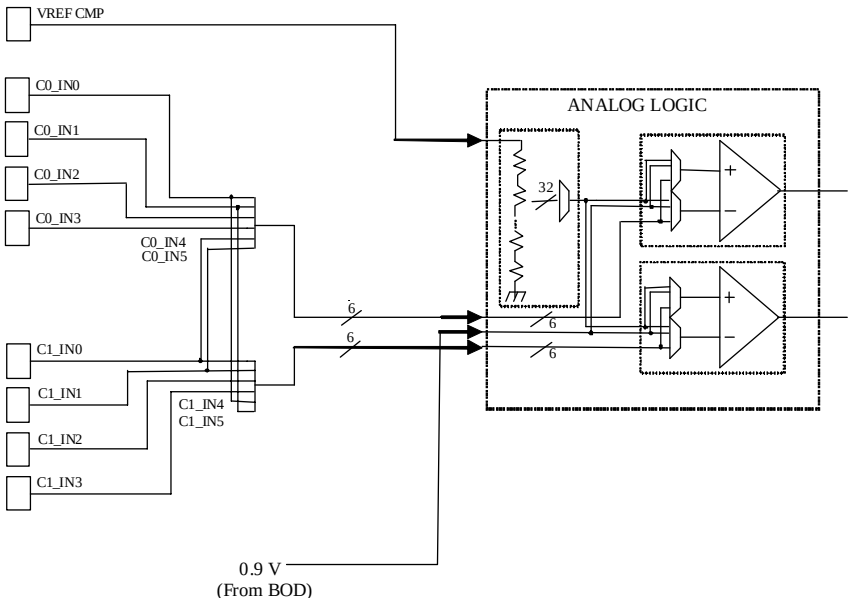


Fig 53. Comparator inputs

18.7.2 Reference voltages

The comparator can use two reference voltages: the external reference voltage VREF_CMP and the internal bandgap voltage, the BOD 0.9 V reference voltage. The external reference voltage can be used with a programmable voltage ladder, which is programmable in 32 levels from the VREF_CMP IO reference voltage down to VSS. The maximum voltage for VREF_CMP is 3.3 V.

18.7.3 Relaxation oscillator

The digital circuitry includes a relaxation oscillator facility where the ROSC pin of the chip can be routed externally back to the comparator input thereby creating oscillation that could be used as a 555 timer.

The 555 has three operating modes:

1. Monostable mode: in this mode, the 555 timer produces one single pulse. The pulse width is configurable by an external RC network.
2. Astable - free running mode: in this mode, the 555 timer produces a continuous stream of rectangular pulses. The frequency of the oscillation is determined by two external resistors and a capacitor.
3. Bistable mode or Schmitt trigger: the 555 can operate as a flip-flop if the timing capacitor pin is not connected and no capacitor is used.

Applications for the relaxation oscillator include precision timing, pulse generation, sequential timing, time delay generation, pulse width generation, pulse position generation, and missing pulse detection.

18.7.4 Interrupts

The interrupt can be selected to be edge or level style. If level is selected, the interrupt leaving this block is synchronized first to the peripheral clock domain to prevent an asynchronous interrupt path crashing the CPU. If edge level interrupts are selected, a choice of active high, active low or active both edges can be selected. Interrupts are cleared by the CPU writing CLINT high.

The combined ORed interrupts of comparator0 and comparator1 are routed to the NVIC for ISR purposes using the CPU.

18.7.5 Comparator outputs

The two comparator level outputs are routed to external pins. The level and edge comparator outputs are also internally connected to the capture inputs of the two 16-bit timers. The outputs can be selected in synchronous or asynchronous modes (see Table 276):

- For internal connection to the timer capture inputs, synchronous or asynchronous modes can be selected.
- When the asynchronous outputs are routed to external pins, the comparators can be used without clocking PCLK to save power once the device is configured.
- When the comparator is configured to wake up the part from Deep-sleep mode, the asynchronous mode must be selected.

In addition, the status of each comparator output can be observed through the comparator status register bits

The level and edge outputs of each comparator are routed to the two 16-bit timers internally as listed below:

- 16-bit timer 0 (CT16B0):
  - capture input 3: comparator 0 edge
  - capture input 2: comparator 0 level
- 16-bit timer 1 (CT16B1):
  - capture input 3: comparator 1 edge
  - capture input 2: comparator 1 level

This feature allows tick-counting on comparator output transitions on either the positive edge, the negative edge, or both edges (depending on the comparator register configuration), if the edge capture is chosen. If the timer level capture is chosen, the counter can run while the comparator output is in a given state.

19.1 How to read this chapter

The ADC is available on all LPC12xx parts.

19.2 Basic configuration

Clocks and power to the ADC are controlled by the SYSAHBCLKDIV register (see Table 20). This clock can be disabled through bit 14 in the AHBCLKCTRL register (Table 21) for power savings.

For connecting to the DMA, see Section 19.7.3.

The ADC can be powered down at run-time through the PDRUNCFG register, see Table 50. However, when using the ADC, both the ADC_PD and the BOD_PD bits must be set to 0 in the PDRUNCFG register. That is, both the ADC and the BOD must be powered to perform ADC conversions.

19.3 Features

- 10-bit successive approximation Analog-to-Digital Converter (ADC).
- Input multiplexing among 8 pins.
- Power-down mode.
- Measurement range 0 to 3 V.
- 10-bit conversion time of 257 kHz.
- Burst conversion mode for single or multiple inputs.
- Optional conversion on transition on input pin or Timer Match signal.
- Individual result registers for each A/D channel to reduce interrupt overhead.

19.4 General description

Basic clocking for the A/D converters is provided by the system clock. A programmable divider is included in each ADC to scale this clock to the 9 MHz (max) clock needed by the successive approximation process. A fully accurate conversion requires 35 of these clocks.

19.5 Pin description

Table 278 gives a brief summary of each of ADC related pins.

Table 278. ADC pin description

Pin	Type	Description
AD[7:0]	Input	**Analog Inputs.** The A/D converter cell can measure the voltage on any of these input signals. The input signal must no exceed V_{REF} (typically 3.3 V).
$V_{DD(3V3)}$	Input	V_{REF}; Reference voltage.

The ADC function must be selected via the IOCON registers in order to get accurate voltage readings on the monitored pin. 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.

19.6 Register description

Table 279. Register overview: ADC (base address 0x4002 0000)

Name	Access	Address offset	Description	Reset value
CR	R/W	0x000	A/D Control Register. The CR register must be written to select the operating mode before A/D conversion can occur.	0x0000 0000
GDR	R/W	0x004	A/D Global Data Register. Contains the result of the most recent A/D conversion.	NA
-	-	0x008	reserved	-
INTEN	R/W	0x00C	A/D Interrupt Enable Register. This register contains enable bits that allow the DONE flag of each A/D channel to be included or excluded from contributing to the generation of an A/D interrupt.	0x0000 0100
DR0	R/W	0x010	A/D Channel 0 Data Register. This register contains the result of the most recent conversion completed on channel 0	NA
DR1	R/W	0x014	A/D Channel 1 Data Register. This register contains the result of the most recent conversion completed on channel 1.	NA
DR2	R/W	0x018	A/D Channel 2 Data Register. This register contains the result of the most recent conversion completed on channel 2.	NA
DR3	R/W	0x01C	A/D Channel 3 Data Register. This register contains the result of the most recent conversion completed on channel 3.	NA
DR4	R/W	0x020	A/D Channel 4 Data Register. This register contains the result of the most recent conversion completed on channel 4.	NA
DR5	R/W	0x024	A/D Channel 5 Data Register. This register contains the result of the most recent conversion completed on channel 5.	NA
DR6	R/W	0x028	A/D Channel 6 Data Register. This register contains the result of the most recent conversion completed on channel 6.	NA
DR7	R/W	0x02C	A/D Channel 7 Data Register. This register contains the result of the most recent conversion completed on channel 7.	NA
STAT	RO	0x030	A/D Status Register. This register contains DONE and OVERRUN flags for all of the A/D channels, as well as the A/D interrupt flag.	0
TRM	R/W	0x034	A/D trim register	0

19.6.1 A/D Control Register

The A/D Control Register provides bits to select A/D channels to be converted, A/D timing, A/D modes, and the A/D start trigger.

Table 280. A/D Control Register (CR - address 0x4002 0000) bit description

Bit	Symbol	Value	Description	Reset value
7:0	SEL		Selects which of the AD7:0 pins is (are) to be sampled and converted. For ADC, bit 0 selects Pin AD0, and bit 7 selects pin AD7. In software-controlled mode, only one of these bits should be 1. In hardware scan mode, any value containing 1 to 8 ones. All zeroes is equivalent to 0x01.	0
15:8	CLKDIV		The APB clock (PCLK) is divided by (this value plus one) to produce the clock for the A/D converter, which should be less than or equal to 9 MHz. Typically, software should program the smallest value in this field that yields a clock of 9 MHz or slightly less, but in certain cases (such as a high-impedance analog source) a slower clock may be desirable.	0
16	BURST		Burst mode control.	0
		0	Conversions are software controlled and require 36 clocks.	
		1	The AD converter does repeated conversions up to 250 kHz, scanning (if necessary) through the pins selected by 1s in the SEL field. The first conversion after the start corresponds to the least-significant 1 in the SEL field, then higher numbered 1 bits (pins) if applicable. Repeated conversions can be terminated by clearing this bit, but the conversion that's in progress when this bit is cleared will be completed. **Important:** START bits must be 000 when BURST = 1 or conversions will not start.	
23:17	-		Reserved. These bits always read as zeros.	0
26:24	START		Conversion start control. When the BURST bit is 0, these bits control whether and when an A/D conversion is started.	0
		0x0	No start (this value should be used when clearing PDN to 0).	
		0x1	Start conversion now.	
		0x2	Start conversion when the edge selected by bit 27 occurs on PIO0_2/SSEL/CT16B0_CAP0.	
		0x3	Start conversion when the edge selected by bit 27 occurs on PIO1_5/DIR/CT32B0_CAP0.	
		0x4	Start conversion when the edge selected by bit 27 occurs on CT32B0_MAT0.	
		0x5	Start conversion when the edge selected by bit 27 occurs on CT32B0_MAT1.	
		0x6	Start conversion when the edge selected by bit 27 occurs on CT16B0_MAT0.	
		0x7	Start conversion when the edge selected by bit 27 occurs on CT16B0_MAT1.	
27	EDGE		Edge control. This bit is significant only when the START field contains 010-111.	0
		1	Start conversion on a falling edge on the selected CAP/MAT signal.	
		0	Start conversion on a rising edge on the selected CAP/MAT signal.	
31:28	-		Reserved. These bits always read as zeros.	0

19.6.2 A/D Global Data Register

The A/D Global Data Register contains the result of the most recent A/D conversion. This includes the data, DONE, and Overrun flags, and the number of the A/D channel to which the data relates.

Table 281. A/D Global Data Register (GDR - address 0x4002 0004) bit description

Bit	Symbol	Description	Reset value
5:0	-	Reserved. These bits always read as zeros.	0
15:6	RESULT	When DONE is 1, this field contains a binary fraction representing the voltage on the ADn pin selected by the SEL field, divided by the voltage on the $V_{DD(3V3)}$ pin: V/V_{REF}. Zero in the field indicates that the voltage on the ADn pin was less than, equal to, or close to that on V_{SS}, while 0x3FF indicates that the voltage on ADn was close to, equal to, or greater than that on V_{REF}.	X
23:16	-	Reserved. These bits always read as zeros.	0
26:24	CHN	These bits contain the channel from which the RESULT bits were converted.	X
29:27	-	Reserved. These bits always read as zeros.	0
30	OVERRUN	This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the RESULT bits.	0
31	DONE	This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read and when the ADCR is written. If the ADCR is written while a conversion is still in progress, this bit is set and a new conversion is started.	0

19.6.3 A/D Interrupt Enable Register

This register allows control over which A/D channels generate an interrupt when a conversion is complete. For example, it may be desirable to use some A/D channels to monitor sensors by continuously performing conversions on them. The most recent results are read by the application program whenever they are needed. In this case, an interrupt is not desirable at the end of each conversion for some A/D channels.

Table 282. A/D Interrupt Enable Register (INTEN - address 0x4002 000C) bit description

Bit	Symbol	Description	Reset value
7:0	ADINTEN	These bits allow control over which A/D channels generate interrupts for conversion completion. When bit 0 is one, completion of a conversion on A/D channel 0 will generate an interrupt, when bit 1 is one, completion of a conversion on A/D channel 1 will generate an interrupt, etc.	0x00
8	ADGINTEN	When 1, enables the global DONE flag in ADDR to generate an interrupt. When 0, only the individual A/D channels enabled by ADINTEN 7:0 will generate interrupts.	1
31:9	-	Reserved. These bits always read as zeros.	0

19.6.4 A/D Data Registers

The A/D Data Register hold the result when an A/D conversion is complete, and also include the flags that indicate when a conversion has been completed and when a conversion overrun has occurred.

Table 283. A/D Data Registers (DR0 to DR7 - addresses 0x4002 0010 to 0x4002 002C) bit description

Bit	Symbol	Description	Reset value
5:0	-	Reserved. These bits always read as zeros.	0
15:6	RESULT	When DONE is 1, this field contains a binary fraction representing the voltage on the ADn pin selected by the SEL field, divided by the voltage on the $V_{DD(3V3)}$ pin: V/V_{REF}. Zero in the field indicates that the voltage on the ADn pin was less than, equal to, or close to that on V_{SSA}, while 0x3FF indicates that the voltage on ADn was close to, equal to, or greater than that on V_{REF}.	X
29:16	-	Reserved. These bits always read as zeros.	0
30	OVERRUN	This bit is 1 in burst mode if the results of one or more conversions was (were) lost and overwritten before the conversion that produced the result in the RESULT bits.This bit is cleared by reading this register.	0
31	DONE	This bit is set to 1 when an A/D conversion completes. It is cleared when this register is read.	0

19.6.5 A/D Status Register

The A/D Status register allows checking the status of all A/D channels simultaneously. The DONE and OVERRUN flags appearing in the ADDRn register for each A/D channel are mirrored in ADSTAT. The interrupt flag (the logical OR of all DONE flags) is also found in ADSTAT.

Table 284. A/D Status Register (STAT - address 0x4002 0030) bit description

Bit	Symbol	Description	Reset value
7:0	DONE	These bits mirror the DONE status flags that appear in the result register for each A/D channel.	0
15:8	OVERRUN	These bits mirror the OVERRRUN status flags that appear in the result register for each A/D channel. Reading ADSTAT allows checking the status of all A/D channels simultaneously.	0
16	ADINT	This bit is the A/D interrupt flag. It is one when any of the individual A/D channel Done flags is asserted and enabled to contribute to the A/D interrupt via the ADINTEN register.	0
31:17	-	Reserved. These bits always read as zeros.	0

19.6.6 A/D Trim register

This register will be set by the bootcode on start-up. It contains the trim values for the ADC. The offset trim values for the ADC can be overwritten by the user.

Table 285. A/D Trim register (TRM - address 0x4002 4034) bit description

Bit	Symbol	Description	Reset value
3:0	-	Reserved. These bits always read as zeros.	0
7:4	ADCOFFS	Offset trim bits for ADC operation. Initialized by the boot code. Can be overwritten by the user.	0
31:8	-	Reserved. These bits always read as zeros.	0

19.7 Operation

Once an ADC conversion is started, it cannot be interrupted. A new software write to launch a new conversion or a new edge-trigger event will be ignored while the previous conversion is in progress.

19.7.1 Hardware-triggered conversion

If the BURST bit in the ADCR is 0 and the START field contains 010-111, the ADC will start a conversion when a transition occurs on a selected pin or Timer Match signal. The choices include conversion on a specified edge of any of 4 Match signals, or conversion on a specified edge of either of 2 Capture/Match pins. The pin state from the selected pad or the selected Match signal, XORed with ADCR bit 27, is used in the edge detection logic.

19.7.2 Interrupts

An interrupt request is asserted to the NVIC when the DONE bit is 1. Software can use the Interrupt Enable bit for the A/D Converter in the NVIC to control whether this assertion results in an interrupt. DONE is negated when the ADDR is read.

19.7.3 DMA control

A DMA transfer request is generated from the ADC interrupt request line. To generate a DMA transfer, the same conditions must be met as the conditions for generating an interrupt (see Section 19.6.3 and Section 19.7.2).

Remark: If the DMA is used, the ADC interrupt must be **disabled** in the NVIC.

For DMA transfers, the transfer size can be set to one in the DMA channel control structure (see Table 347) or to the number of ADC channels that are converted. The DMA transfer size determines when a DMA interrupt is generated.

20.1 How to read this chapter

See Table 286 for different flash configurations.

Table 286. LPC122x flash configurations

Type number	Flash
LPC1227	
LPC12D27FBD100/301	128 kB
LPC1227FBD64/301	128 kB
LPC1227FBD48/301	128 kB
LPC1226	
LPC1226FBD64/301	96 kB
LPC1226FBD48/301	96 kB
LPC1225	
LPC1225FBD64/321	80 kB
LPC1225FBD64/301	64 kB
LPC1225FBD48/321	80 kB
LPC1225FBD48/301	64 kB
LPC1224	
LPC1224FBD64/121	48 kB
LPC1224FBD64/101	32 kB
LPC1224FBD48/121	48 kB
LPC1224FBD48/101	32 kB

20.2 Features

- In-System Programming: In-System programming (ISP) is programming or reprogramming the on-chip flash memory, using the boot loader software and UART0 serial port. This can be done when the part resides in the end-user board.
- In-Application Programming: In-Application (IAP) programming is performing erase and write operation on the on-chip flash memory, as directed by the end-user application code.

20.3 Boot loader

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

20.4 Applications

The boot loader provides both In-System and In-Application programming interfaces for programming the on-chip flash memory.

20.5 Description

The boot loader code is executed every time the part is powered on or reset. The loader can execute the ISP command handler or the user application code. A LOW level after reset at the PIO0_12 pin is considered as an external hardware request to start the ISP command handler. Assuming that power supply pins are on their nominal levels when the rising edge on $\overline{\text{RESET}}$ pin is generated, it may take up to 3 ms before PIO0_12 is sampled and the decision on whether to continue with user code or ISP handler is made. If PIO0_12 is sampled low and the watchdog overflow flag is set, the external hardware request to start the ISP command handler is ignored. If there is no request for the ISP command handler execution (PIO0_12 is sampled HIGH after reset), a search is made for a valid user program. If a valid user program is found then the execution control is transferred to it. If a valid user program is not found, the auto-baud routine is invoked.

Pin PIO0_12 that is used as hardware request for ISP requires special attention. Since PIO0_12 is in high impedance mode after reset, it is important that the user provides external hardware (a pull-up resistor or other device) to put the pin in a defined state. Otherwise unintended entry into ISP mode may occur.

20.5.1 Memory map after any reset

The boot block is 8 kB in size. The boot block is located in the memory region starting from the address 0x1FFF 0000. The boot loader is designed to run from this memory area, but both the ISP and IAP software use parts of the on-chip RAM. The RAM usage is described later in this chapter. 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.

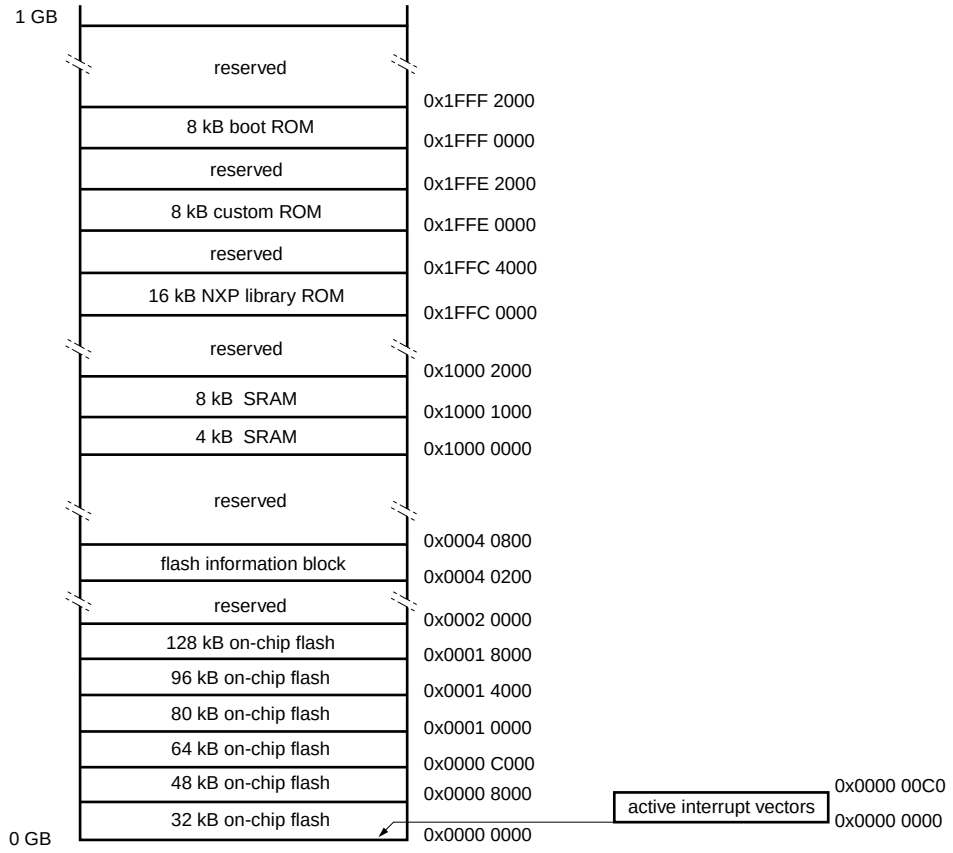


Fig 54. Memory map of flash and boot ROM memory area

20.5.2 Flash sectors

The LPC122x flash block is divided into pages with a page size of 512 byte. A page is the smallest flash area that can be erased. In addition, virtual sectors are used for software compatibility with existing ISP and IAP commands. The flash block is divided into up to 32 virtual sectors (see Table 287). The size of each virtual sector is 4 kB. A virtual sector consists of eight 512-byte pages. Erase operations can be performed on virtual sector or page boundaries.

Table 287. Flash configuration of the LPC122x

Page number (8 pages per sector)	Virtual sector number (4 kB per sector)	Address range	Flash size					
			32 kB	48 kB	64 kB	80 kB	96 kB	128 kB
0 - 63	0 -7	0x0000 0000 - 0x0000 7FFF	yes	yes	yes	yes	yes	yes
64 - 95	8 - 11	0x0000 8000 - 0x0000 BFFF	-	yes	yes	yes	yes	yes
96 - 127	12 - 15	0x0000 8000 - 0x0000 FFFF	-	-	yes	yes	yes	yes
128 - 149	16 - 19	0x0001 0000 - 0x0001 3FFF	-	-	-	yes	yes	yes
150 - 181	20 - 23	0x0001 4000 - 0x0001 7FFF	-	-	-	-	yes	yes
182 - 255	24 - 31	0x0001 8000 - 0x0001 FFFF	-	-	-	-	-	yes

20.5.2.1 Information block

The flash memory area also contains a flash information block starting at address 0x0004 0200 with a total size of 3 pages (see Figure 54). This flash area is available to the user for storing variables, configuration parameters, and other user-defined data.

A read access by the CPU at the information block address returns the data contained in this location. Pages in the information block can be erased by issuing the IAP command “Erase info page”, which is specific to the information block. The IAP “Copy RAM to flash” command then can be used to write new user data to the information block.

20.5.2.2 Erase operations for the flash block

ISP and IAP commands allow to prepare one or more sectors for erase operations and to erase those sectors.

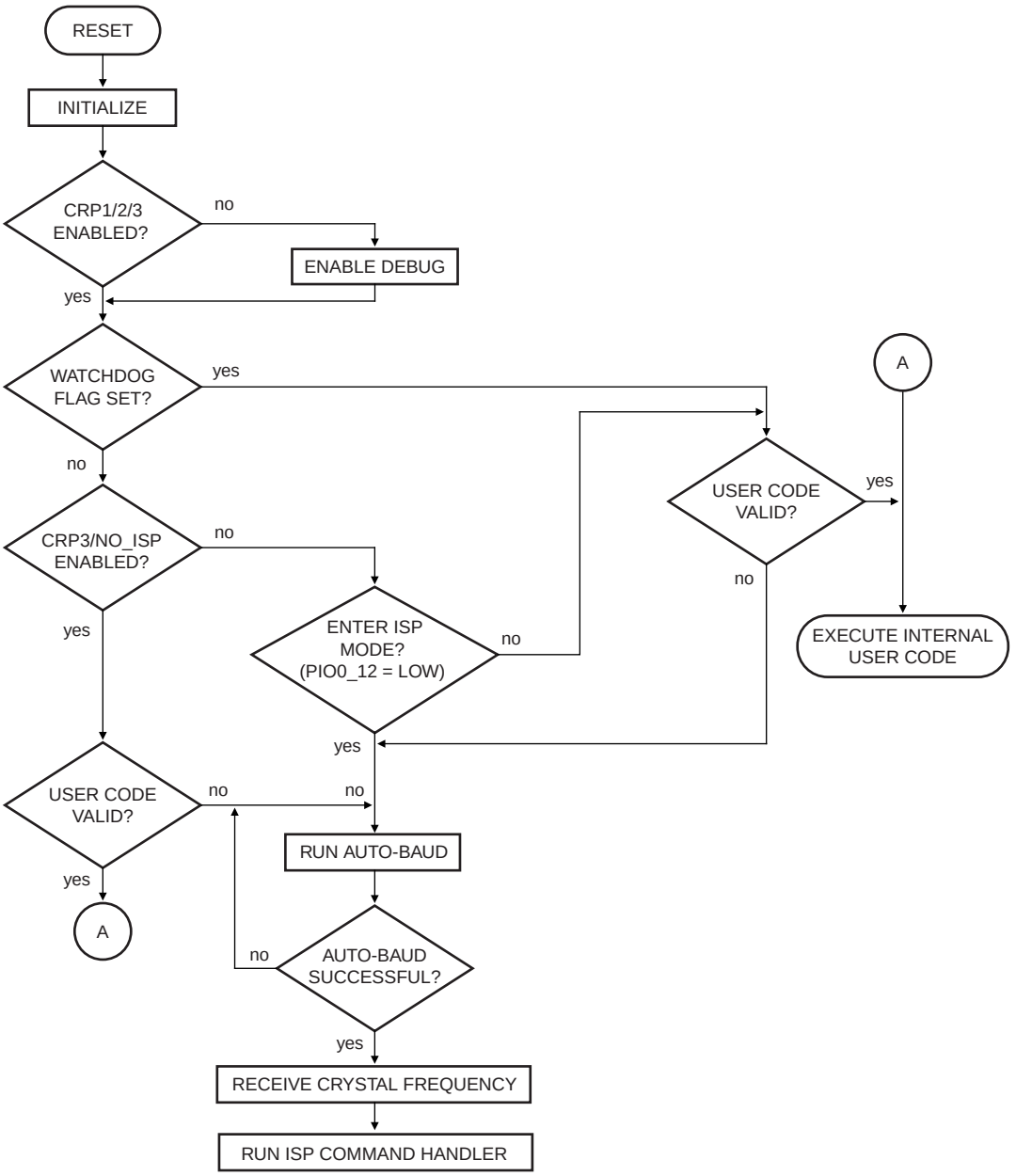
In addition, the IAP commands include a page erase command, which allows erasing a range of pages in the user flash memory regardless of sector boundaries.

On the LPC122x, the erase time increases with the number of pages or sectors to be erased. If all sectors of the flash area are selected to be erased, the ISP and IAP routines automatically use a more efficient “mass erase” procedure, which shortens the erase time significantly. For details on the flash timing parameters, see the *LPC122x data sheet*.

20.5.2.3 Write operations for the flash block

Write-to-flash operations using the ISP and IAP “Copy RAM to flash” commands enable the user code to write to flash. The minimum write size is 4 bytes, equivalent to one flash word. The total programming time is proportional to the number of flash words written. However, writing a whole row of 32 flash words or 128 bytes at a time is more efficient. The “Copy RAM to flash” commands automatically detect row aligned addresses and use a more efficient write procedure to achieve shorter row programming times.

20.5.3 Boot process flowchart



(1) For details on handling the crystal frequency, see Section 20.8.8 “Reinvoke ISP” on page 327

Fig 55. Boot process flowchart

20.5.4 Criterion for Valid User Code

Criterion for valid user code: The reserved Cortex-M0 exception vector location 7 (offset 0x 0000 001C in the vector table) should contain the 2’s complement of the check-sum of table entries 0 through 6. This causes the checksum of the first 8 table entries to be 0. The boot loader code checksums the first 8 locations in sector 0 of the flash. If the result is 0, then execution control is transferred to the user code.

If the signature is not valid, the auto-baud routine synchronizes with the host via serial port 0. The host should send a '?' (0x3F) as a synchronization character and wait for a response. The host side serial port settings should be 8 data bits, 1 stop bit and no parity. The auto-baud routine measures the bit time of the received synchronization character in terms of its own frequency and programs the baud rate generator of the serial port. It also sends an ASCII string ("Synchronized<CR><LF>") to the Host. In response to this host should send the same string ("Synchronized<CR><LF>"). The auto-baud routine looks at the received characters to verify synchronization. If synchronization is verified then "OK<CR><LF>" string is sent to the host. Host should respond by sending the crystal frequency (in kHz) at which the part is running. For example, if the part is running at 10 MHz, the response from the host should be "10000<CR><LF>". "OK<CR><LF>" string is sent to the host after receiving the crystal frequency. If synchronization is not verified then the auto-baud routine waits again for a synchronization character. For auto-baud to work correctly in case of user invoked ISP, the CCLK frequency should be greater than or equal to 10 MHz.

Once the crystal frequency is received the part is initialized and the ISP command handler is invoked. For safety reasons an "Unlock" command is required before executing the commands resulting in flash erase/write operations and the "Go" command. The rest of the commands can be executed without the unlock command. The Unlock command is required to be executed once per ISP session. The Unlock command is explained in Section 20.7 "ISP commands" on page 315.

20.5.5 Communication protocol

All 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 UU-encoded format.

20.5.5.1 ISP command format

"Command Parameter_0 Parameter_1 ... Parameter_n<CR><LF>" "Data" (Data only for Write commands).

20.5.5.2 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).

20.5.5.3 ISP data format

The data stream is in UU-encode format. The UU-encode algorithm converts 3 bytes of binary data in to 4 bytes of printable ASCII character set. It is more efficient than Hex format which converts 1 byte of binary data in to 2 bytes of ASCII hex. The sender should send the check-sum after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters (bytes) i.e. it can hold 45 data bytes. The receiver should compare it with the check-sum of the received bytes. If the check-sum matches then the receiver should respond with "OK<CR><LF>" to continue further transmission. If the check-sum does not match the receiver should respond with "RESEND<CR><LF>". In response the sender should retransmit the bytes.

A description of UU-encode is available at the wotsit webpage.

20.5.5.4 ISP flow control

A software XON/XOFF flow control scheme is used to prevent data loss due to buffer overrun. When the data arrives rapidly, the ASCII control character DC3 (stop) is sent to stop the flow of data. Data flow is resumed by sending the ASCII control character DC1 (start). The host should also support the same flow control scheme.

20.5.5.5 ISP command abort

Commands can be aborted by sending the ASCII control character "ESC". This feature is not documented as a command in Section 20.7. Once the escape code is received the ISP command handler waits for a new command.

20.5.5.6 Interrupts during ISP

The boot block interrupt vectors located in the boot block of the flash are active after any reset.

20.5.5.7 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. The user should either disable interrupts, or ensure that user interrupt vectors are active in RAM and that the interrupt handlers reside in RAM, before making a flash erase/write IAP call. The IAP code does not use or disable interrupts.

20.5.5.8 RAM used by ISP command handler

ISP commands use on-chip RAM from 0x1000 017C to 0x1000 025B. The user could use this area, but the contents may be lost upon reset. Flash programming commands use the top 32 bytes of on-chip RAM. The stack is located at RAM top − 32. The maximum stack usage is 256 bytes and it grows downwards.

20.5.5.9 RAM used by IAP command handler

Flash programming commands use the top 32 bytes of on-chip RAM. The maximum stack usage in the user allocated stack space is 128 bytes and it grows downwards.

20.6 Code Read Protection (CRP)

Code Read Protection is a mechanism that allows the user to enable different levels of security in the system so that access to the on-chip flash and use of the ISP can be restricted. When needed, CRP is invoked by programming a specific pattern in flash location at 0x0000 02FC. IAP commands are not affected by the code read protection.

Important: any CRP change becomes effective only after the device has gone through a power cycle.

Table 288. Code Read Protection options

Name	Pattern programmed in 0x0000 02FC	Description
NO_ISP	0x4E69 7370	Prevents sampling of pin PIO0_12 for entering ISP mode. PIO0_12 is available for other uses. Flash contents are not protected, and the flash can be reprogrammed through the SWD interface.
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 cannot access RAM below 0x1000 0300. • Copy RAM to flash command can not write to Sector 0. • Erase command can erase Sector 0 only when all sectors are selected for erase. • Compare command is disabled. • Read Memory command is disabled. 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 PIO0_12 LOW is disabled if a valid user code is present in flash sector 0. This mode effectively disables ISP override using PIO0_12 pin. It is up to the user's application to provide a flash update mechanism using IAP calls or call reinvoke ISP command to enable flash update via UART0. **Caution: If CRP3 is selected, no future factory testing can be performed on the device.**

Table 289. Code Read Protection hardware/software interaction

CRP option	User Code Valid	PIO0_12 pin at reset	SWD enabled	LPC122x enters ISP mode	partial flash update in ISP mode
No	No	x	Yes	Yes	Yes
No	Yes	High	Yes	No	NA
No	Yes	Low	Yes	Yes	Yes
CRP1	Yes	High	No	No	NA
CRP1	Yes	Low	No	Yes	Yes
CRP2	Yes	High	No	No	NA

Table 289. Code Read Protection hardware/software interaction

CRP option	User Code Valid	PIO0_12 pin at reset	SWD enabled	LPC122x enters ISP mode	partial flash update in ISP mode
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 290. 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 one of the CRP modes 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.

20.6.1 ISP entry protection

In addition to the three CRP modes, the user can prevent the sampling of pin PIO0_12 for entering ISP mode without enabling CRP and thereby release pin PIO0_12 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.

In NO_ISP mode, the flash can be reprogrammed through the SWD interface. Flash contents are not protected in this mode.

20.7 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 the 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 291. ISP command summary

ISP Command	Usage	Described in
Unlock	U <Unlock Code>	Table 292
Set Baud Rate	B <Baud Rate> <stop bit>	Table 293
Echo	A <setting>	Table 294
Write to RAM	W <start address> <number of bytes>	Table 295
Read Memory	R <address> <number of bytes>	Table 296
Prepare sector(s) for write operation	P <start sector number> <end sector number>	Table 297
Copy RAM to flash	C <Flash address> <RAM address> <number of bytes>	Table 298
Go	G <address> <Mode>	Table 299
Erase sector(s)	E <start sector number> <end sector number>	Table 300
Blank check sector(s)	I <start sector number> <end sector number>	Table 301
Read Part ID	J	Table 302
Read Boot code version	K	Table 304
Compare	M <address1> <address2> <number of bytes>	Table 305
ReadUID	N	Table 306

20.7.1 Unlock <Unlock code>

Table 292. 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, and Go commands.

20.7.2 Set Baud Rate <Baud Rate> <stop bit>

Table 293. 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.

20.7.3 Echo <setting>

Table 294. 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.

20.7.4 Write to RAM <start address> <number of bytes>

The host should send the data only after receiving the CMD_SUCCESS return code. The host should send the check-sum after transmitting 20 UU-encoded lines. The checksum is generated by adding raw data (before UU-encoding) bytes and is reset after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters (bytes) i.e. it can hold 45 data bytes. When the data fits in less then 20 UU-encoded lines then the check-sum should be of the actual number of bytes sent. The ISP command handler compares it with the check-sum of the received bytes. If the check-sum matches, the ISP command handler responds with "OK<CR><LF>" to continue further transmission. If the check-sum does not match, the ISP command handler responds with "RESEND<CR><LF>". In response the host should retransmit the bytes.

Table 295. 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. Data should be in UU-encoded format. This command is blocked when code read protection is enabled.
Example	"W 268436224 4<CR><LF>" writes 4 bytes of data to address 0x1000 0300.

20.7.5 Read Memory <address> <no. of bytes>

The data stream is followed by the command success return code. The check-sum is sent after transmitting 20 UU-encoded lines. The checksum is generated by adding raw data (before UU-encoding) bytes and is reset after transmitting 20 UU-encoded lines. The length of any UU-encoded line should not exceed 61 characters (bytes) i.e. it can hold 45 data bytes. When the data fits in less then 20 UU-encoded lines then the check-sum is of actual number of bytes sent. The host should compare it with the checksum of the received bytes. If the check-sum matches then the host should respond with "OK<CR><LF>" to continue further transmission. If the check-sum does not match then the host should respond with "RESEND<CR><LF>". In response the ISP command handler sends the data again.

Table 296. 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 (UU-encoded)> | 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 or the information block. 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.

20.7.6 Prepare sector(s) for write operation <start sector number> <end sector number>

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

Table 297. 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.

20.7.7 Copy RAM to flash <Flash address> <RAM address> <no of bytes>

Table 298. ISP Copy command

Command	C
Input	**Flash Address(DST):** Destination flash address where data bytes are to be written. The destination address should be on a flash word (4 bytes) boundary. Writes on a row boundary (128 bytes) are preferred as they can be performed more efficiently. The total number of bytes copied must be ≤ 4096 bytes. **RAM Address(SRC):** Source RAM address from where data bytes are to be read. **Number of Bytes:** Number of bytes to be written. Must be a multiple of 4 bytes.
Return Code	CMD_SUCCESS | SRC_ADDR_ERROR (Address not on word boundary) | DST_ADDR_ERROR (Address not on a word boundary correct boundary) | SRC_ADDR_NOT_MAPPED | DST_ADDR_NOT_MAPPED | COUNT_ERROR (Byte count is not 4 bytes) | 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. This command is blocked when code read protection is enabled.
Example	"C 0 268467504 512<CR><LF>" copies 512 bytes from the RAM address 0x1000 0800 to the flash address 0.

20.7.8 Go <address> <mode>

Table 299. 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) | A (Execute program in ARM 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.
Example	"G 0 A<CR><LF>" branches to address 0x0000 0000 in ARM mode.

20.7.9 Erase sector(s) <start sector number> <end sector number>

Table 300. 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.

20.7.10 Blank check sector(s) <sector number> <end sector number>

Table 301. 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 512 bytes are mapped to the ROM boot block.**
Example	"I 2 3<CR><LF>" blank checks the flash sectors 2 and 3.

20.7.11 Read Part Identification number

Table 302. ISP Read Part Identification command

Command	J
Input	None.
Return Code	CMD_SUCCESS followed by part identification number in ASCII (see Table 303).
Description	This command is used to read the part identification number.

Table 303. LPC122x part identification numbers

Device	Hex coding
LPC12D27FBD100/301	0x3670 002B
LPC1227FBD64/301	0x3670 002B
LPC1227FBD48/301	0x3670 002B
LPC1226FBD64/301	0x3660 002B
LPC1226FBD48/301	0x3660 002B
LPC1225FBD64/321	0x3652 002B
LPC1225FBD64/301	0x3650 002B
LPC1225FBD48/321	0x3652 002B
LPC1225FBD48/301	0x3650 002B
LPC1224FBD64/121	0x3642 C02B
LPC1224FBD64/101	0x3640 C02B
LPC1224FBD48/121	0x3642 C02B
LPC1224FBD48/101	0x3640 C02B

20.7.12 Read Boot code version number

Table 304. 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.

20.7.13 Compare <address1> <address2> <no of bytes>

Table 305. 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. **Compare result may not be correct when source or destination address contains any of the first 512 bytes starting from address zero. First 512 bytes are re-mapped to boot ROM**
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.

20.7.14 ReadUID

Table 306. 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.

20.7.15 ISP Return Codes

Table 307. 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.

20.8 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 if 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.

The IAP function could be called in the following way using C.

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 long command[5];
unsigned long result[4];
```

or

```
unsigned long * command;
unsigned long * result;
command=(unsigned long *) 0x..
result= (unsigned long *) 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 function pointer:

```
iap_entry=(IAP) IAP_LOCATION;
```

Whenever you wish to call IAP you could use the following statement.

```
iap_entry (command, 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 308. IAP Command Summary

IAP Command	Command Code (decimal)	Described in
Prepare sector(s) for write operation	50	Table 309
Copy RAM to flash	51	Table 310
Erase sector(s)	52	Table 311
Blank check sector(s)	53	Table 312
Read Part ID	54	Table 313
Read Boot code version	55	Table 314
Compare	56	Table 315
Reinvoke ISP	57	Table 316
Read UID	58	Table 317
Erase page	59	Table 318
Erase info page	60	Table 319

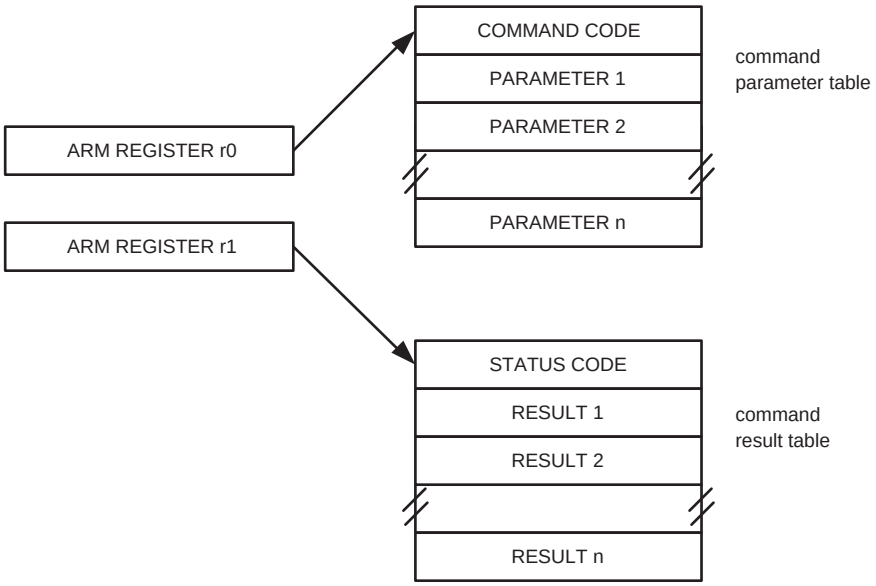


Fig 56. IAP parameter passing

20.8.1 Prepare sector(s) for write operation

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

Table 309. 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).
Return 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. To prepare a page or pages for erasing, use this command and specify the sector or sectors that contain the pages to be erased.

20.8.2 Copy RAM to flash

Table 310. 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 must be on a 4 byte boundary. The destination address should be on a flash word (4 bytes) boundary. Writes on a row boundary (128 bytes) are preferred as they can be performed more efficiently. The total number of bytes copied must be $\leq$ 4096 bytes. **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. Must be a multiple of 4 bytes. **Param3:** System Clock Frequency (CCLK) in kHz.
Return 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 a multiple of 4) | 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.

20.8.3 Erase Sector(s)

Table 311. IAP Erase Sector(s) command

Command	Erase Sector(s)
Input	**Command code: 52 (decimal)** **Param0:** Start Sector Number **Param1:** End Sector Number (should be greater than or equal to start sector number). **Param2:** System Clock Frequency (CCLK) in kHz.
Return 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.

20.8.4 Blank check sector(s)

Table 312. 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).
Return 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.

20.8.5 Read Part Identification number

Table 313. IAP Read Part Identification command

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

20.8.6 Read Boot code version number

Table 314. IAP Read Boot Code version number command

Command	Read boot code version number
Input	**Command code: 55 (decimal)** **Parameters:** None
Return Code	CMD_SUCCESS |
Result	**Result0:** 2 bytes of boot code version number. It is to be interpreted as <byte1(Major)>.<byte0(Minor)>
Description	This command is used to read the boot code version number.

20.8.7 Compare <address1> <address2> <no of bytes>

Table 315. 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.
Return Code	CMD_SUCCESS | COMPARE_ERROR | COUNT_ERROR (Byte count is not a multiple of 4) | ADDR_ERROR | ADDR_NOT_MAPPED
Result	**Result0:** Offset of the first mismatch if the Status Code is COMPARE_ERROR.
Description	This command is used to compare the memory contents at two locations. **The result may not be correct when the source or destination includes any of the first 512 bytes starting from address zero. The first 512 bytes can be re-mapped to RAM.**

20.8.8 Reinvoke ISP

Table 316. IAP Reinvoke ISP

Command	Reinvoke ISP
Input	**Command code: 57 (decimal)**
Return Code	None
Result	None.
Description	This command is used to invoke the boot loader in ISP mode. It maps boot vectors, sets PCLK = CCLK, configures UART0 pins RXD0 and TXD0, resets counter/timer CT32B1 and resets the U0FDR (see Table 154). This command may be used when a valid user program is present in the internal flash memory and the PIO0_12 pin is not accessible to force the ISP mode.

20.8.9 ReadUID

Table 317. IAP ReadUID command

Command	ReadUID
Input	**Command code: 58 (decimal)**
Return 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.

20.8.10 Erase page

Table 318. IAP Erase page command

Command	Erase page
Input	**Command code: 59 (decimal)** **Param0:** Erase start page. **Param1:** Erase end page (should be greater than or equal to start page) **Param2:** System Clock Frequency (CCLK) in kHz.
Return 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.

20.8.11 Erase info page

Table 319. IAP Erase info page command

Command	Erase info page
Input	**Command code: 60 (decimal)** **Param0:** Erase start page in information block. **Param1:** Erase end page in information block (should be greater than or equal to start page) **Param2:** System Clock Frequency (CCLK) in kHz.
Return 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 the information block. The information block consists of three pages, numbered 0, 1, and 2. To erase a single page use the same start and end page numbers.

20.8.12 IAP Status Codes

Table 320. 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. Invalid information page address.
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 or page 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.

20.9 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.

21.1 How to read this chapter

The micro DMA controller is available on all LPC122x parts.

21.2 Introduction

The micro DMA controller is a very low-gate-count DMA compatible with the AMBA AHB-Lite protocol for DMA transfers. The micro DMA registers are programmed through the APB interface.

21.3 Features

- Single AHB-Lite master for transferring data using a 32-bit address bus and 32-bit data bus.
- 21 DMA channels plus one channel for memory-to-memory transfers.
- Dedicated handshake signals and programmable priority level for each channel.
- Each priority level arbitrates using a fixed priority that is determined by the DMA channel number.
- Supports memory-to-memory, memory-to-peripheral, and peripheral-to-memory transfers.
- Supports multiple DMA cycle types and multiple DMA transfer widths.
- Each DMA channel can access a primary and an alternate channel control data structure.
- The channel control data is stored in system memory in little-endian format.
- Performs all DMA transfers using single AHB-Lite transfers. Burst transfers are not supported.
- The destination data width is equal to the source data width.
- The number of transfers in a single DMA cycle can be programmed from 1 to 1024.
- The transfer address increment can be greater than the data width.
- Single output to indicate when an error condition occurs on the AHB.

21.4 Description

The micro DMA controller contains an APB register interface, the AHB-Lite master interface to the AHB multi-layer matrix, and the DMA control block (see Figure 57).

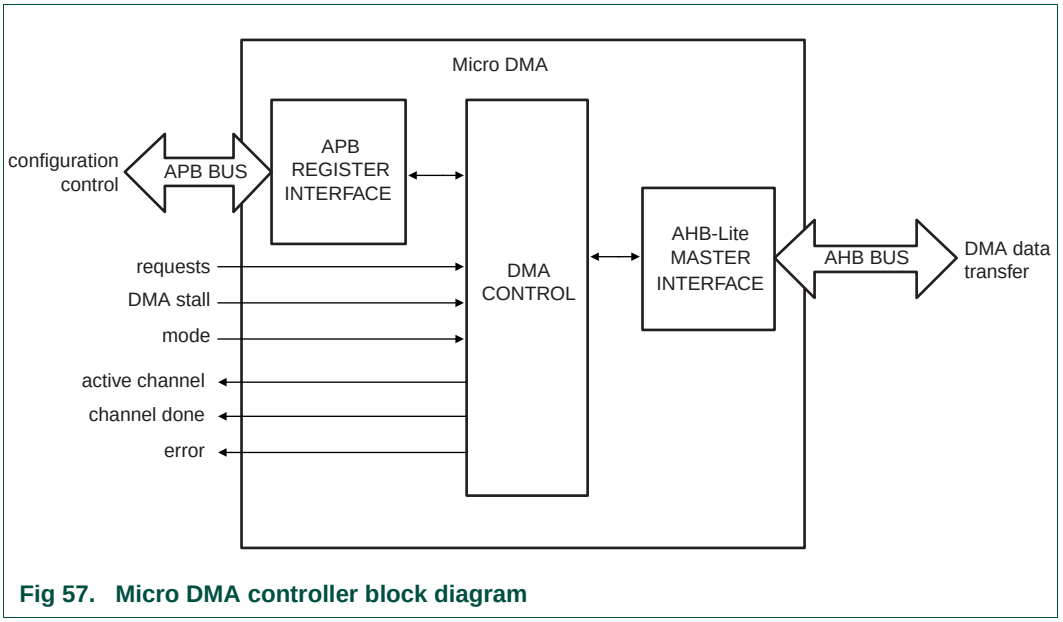


Fig 57. Micro DMA controller block diagram

The DMA controller supports data transfer sizes of 8, 16, or 32 bit (byte, half-word, or word), configured through the channel control data structure (see Table 347). The source data transfer size and the destination data transfer size must be the same.
The controller always uses 32-bit data transfers when it accesses a channel control data structure.

The DMA control block contains the control logic that performs the following tasks:

- Arbitrates the incoming requests.
- Indicates which channel is active.
- Indicates when a channel is complete.
- Indicates when an error has occurred on the AHB-Lite interface.
- Enables slow peripherals to stall the completion of a DMA cycle.
- Waits for a request to clear before completing a DMA cycle.
- Performs multiple or single DMA transfers for each request.
- Performs the following types of DMA transfers:
  - memory-to-memory
  - memory-to-peripheral
  - peripheral-to-memory

Remark: Peripheral-to-peripheral transactions are not supported because each channel only provides a single DMA request interface.

21.4.1 Memory regions accessible by the micro DMA controller

The DMA channel control data structure is written to and updated in SRAM.
Memory-to-memory DMA transfers are supported as software-controlled transfers.

21.4.2 DMA system connections

The type of connection between the micro DMA and the supported peripheral devices depends on the DMA functions implemented in those peripherals. SSP uses single transfer and transfer requests, UART and ADC use transfer requests allowing one or more transfers. See Table 343 for more details on transfer types. Table 321 shows the DMA channel numbers used by the supported peripherals.

Table 321. DMA connections

Peripheral	DMA channel	single DMA transfer request	DMA transfer request
UART0 Tx	0	-	yes
UART0 Rx	1	-	yes
UART1 Tx	2	-	yes
UART1 Rx	3	-	yes
SSP Tx	4	yes	yes
SSP Rx	5	yes	yes
ADC	6	-	yes
RTC	7	-	yes
32-bit Timer 0 match 0	8	-	yes
32-bit Timer 0 match 1	9	-	yes
32-bit Timer 1 match 0	10	-	yes
32-bit Timer 1 match 1	11	-	yes
16-bit Timer 0 match 0	12	-	yes
16-bit Timer 1 match 0	13	-	yes
Comparator 0	14	-	yes
Comparator 1	15	-	yes
PIO 0	16	-	yes
PIO 1	17	-	yes
PIO 2	18	-	yes
reserved	19	-	yes
reserved	20	-	yes
memory-to-memory	21	-	software only

[1] Any DMA channel not used for peripheral DMA may be used to do a software triggered memory-to-memory transfer.

The DMA interrupt is connected to the NVIC (see Table 4). The interrupt signals are generated by the DMA done signals for each channel and the DMA error condition, if enabled. See also Table 343.

21.5 Clocking and power control

The clock to the micro DMA controller is provided by the system clock, which is controlled by the SYSAHBCLKDIV register (Table 20). The micro DMA controller can be disabled through the System AHB clock control register bit 12 (Table 21) for power savings.

21.6 Register description

Table 322 shows the micro DMA register map.

Table 322. Register overview: micro DMA (base address 0x4004 C000)

Name	Access	Address offset	Description	Reset value
DMA_STATUS	RO	0x000	DMA status register	-
DMA_CFG	WO	0x004	DMA configuration register	-
CTRL_BASE_PTR	R/W	0x008	Channel control base pointer register	0x0000 0000
ATL_CTRL_BASE_PTR	RO	0x00C	Channel alternate control base pointer register	
DMA_WAITONREQ_STATUS	RO	0x010	Channel wait on request status register	<tbd>
CHNL_SW_REQUEST	WO	0x014	Channel software request register	-
CHNL_USEBURST_SET	R/W	0x018	Channel useburst set register	0x0000 0000
CHNL_USEBURST_CLR	WO	0x01C	Channel useburst clear register	-
CHNL_REQ_MASK_SET	R/W	0x020	Channel request mask set register	0x0000 0000
CHNL_REQ_MASK_CLR	WO	0x024	Channel request mask clear register	-
CHNL_ENABLE_SET	R/W	0x028	Channel enable set register	0x0000 0000
CHNL_ENABLE_CLR	WO	0x02C	Channel enable clear register	-
CHNL_PRI_ALT_SET	R/W	0x030	Channel primary-alternate set register	0x0000 0000
CHNL_PRI_ALT_CLR	WO	0x034	Channel primary-alternate clear register	-
CHNL_PRIORITY_SET	R/W	0x038	Channel priority set register	0x0000 0000
CHNL_PRIORITY_CLR	WO	0x03C	Channel priority clear register	-
-	-	0x040 - 0x048	Reserved	-
ERR_CLR	R/W	0x04C	Bus error clear register	0x0000 0000
-	-	0x050 - 0x07C	Reserved	-
CHNL_IRQ_STATUS	R/W	0x080	Channel DMA interrupt status register	0x0000 0000
IRQ_ERR_ENABLE	R/W	0x084	DMA error interrupt enable register	0x0000 0000
CHNL_IRQ_ENABLE	R/W	0x088	Channel DMA interrupt enable register	0x0000 0000

21.6.1 DMA status register

This register is a read-only register and returns the status of the micro DMA controller. This register cannot be read when the micro DMA controller is in the reset state.

Table 323. DMA status register (DMA_STATUS, address 0x4004 C000) bit description

Bit	Symbol	Description	Reset value
0	MASTER_EN	Enable status of the controller: 0 = controller disabled. 1 = controller enabled.	
3:1	-	Reserved.	-
7:4	STATE	Current state of the control state machine. State can be one of the following: 0000 = idle 0001 = reading channel controller data 0010 = reading source data end pointer 0011 = reading destination data end pointer 0100 = reading source data 0101 = writing destination data 0110 = waiting for DMA request to clear 0111 = writing channel controller data 1000 = stalled 1001 = done 1010 = peripheral scatter-gather transition 1011-1111 = undefined	
15:8	-	Reserved.	-
20:16	CHNLS	The LPC122x micro DMA controller is configured for 22 channels. Reset value is number of channels – 1.	10100
31:21	-	Reserved.	-

21.6.2 DMA configuration register

This register is a write-only register and configures the micro DMA controller.

The CHNL_PROT_CTRLn bits (bits 6:5) configure the protection control for AHB access to the channel control data structure and should be set to zero for normal operation of the DMA controller.

Table 324. DMA configuration register (DMA_CFG, address 0x4004 C004) bit description

Bit	Symbol	Value	Description	Reset value
0	MASTER_EN		Enable for the controller:	-
		0	Disables the controller.	
		1	Enables the controller.	
4:1	-		Reserved. Write as zero.	-
5	CHNL_PROT_CTRL1		Channel protection access control:	-
		0	User access	
		1	Privileged access	
6	CHNL_PROT_CTRL2		Channel protection buffer control:	-
		0	Non-bufferable	
		1	Bufferable	
31:7	-		Reserved. Write as zero.	-

21.6.3 Channel control base pointer register

This register is a read/write register and configures the base pointer. The base pointer must point to a location in the LPC122x's SRAM because the micro DMA controller provides no internal memory for storing the channel control data structure.

The register cannot be read when the micro DMA controller is in the reset state.

Table 325. Channel control base pointer register (CTRL_BASE_PTR, address 0x4004 C008) bit description

Bit	Symbol	Description	Reset value
7:0	-	Reserved. Write as zero.	0x0
31:8	CTRL_BASE_PTR	Pointer to the base address of the primary data structure.	0x0

21.6.4 Channel alternate control base pointer register

This register is a read-only register and returns the base address of the alternate data structure.

This register removes the necessity for application software to calculate the base address of the alternate data structure (see Section 21.7.5).

The register cannot be read when the micro DMA controller is in the reset state.

Table 326. Channel alternate control base pointer register (ATL_CTRL_BASE_PTR, address 0x4004 C00C) bit description

Bit	Symbol	Description	Reset value
31:0	ALT_CTRL_BASE_PTR	Base address of the alternate data structure	0x0

21.6.5 Channel wait on request status register

This register is a read-only register and returns the status of the dma_waitonreq[c] signal for a channel c (c = 0 to 20).

The register cannot be read when the micro DMA controller is in the reset state.

Table 327. Channel wait on request status register (DMA_WAITONREQ_STATUS, address 0x4004 C010) bit description

Bit	Symbol	Description	Reset value
21:0	DMA_WAITONREQ_STATUS	Channel c wait-on-request status (c = 0 to 20): Bit c = 0: dma_waitonreq[c] is LOW. Bit c = 1: dma_waitonreq[c] is HIGH.	-
31:22	-	Reserved.	-

21.6.6 Channel software request register

This is a write-only register and enables the generation of a software DMA request for a channel c (c = 0 to 20).

Writing to a bit where a DMA channel is not implemented does not create a DMA request for that channel.

Table 328. Channel software request register (CHNL_SW_REQUEST, address 0x4004 C014) bit description

Bit	Symbol	Description	Reset value
21:0	DMA_SW_REQUEST	Set the appropriate bit to generate a software DMA request on the corresponding DMA channel. Write as: Bit [c] = 0: Does not create a DMA request for channel c. Bit [c] = 1: Creates a DMA request for channel c.	-
31:22	-	Reserved.	-

21.6.7 Channel useburst set register

This register is a read/write register and disables the single DMA request (dma_sreq[c]) input for a channel c (c = 0 to 20) from generating requests. Therefore, only the dma_req[c] signal generates requests.

Remark: The useburst status register applies to channels 4 and 5 (SSP) only.

Reading this register returns the useburst status. Writing to a bit where a DMA channel is not implemented has no effect.

Table 329. Channel useburst set register (CHNL_USEBURST_SET, address 0x4004 C018) bit description

Bit	Symbol	Description	Reset value
21:0	DMA_USEBURST_SET	Returns the useburst status for channel c (c = 0 to 20) or disables dma_sreq[c] from generating DMA requests. (For channels 4 and 5 only.) Read as: Bit [c] = 0: DMA channel c responds to requests that it receives on dma_req[c] or dma_sreq[c]. The controller performs 2^R, or single, bus transfers. Bit [c] = 1: DMA channel c does not respond to requests that it receives on dma_sreq[c]. The controller only responds to dma_req[c] requests and performs 2^R transfers. Write as: Bit [c] = 0: No effect. Use the CHNL_USEBURST_CLR register to set bit [c] to 0. Bit [c] = 1: Disables dma_sreq[C] from generating DMA requests. The controller performs 2^R transfers.	0x0
31:22	-	Reserved.	-

21.6.8 Channel useburst clear register

This register is a write-only register and enables the DMA single request for a channel c (c = 0 to 7, dma_sreq[c]) to generate requests. Writing to a bit where a DMA channel is not implemented has no effect.

Remark: The useburst clear register applies to channels 4 and 5 (SSP) only.

Table 330. Channel useburst clear register (CHNL_USEBURST_CLR, address 0x4004 C01C) bit description

Bit	Symbol	Description	Reset value
21:0	CHNL_USEBURST_CLR	Set the appropriate bit to enable dma_sreq[c] to generate requests. (For channels 4 and 5 only.) Write as: Bit [c] = 0: No effect. Use the chnl_useburst_set Register to disable dma_sreq[c] from generating requests. Bit [c] = 1: Enables dma_sreq[c] to generate DMA requests.	-
31:22	-	Reserved.	-

21.6.9 Channel request mask set register

This register is a read/write register and disables a HIGH on the DMA request signal for a channel c (c = 0 to 20) (dma_req[c] signal), or the single DMA request signal (dma_sreqc[c]), from generating a request. Reading the register returns the request mask status for dma_req[c] and dma_sreq[c]. Writing to a bit where a DMA channel is not implemented has no effect.

Table 331. Channel request mask set register (CHNL_REQ_MASK_SET, address 0x4004 C020) bit description

Bit	Symbol	Description	Reset value
21:0	CHNL_REQ_MASK_SET	Returns the request mask status of dma_req[c] and dma_sreq[c], or disables the corresponding channel from generating DMA requests. Read as: Bit [c] = 0: External requests are enabled for channel c. Bit [c] = 1: External requests are disabled for channel c. Write as: Bit [c] = 0: No effect. Use the CHNL_REQ_MASK_CLR Register to enable DMA requests. Bit [c] = 1: Disables dma_req[c] and dma_sreq[c] from generating DMA requests.	0x0
31:22	-	Reserved.	-

21.6.10 Channel request mask clear register

This register is a write-only register and for a channel c (c = 0 to 20) enables a HIGH on dma_req[c], or dma_sreq[c], to generate a request. Writing to a bit where a DMA channel is not implemented has no effect.

Table 332. Channel request mask clear register (CHNL_REQ_MASK_CLR, address 0x4004 C024) bit description

Bit	Symbol	Description	Reset value
21:0	CHNL_REQ_MASK_CLR	Set the appropriate bit to enable DMA requests for the channel corresponding to dma_req[c] and dma_sreq[c]. Write as: Bit [c] = 0: No effect. Use the chnl_req_mask_set Register to disable dma_req[c] and dma_sreq[c] from generating requests. Bit [c] = 1: Enables dma_req[c] or dma_sreq[c] to generate DMA requests.	-
31:22	-	Reserved.	-

21.6.11 Channel enable set register

This register is a read/write register and enables a DMA channel c (c = 0 to 20). Reading the register returns the enable status of the channels. Writing to a bit where a DMA channel is not implemented has no effect.

Table 333. Channel enable set register (CHNL_ENABLE_SET, address 0x4004 C028) bit description

Bit	Symbol	Description	Reset value
21:0	CHNL_ENABLE_SET	Returns the enable status of the channels, or enables the corresponding channels. Read as: Bit [c] = 0: Channel c is disabled. Bit [c] = 1 Channel c is enabled. Write as: Bit [c] = 0: No effect. Use the CHNL_ENABLE_CLR Register to disable a channel. Bit [c] = 1: Enables channel c.	0x0
31:22	-	Reserved.	-

21.6.12 Channel enable clear register

This register is a write-only register and disables a DMA channel. Writing to a bit where a DMA channel is not implemented has no effect.

Remark: The controller disables a channel by setting the appropriate bit when either:

- The controller completes the DMA cycle.
- The controller reads a channel_cfg memory location which has cycle_ctrl = 000.
- An error occurs on the AHB-Lite bus.

Table 334. Channel enable clear register (CHNL_ENABLE_CLR, address 0x4004 C02C) bit description

Bit	Symbol	Description	Reset value
21:0	CHNL_ENABLE_CLR	Set the appropriate bit to disable the corresponding DMA channel. Write as: Bit [c] = 0: No effect. Use the CHNL_ENABLE_SET Register to enable DMA channels. Bit [c] = 1 Disables channel c.	-
31:22	-	Reserved.	-

21.6.13 Channel primary-alternate set register

This register is a read/write register and configures a DMA channel c (c = 0 to 20) to use the alternate data structure. Reading the register returns the status of which data structure is in use for the corresponding DMA channel. Writing to a bit where a DMA channel is not implemented has no effect.

Remark: The controller toggles the value of the CHNL_PRI_ALT_SET[c] bit after it completes one of the following:

- the four transfers that the primary data structure specifies for a memory scatter-gather or peripheral scatter-gather DMA cycle. For details, see the *ARM micro DMA documentation*.
- all the transfers that the primary data structure specifies for a ping-pong DMA cycle.
- all the transfers that the alternate data structure specifies for the following DMA cycle types:
  - ping-pong.
  - memory scatter-gather. For details, see the *ARM micro DMA (PL230) documentation*.
  - peripheral scatter-gather. For details, see the *ARM micro DMA (PL230) documentation*.

Table 335. Channel primary-alternate set register (CHNL_PRI_ALT_SET, address 0x4004 C030) bit description

Bit	Symbol	Description	Reset value
21:0	CHNL_PRI_ALT_SET	Returns the channel control data structure status, or selects the alternate data structure for the corresponding DMA channel c. Read as: Bit [c] = 0: DMA channel c is using the primary data structure. Bit [c] = 1: DMA channel c is using the alternate data structure. Write as: Bit [c] = 0: No effect. Use the CHNL_PRI_ALT_CLR Register to set bit [c] to 0. Bit [c] = 1: Selects the alternate data structure for channel c.	0x0
31:22	-	Reserved.	-

21.6.14 Channel primary-alternate clear register

This register is a write-only register and configures a DMA channel to use the primary data structure. Writing to a bit where a DMA channel is not implemented has no effect.

Remark: The controller toggles the value of the CHNL_PRI_ALT_CLR[c] bit after it completes one of the following:

- the four transfers that the primary data structure specifies for a memory scatter-gather or peripheral scatter-gather DMA cycle.
- all the transfers that the primary data structure specifies for a ping-pong DMA cycle.
- all the transfers that the alternate data structure specifies for the following DMA cycle types:
  - ping-pong.
  - memory scatter-gather. For details, see the *ARM micro DMA (PL230) documentation*.
  - peripheral scatter-gather. For details, see the *ARM micro DMA (PL230) documentation*.

Table 336. Channel primary-alternate clear register (CHNL_PRI_ALT_CLR, address 0x4004 C034) bit description

Bit	Symbol	Description	Reset value
21:0	CHNL_PRI_ALT_CLR	Set the appropriate bit to select the primary data structure for the corresponding DMA channel c. Write as: Bit [c] = 0: No effect. Use the CHNL_PRI_ALT_SET Register to select the alternate data structure. Bit [c] = 1: Selects the primary data structure for channel c.	-
31:22	-	Reserved.	-

21.6.15 Channel priority set register

This register is read/write register and configures a DMA c (c = 0 to 20) channel to use the high priority level. Reading the register returns the status of the channel priority mask. Writing to a bit where a DMA channel is not implemented has no effect.

Table 337. Channel priority set register (CHNL_PRIORITY_SET, address 0x4004 C038) bit description

Bit	Symbol	Description	Reset value
21:0	CHNL_PRIORITY_SET	Returns the channel priority mask status, or sets the channel priority to high. Read as: Bit [c] = 0: DMA channel c is using the default priority level. Bit [c] = 1: DMA channel c is using a high priority level. Write as: Bit [c] = 0: No effect. Use the CHNL_PRIORITY_CLR Register to set channel c to the default priority level. Bit [c] = 1: Channel c uses the high priority level.	0x0
31:22	-	Reserved.	-

21.6.16 Channel priority clear register

This register is a write-only register and configures a DMA channel c (c = 0 to 20) to use the default priority level. Writing to a bit where a DMA channel is not implemented has no effect.

Table 338. Channel priority clear register (CHNL_PRIORITY_CLR, address 0x4004 C03C) bit description

Bit	Symbol	Description	Reset value
21:0	CHNL_PRIORITY_CLR	Set the appropriate bit to select the default priority level for the specified DMA channel. Write as: Bit [c] = 0: No effect. Use the CHNL_PRIORITY_SET Register to set channel c to the high priority level. Bit [c] = 1: Channel c uses the default priority level.	-
31:22	-	Reserved.	-

21.6.17 Bus error clear register

This register is a read/write register and returns the status of dma_err or sets the dma_err signal LOW.

Remark: If dma_err is deasserted at the same time as an error occurs on the AHB-Lite bus, then the error condition takes precedence and dma_err remains asserted.

Table 339. Bus error clear register (ERR_CLR, address 0x4004 C04C) bit description

Bit	Symbol	Description	Reset value
0	ERR_CLR	Returns the status of dma_error or sets the signal LOW. Read as: 0 = dma_err is LOW. 1 = dma_err is HIGH. Write as: 0 = No effect, status of dma_err is unchanged. 1 = Sets dma_err LOW.	-
31:1	-	Reserved. Write as 0.	0x0

21.6.18 Channel DMA interrupt status register

This register is a read/write register and shows the DMA done interrupt status for each DMA channel c (c = 0 to 20). Writing a one clears the status bit. Writing to a bit where a DMA channel is not implemented has no effect.

Table 340. Channel DMA interrupt status register (CHNL_IRQ_STATUS, address 0x4004 C080) bit description

Bit	Symbol	Description	Reset value
21:0	CHNL_IRQ_STAT	Returns the status of the DMA done interrupt for each channel. Read as: Bit [c] = 0: DMA done interrupt not asserted. Bit [c] = 1: DMA transfer complete for Channel c. Write as: Bit [c] = 0: No effect. Bit [c] = 1: Clears the DMA done status for Channel c.	0x0
31:22	-	Reserved.	-

21.6.19 DMA error interrupt enable register

This register is a read/write register and enables the DMA error interrupt. Writing to a bit where a DMA channel is not implemented has no effect.

Table 341. DMA error interrupt enable register (IRQ_ERR_ENABLE, address 0x4004 C084) bit description

Bit	Symbol	Description	Reset value
0	IRQ_ERR_ENABLE	Enables the DMA error (dma_error) signal to create an interrupt. Write as: Bit 0 = 0: DMA error interrupt disabled. Bit 0 = 1: DMA error interrupt enabled.	0x0
31:1	-	Reserved.	-

21.6.20 Channel DMA interrupt enable register

This register is a read/write register and enables the completion of a DMA transfer to create an interrupt for DMA channel c (c = 0 to 20). Writing to a bit where a DMA channel is not implemented has no effect.

Table 342. Channel DMA interrupt enable register (CHNL_IRQ_ENABLE, address 0x4004 C088) bit description

Bit	Symbol	Description	Reset value
21:0	CHNL_IRQ_ENABLE	Enables the DMA done (dma_done[c]) signal to create an interrupt. Write as: Bit[c] = 0: DMA done interrupt disabled for channel c. Bit[c] = 1: DMA done interrupt enabled for channel c.	0x0
31:22	-	Reserved.	-

21.7 Functional description

21.7.1 DMA control signals

The DMA control signals for DMA transfers and for providing the handshake for peripheral-to-memory transfers are listed in Table 343.

Table 343. DMA control signals

Signal	Name	Source/ destination	Description
DMA channel request	dma_req[c]	Peripheral/ controller	The peripheral asserts dma_req[c] when it has one or more data transfers that require servicing. The controller services the request by performing the DMA cycle using 2^R DMA transfers with possible arbitration between transfers depending on the setting of R_power (see Table 347). The dma_req[c] signal stays HIGH until the transfer for channel c is complete. Then the request is deasserted. All peripherals wait for a transfer request to clear before starting the next transfer.
DMA single channel request	dma_sreq[c]	Peripheral/ controller	The peripheral asserts dma_sreq when it has one data transfer that requires servicing. The controller services the request by performing the DMA cycle using one single DMA transfer. The dma_sreq[c] signal stays HIGH until the transfer for channel c is complete. Then the request is deasserted. All peripherals wait for a transfer request to clear before starting the next transfer.
DMA channel request control	dma_waitonreq[c]	Peripheral/ controller	When a group of 2^R DMA transfers complete, this signal prevents the controller from deasserting dma_active, until the corresponding dma_req, or dma_sreq, is negated.
DMA stall	dma_stall[c]	Peripheral/ controller	The peripheral can use dma_stall to extend the current DMA cycle. Peripherals that are slow to deassert a request, after a DMA transfer commences, might inadvertently trigger an additional request. Under these circumstances, the peripheral must assert dma_stall until the request can be negated.

Table 343. DMA control signals

Signal	Name	Source/ destination	Description
DMA active	dma_active[c]	Controller/ peripheral	The controller asserts the dma_active signal of a currently active channel that is transferring data. Only one dma_active signal can be active at any one time.
DMA done	dma_done[c]	Controller/ NVIC	When the controller completes a DMA cycle, it asserts dma_done for a duration of one system clock cycle. For enabled channels, only one dma_done signal can be asserted at any one time. The DMA done signal sets the corresponding bit in the CHNL_IRQ_STATUS register (Table 340) and, if enabled in the CHNL_IRQ_ENABLE register (Table 342), will generate an interrupt to the NVIC.
Bus error	dma_err	Controller/ NVIC	This is the DMA interrupt signal. The controller asserts dma_err HIGH, if an error occurs on the AHB-Lite master interface. The signal remains HIGH until the user writes to the ERR_CLR register (Table 339). The dma_err signal can generate an interrupt to the NVIC if enabled in the IRQ_ERR_ENABLE register (Table 341).

21.7.2 DMA arbitration

The controller can be configured to perform arbitration during a DMA cycle before and after a programmable number of transfers. This reduces the latency for servicing a higher priority channel.

The controller uses four bits in the channel control data structure (see Table 347) that configure how many AHB bus transfers occur before the controller re-arbitrates. These bits are known as the R_power bits because the value R is raised to the power of two and this determines the arbitration rate. For example, if R = 4 then the arbitration rate is 2^4, that is, the controller arbitrates every 16 DMA transfers.

Remark: Do not assign a low-priority channel with a large R_power value because this prevents the controller from servicing high-priority requests until it re-arbitrates.

When $N > 2^R$ and is not an integer multiple of 2^R then the controller always performs sequences of 2^R transfers until $N < 2^R$ remain to be transferred. The controller performs the remaining N transfers at the end of the DMA cycle.

21.7.3 DMA priority

Each channel can be configured to use either the default priority level or a high priority level by setting the CHNL_PRIORITY_SET Register.

When the controller arbitrates, it determines the next channel to service by using the following information:

- Priority level (default or high) assigned to the channel
- Channel number

Channel number zero has the highest priority and as the channel number increases, the priority of a channel decreases. The controller services all enabled channels with high priority first in increasing order of their channel number and then all channels with default priority (see Table 344).

Table 344. DMA channel priority

Channel number	Priority level setting	Arbitration priority in descending order
0	High	Highest
1	High	Next highest
...	...	...
20	High	...
0	Default	...
1	Default	...
...	...	...
20	Default	Lowest

21.7.4 DMA cycle types

The cycle_ctrl bits in the channel control data structure control how the DMA controller performs a cycle (see Table 347).

The controller uses four cycle types described in this manual:

- Invalid
- Basic
- Auto-request
- Ping-pong

See *ARM micro DMA (PL230) documentation* for additional cycle types.

For all cycle types, the controller arbitrates after 2^R DMA transfers. If a low-priority channel is set to a large 2^R value then it prevents all other channels from performing a DMA transfer until the low-priority DMA transfer completes. Therefore, the user must take care when setting the R_power bit in the channel_cfg data structure, that the latency for high-priority channels is not significantly increased.

21.7.4.1 Invalid cycle

After the controller completes a DMA cycle, it sets the cycle type to invalid to prevent it from repeating the same DMA cycle.

21.7.4.2 Basic cycle

In this mode, the controller can be configured to use either the primary or the alternate channel control data structure. After the channel is enabled and the controller receives a request for this channel, the flow for the basic cycle is as follows:

1. The controller performs 2^R transfers. If the number of transfers remaining is zero the flow continues at step 3.
2. The controller arbitrates:
   - If a higher-priority channel is requesting service then the controller services that channel.
   - If the peripheral or software signals a request to the controller then it continues at step 1.

3. The controller sets dma_done[c] signal for this channel HIGH for one system clock cycle. This indicates to the host processor that the DMA cycle is complete.

21.7.4.3 Auto-request cycle

When the controller operates in this mode, it is only necessary for it to receive a single request to enable it to complete the entire DMA cycle. This enables a large data transfer to occur, without significantly increasing the latency for servicing higher priority requests or requiring multiple requests from the processor or peripheral.

The auto-request cycle is typically used for memory-to-memory requests. In this case, software generates the starting request for the 2^R transfers after setting up the DMA control data structure.

In this mode, the controller can be configured to use either the primary or the alternate channel control data structure. After the channel is enabled and the controller receives a request for this channel, the flow for the auto-request cycle is as follows:

1. The controller performs 2^R transfers. If the number of transfers remaining is zero the flow continues at step 3.
2. The controller arbitrates if there are any transfers remaining after 2^R transfers. If the current channel c has the highest priority, the cycle continues at step 1.
3. The controller sets dma_done[c] signal for this channel HIGH for one system clock cycle. This indicates to the host processor that the DMA cycle is complete.

21.7.4.4 Ping-pong cycle

In this mode, the controller performs a DMA cycle using one of the data structures and then performs a DMA cycle using the other data structure. The controller continues to switch between primary and alternate structures until it reads a data structure that is invalid, until the user reprograms the cycle_type to basic, or until the host processor disables the channel.

In ping-pong mode, the user can program or reprogram one of the two channel data structures (primary or alternate) while using the other channel data structure for the active transfer. When a transfer is done, the next transfer can be started immediately using the prepared channel data structure - provided that a higher priority channel does not require servicing. If the user does not reprogram the channel control data structure not in use for a transfer, the cycle type remains invalid (which is the value at the end of the last transfer using that structure), and the ping-pong cycle completes.

The ping-pong cycle can be used for transfers to or from peripherals or for memory-to-memory transfers.

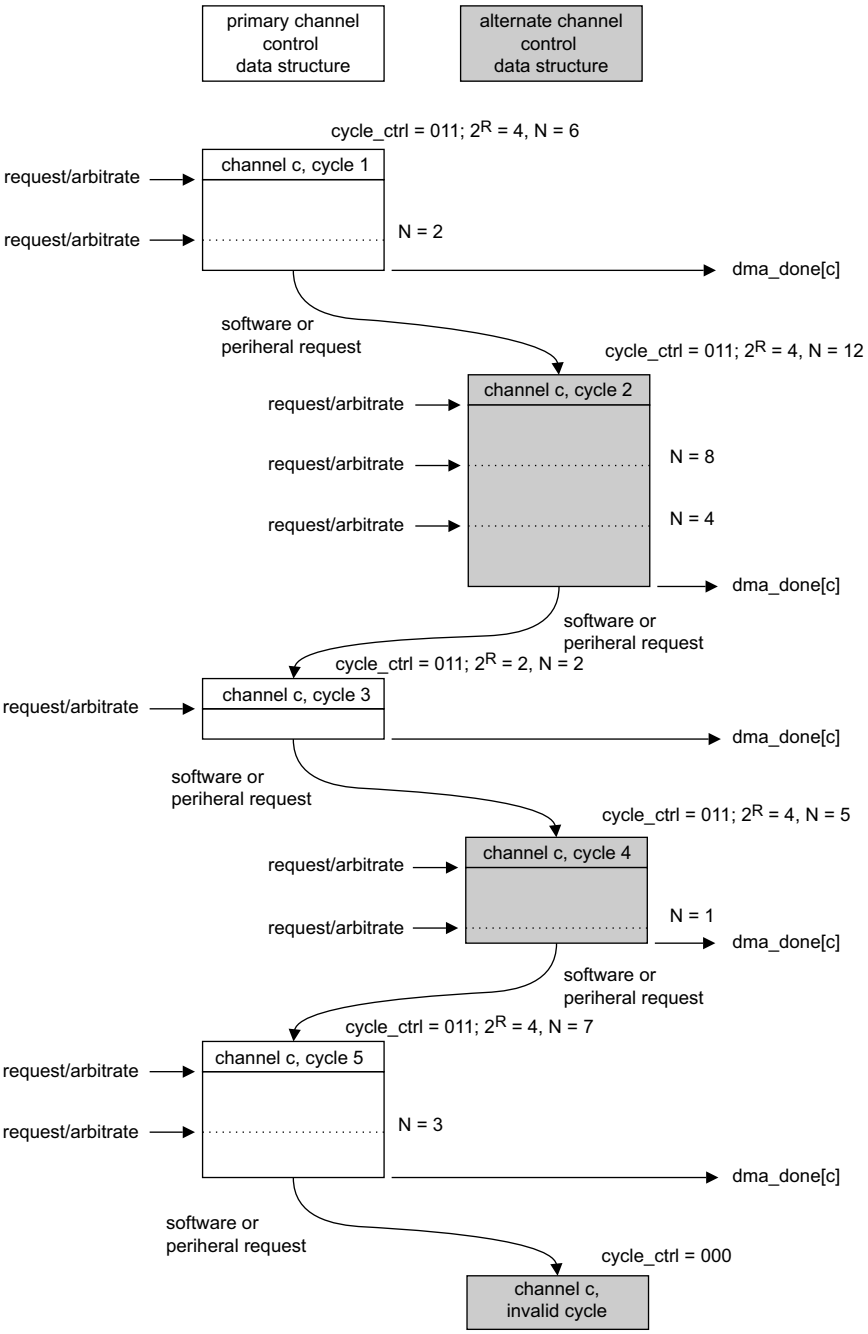


Fig 58. DMA ping-pong cycle

21.7.5 DMA control

The controller uses the SRAM to enable it to access two pointers and the control information that it requires for each channel.The channel control information is contained in the channel control data structure, and the source and destination addresses for the DMA transfer are defined by the source end and destination end pointers.

The DMA channel control data structure must be programmed in the LPC122x's SRAM memory. The data structure must be aligned on a 1024 byte boundary to create a contiguous area for 21 channel control data structures and 21 alternate channel control data structures. The base address of the primary channel control data structure in SRAM must be between 0x1000 0000 and 0x1000 1F00 in increments of 0x400 (see Figure 59).

The pointer to the base address of the channel control data structure is programmed in the CTRL_BASE_PTR register (Table 325).

Remark: The user memory (SRAM) is not accessed by the DMA controller unless the channel is enabled and a transfer is started for this channel.

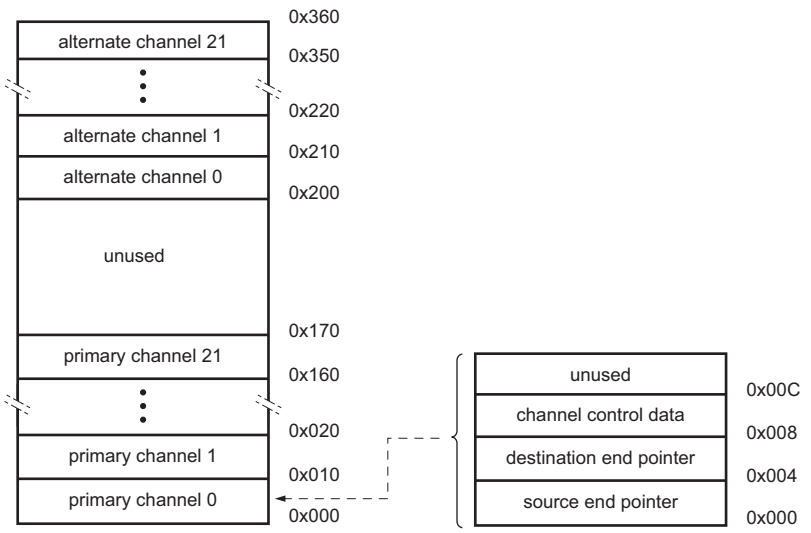


Fig 59. Memory map for the micro DMA channel control data structure (21 channels)

21.7.5.1 Source data end pointer

The src_data_end_ptr memory location contains a pointer to the end address of the source data.

Before the controller can perform a DMA transfer, this memory location must be programmed with the end address of the source data. The controller reads this memory location when it starts a 2^R DMA transfer. During a DMA transfer cycle, the controller counts down from the end address, and before each arbitration, the channel control data structure is updated with the number of remaining transfers.

Remark: The controller does not write to this memory location.

Table 345. src_data_end_ptr bit assignments

Bit	Name	Description
31:0	src_data_end_ptr	Pointer to the end address of the source data

21.7.5.2 Destination data end pointer

The dst_data_end_ptr memory location contains a pointer to the end address of the destination data.

Before the controller can perform a DMA transfer, this memory location must be programmed with the end address of the destination data. The controller reads this memory location when it starts a 2^R DMA transfer, and before each arbitration the channel control data structure is updated with the number of remaining transfers.

Remark: The controller does not write to this memory location.

Table 346. dst_data_end_ptr bit assignments

Bit	Name	Description
31:0	dst_data_end_ptr	Pointer to the end address of the destination data

21.7.5.3 Control data configuration

For each DMA transfer, the channel_cfg memory location provides the control information for the controller.

At the start of a DMA cycle, or 2^R DMA transfer, the controller fetches the channel_cfg word from SRAM memory. After the controller performs 2^R, or N, transfers, it stores the updated channel_cfg word in SRAM.

The controller does not support a dst_size value that is different from the src_size value. If the controller detects a mismatch in these values, it uses the src_size value for source and destination, and when it next updates the n_minus_1 field, it also sets the dst_size field to the same value as the src_size field.

After the controller completes the N transfers, it sets the cycle_ctrl field to 000 to indicate that the channel_cfg data is invalid. At this point, the channel configuration is overwritten in SRAM. This prevents the controller from repeating the same DMA transfer.

Remark: The controller updates the channel control data structure in SRAM after each arbitration.

Table 347. channel_cfg bit assignments

Bit	Name	Description
2:0	cycle_ctrl	The operating mode of the DMA cycle. The modes are: 000: Stop. Indicates that the data structure is invalid. 001: Basic. The controller must receive a new request, prior to it entering the arbitration process, to enable the DMA cycle to complete. 010: Auto-request. The controller automatically inserts a request for the appropriate channel during the arbitration process. This means that the initial request is sufficient to enable the DMA cycle to complete. 011: Ping-pong. The controller performs a DMA cycle using one of the data structures. After the DMA cycle completes, it performs a DMA cycle using the other data structure. After the DMA cycle completes and provided that the host processor has updated the original data structure, it performs a DMA cycle using the original data structure. The controller continues to perform DMA cycles until it either reads an invalid data structure or the host processor changes the cycle_ctrl bits to 001 or 010. 100 - 111: not used.
3	next_useburst	Controls if the chnl_useburst_set [C] bit is set to a 1, when the controller is performing a peripheral scatter-gather and is completing a DMA cycle that uses the alternate data structure. See the *ARM micro DMA (PL230) documentation* for details.
13:4	n_minus_1	Prior to the DMA cycle commencing, these bits represent the total number of DMA transfers that the DMA cycle contains. These bits must be set according to the size of DMA cycle. The 10-bit value indicates the number of DMA transfers, minus one. The possible values are: 000000000 = 1 DMA transfer 000000001 = 2 DMA transfers 000000010 = 3 DMA transfers 000000011 = 4 DMA transfers 000000100 = 5 DMA transfers ... 111111111 = 1024 DMA transfers. The controller updates this field immediately prior to it entering the arbitration process. This enables the controller to store the number of outstanding DMA transfers that are necessary to complete the DMA cycle.
17:14	R_power	Set these bits to control how many DMA transfers can occur before the controller re-arbitrates. The possible arbitration rate settings are: 0000: Arbitrates after each DMA transfer. 0001: Arbitrates after 2 DMA transfers. 0010: Arbitrates after 4 DMA transfers. 0011: Arbitrates after 8 DMA transfers. 0100: Arbitrates after 16 DMA transfers. 0101: Arbitrates after 32 DMA transfers. 0110: Arbitrates after 64 DMA transfers. 0111: Arbitrates after 128 DMA transfers. 1000: Arbitrates after 256 DMA transfers. 1001: Arbitrates after 512 DMA transfers. 1010-1111: Arbitrates after 1024 DMA transfers. This means that no arbitration occurs during the DMA transfer because the maximum transfer size is 1024.

Table 347. channel_cfg bit assignments *…continued*

Bit	Name	Description
20:18	src_prot_ctrl3:1	Set the bits to control AHB Lite access protection when the controller reads the source data. Bit [20] controls the state access as follows: 0 = access is non-cacheable. 1 = access is cacheable. Bit [19] controls the access as follows: 0 = access is non-bufferable. 1 = access is bufferable. Bit [18] controls the access as follows: 0 = access is non-privileged. 1 = access is privileged.
23:21	dst_prot_ctrl3:1	Set the bits to control AHBLite access protection when the controller writes the destination data. For normal operation of the DMA controller, set bits 21 to 23 to zero. Bit [23]: reserved. Bit [22] controls the access as follows: 0 = access is non-bufferable. 1 = access is bufferable. Bit [21] controls the access as follows: 0 = access is non-privileged. 1 = access is privileged.
25:24	src_size	Set the bits to match the size of the source data: 00 = byte 01 = half-word 10 = word 11 = reserved

Table 347. channel_cfg bit assignments *…continued*

Bit	Name	Description
27:26	src_inc	Set the bits to control the source address increment. The address increment depends on the source data width as follows: Source data width = byte 00 = byte. 01 = half-word. 10 = word. 11 = no increment. Address remains set to the value that the src_data_end_ptr memory location contains. Source data width = half-word 00 = reserved. 01 = half-word. 10 = word. 11 = no increment. Address remains set to the value that the src_data_end_ptr memory location contains. Source data width = word 00 = reserved. 01 = reserved. 10 = word. 11 = no increment. Address remains set to the value that the src_data_end_ptr memory location contains.
29:28	dst_size	Destination data size. Must be set to the same value as source data size.
31:30	dst_inc	Destination address increment. The address increment depends on the source data width as follows: Source data width = byte 00 = byte. 01 = half-word. 10 = word. 11 = no increment. Address remains set to the value that the dst_data_end_ptr memory location contains. Source data width = half-word 00 = reserved. 01 = half-word. 10 = word. 11 = no increment. Address remains set to the value that the dst_data_end_ptr memory location contains. Source data width = word 00 = reserved. 01 = reserved. 10 = word. 11 = no increment. Address remains set to the value that the dst_data_end_ptr memory location contains.

22.1 How to read this chapter

The CRC engine is available on all LPC122x parts.

22.2 Introduction

The Cyclic Redundancy Check (CRC) generator with programmable polynomial settings supports several CRC standards commonly used. To save system power and bus bandwidth, the CRC engine supports DMA transfers in addition to software PIO operations using the CPU.

22.3 Features

- Supports three common polynomials CRC-CCITT, CRC-16, and CRC-32.
  - CRC-CCITT: $x^{16} + x^{12} + x^5 + 1$
  - CRC-16: $x^{16} + x^{15} + x^2 + 1$
  - CRC-32: $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$
- Bit order reverse and 1’s complement programmable setting for input data and CRC sum.
- Programmable seed number setting.
- Supports CPU PIO or DMA back-to-back transfer.
- Accept any size of data width per write: 8, 16 or 32-bit.
  - 8-bit write: 1-cycle operation
  - 16-bit write: 2-cycle operation (8-bit x 2-cycle)
  - 32-bit write: 4-cycle operation (8-bit x 4-cycle)

22.4 Description

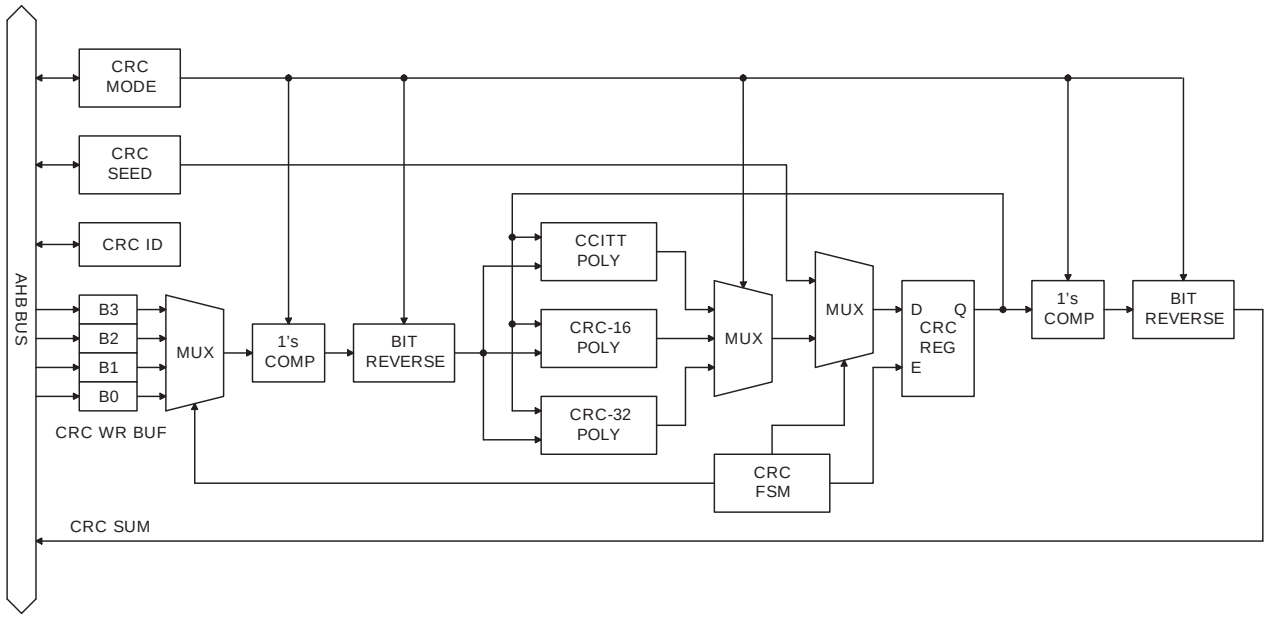


Fig 60. CRC block diagram

22.5 Register description

Table 348. Register overview: CRC engine (base address 0x5007 0000)

Name	Access	Address offset	Description	Reset value
MODE	R/W	0x00	CRC mode register	0x0000 0000
SEED	R/W	0x04	CRC seed register	0x0000 FFFF
SUM	RO	0x08	CRC checksum register	0x0000 FFFF
WR_DATA	WO	0x08	CRC data register	-

22.5.1 CRC mode register

Table 349. CRC mode register (MODE, address 0x5007 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

22.5.2 CRC seed register

Table 350. CRC seed register (SEED, address 0x5007 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

22.5.3 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 351. CRC checksum register (SUM, address 0x5007 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

22.5.4 CRC data register

This register is a Write-only register containing the data block for which the CRC sum will be calculated.

Table 352. CRC data register (WR_DATA, address 0x5007 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.	-

22.6 Functional description

The following sections describe the register settings for each supported CRC standard:

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

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

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

23.1 How to read this chapter

The integer division routines are available on all LPC122x parts.

23.2 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.

23.3 Description

The integer division routines perform arithmetic integer division operations and can be called in the application code through simple API calls.

The following function prototypes are used:

```
typedef struct { int quot; int rem; } idiv_return;

typedef struct { unsigned quot; unsigned rem; } uidiv_return;

typedef struct{
     /* Signed integer division */
     int             (*sidiv)(int numerator, int denominator);
     /* Unsigned integer division */
     unsigned(*uidiv) (unsigned numerator, unsigned denominator);
     /* Signed integer division with remainder */
     idiv_return(*sidivmod)(int numerator, int denominator);
     /* Unsigned integer division with remainder */
     uidiv_return (*uidivmod)(unsigned numerator, unsigned denominator);
} LPC_ROM_DIV_STRUCT;

/* The value at this address is the entry to ROM Division API.
Once it is dereferenced, individual API functions can be used
*/
#define LPC_122x_DIVROM_LOC (0x1FFC0000)
```

23.4 Examples

23.4.1 Initialization

The example C-code listing below shows how to initialize the API's ROM table pointer.

```
/* Declare table pointer */
LPC_ROM_DIV_STRUCT const* pDivROM;
/* Assign pointer to table */
pDivROM = *((LPC_ROM_DIV_STRUCT**)LPC_122x_DIVROM_LOC);
```

23.4.2 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 = pDivROM->sidiv(-99, 6);
/* result now contains (-16) */
```

23.4.3 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 = pDivROM-> uidivmod (+99, 4);
/* result.quot contains (+24) */
/* result.rem contains (+3) */
```

24.1 How to read this chapter

The debug functionality is identical on all LPC122x parts.

24.2 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, which can also be used as trace triggers.

24.3 General description

Debug functions are integrated into the ARM Cortex-M0. Debugging with the LPC122x uses the Serial Wire Debug mode.

24.4 Pin description

Table 353. Serial Wire Debug pin description

Pin Name	Type	Description
SWCLK	Input	**Serial Wire Clock.** This pin is the clock for debug logic when in the Serial Wire Debug mode (SWDCLK).
SWDIO	Input / Output	**Serial wire debug data input/output.** The SWDIO pin is used by an external debug tool to communicate with and control the ARM Cortex-M0 CPU.

24.5 Debug notes

24.5.1 Debug limitations

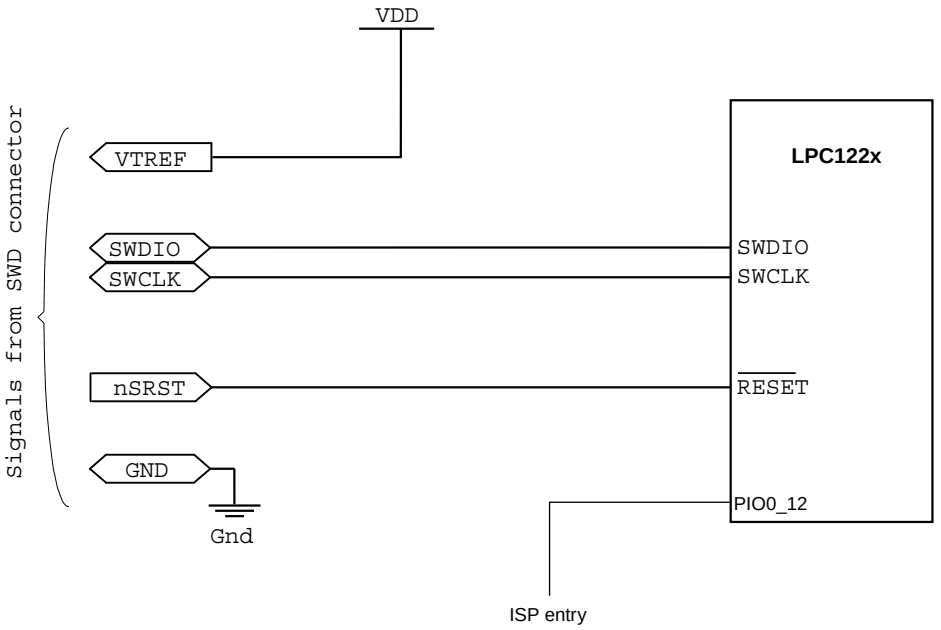
Important: The user should be aware of certain limitations during debugging. The most important is that, due to limitations of the ARM Cortex-M0 integration, the LPC122x cannot wake up in the usual manner from Deep-sleep mode. It is recommended not to use this mode during debug.

Another issue is that debug mode changes the way in which reduced power modes work internal to the ARM Cortex-M0 CPU, and this ripples through the entire system. These differences mean that power measurements should not be made while debugging, the results will be higher than during normal operation in an application.

During a debugging session, the System Tick Timer is automatically stopped whenever the CPU is stopped. Other peripherals are not affected.

24.5.2 Debug connections

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 or reconfiguration of SWD pins as ADC inputs, 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 61. Connecting the SWD pins to a standard SWD connector

25.1 Introduction

The following material is using the ARM *Cortex-M0 User Guide*. Minor changes have been made regarding the specific implementation of the Cortex-M0 for the LPC122x.

25.2 About the Cortex-M0 processor and core peripherals

The Cortex-M0 processor is an entry-level 32-bit ARM Cortex processor designed for a broad range of embedded applications. It offers significant benefits to developers, including:

- a simple architecture that is easy to learn and program
- ultra-low power, energy efficient operation
- excellent code density
- deterministic, high-performance interrupt handling
- upward compatibility with Cortex-M processor family.

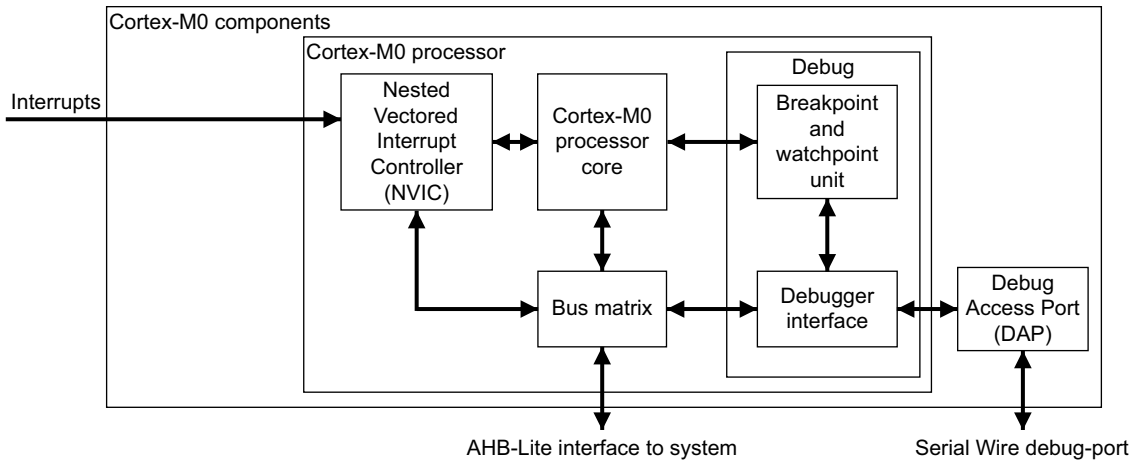


Fig 62. Cortex-M0 implementation

The Cortex-M0 processor is built on a highly area and power optimized 32-bit processor core, with a 3-stage pipeline von Neumann architecture. The processor delivers exceptional energy efficiency through a small but powerful instruction set and extensively optimized design, providing high-end processing hardware including a single-cycle multiplier.

The Cortex-M0 processor implements the ARMv6-M architecture, which is based on the 16-bit Thumb instruction set and includes Thumb-2 technology. This provides the exceptional performance expected of a modern 32-bit architecture, with a higher code density than other 8-bit and 16-bit microcontrollers.

The Cortex-M0 processor closely integrates a configurable **Nested Vectored Interrupt Controlle**r (NVIC), to deliver industry-leading interrupt performance. The NVIC:

- includes a **non-maskable interrupt** (NMI).
- provides zero jitter interrupt option.
- provides four interrupt priority levels.

The tight integration of the processor core and NVIC provides fast execution of **interrupt service routines** (ISRs), dramatically reducing the interrupt latency. This is achieved through the hardware stacking of registers, and the ability to abandon and restart load-multiple and store-multiple operations. Interrupt handlers do not require any assembler wrapper code, removing any code overhead from the ISRs. Tail-chaining optimization also significantly reduces the overhead when switching from one ISR to another.

To optimize low-power designs, the NVIC integrates with the sleep modes, that include a Deep-sleep function that enables the entire device to be rapidly powered down.

25.2.1 System-level interface

The Cortex-M0 processor provides a single system-level interface using AMBA technology to provide high speed, low latency memory accesses.

25.2.2 Integrated configurable debug

The Cortex-M0 processor implements a complete hardware debug solution, with extensive hardware breakpoint and watchpoint options. This provides high system visibility of the processor, memory and peripherals through a 2-pin **Serial Wire Debug** (SWD) port that is ideal for microcontrollers and other small package devices.

25.2.3 Cortex-M0 processor features summary

- high code density with 32-bit performance
- tools and binary upwards compatible with Cortex-M processor family
- integrated ultra low-power sleep modes
- efficient code execution permits slower processor clock or increases sleep mode time
- single-cycle 32-bit hardware multiplier
- zero jitter interrupt handling
- extensive debug capabilities.

25.2.4 Cortex-M0 core peripherals

These are:

NVIC — The NVIC is an embedded interrupt controller that supports low latency interrupt processing.

System Control Block — The **System Control Block** (SCB) is the programmers model interface to the processor. It provides system implementation information and system control, including configuration, control, and reporting of system exceptions.

System timer — The system timer, SysTick, is a 24-bit count-down timer. Use this as a Real Time Operating System (RTOS) tick timer or as a simple counter.

25.3 Processor

25.3.1 Programmers model

This section describes the Cortex-M0 programmers model. In addition to the individual core register descriptions, it contains information about the processor modes and stacks.

25.3.1.1 Processor modes

The processor **modes** are:

Thread mode — Used to execute application software. The processor enters Thread mode when it comes out of reset.

Handler mode — Used to handle exceptions. The processor returns to Thread mode when it has finished all exception processing.

25.3.1.2 Stacks

The processor uses a full descending stack. This means the stack pointer indicates the last stacked item on the stack memory. When the processor pushes a new item onto the stack, it decrements the stack pointer and then writes the item to the new memory location. The processor implements two stacks, the main stack and the process stack, with independent copies of the stack pointer, see Section 25.3.1.3.2.

In Thread mode, the CONTROL register controls whether the processor uses the main stack or the process stack, see Section 25–25.3.1.3.7. In Handler mode, the processor always uses the main stack. The options for processor operations are:

Table 354. Summary of processor mode and stack use options

Processor mode	Used to execute	Stack used
Thread	Applications	Main stack or process stack See Section 25–25.3.1.3.7
Handler	Exception handlers	Main stack

25.3.1.3 Core registers

The processor core registers are:

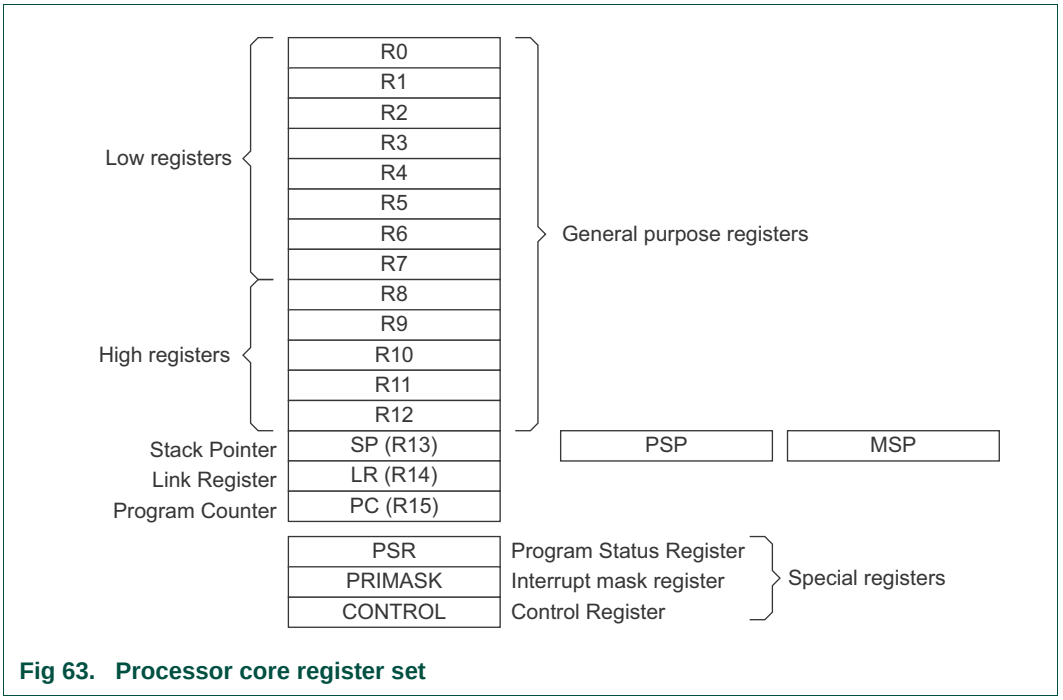


Fig 63. Processor core register set

Table 355. Core register set summary

Name	Type[1]	Reset value	Description
R0-R12	RW	Unknown	Section 25–25.3.1.3.1
MSP	RW	See description	Section 25–25.3.1.3.2
PSP	RW	Unknown	Section 25–25.3.1.3.2
LR	RW	Unknown	Section 25–25.3.1.3.3
PC	RW	See description	Section 25–25.3.1.3.4
PSR	RW	Unknown[2]	Table 25–356
APSR	RW	Unknown	Table 25–357
IPSR	RO	0x00000000	Table 358
EPSR	RO	Unknown[2]	Table 25–359
PRIMASK	RW	0x00000000	Table 25–360
CONTROL	RW	0x00000000	Table 25–361

[1] Describes access type during program execution in thread mode and Handler mode. Debug access can differ.

[2] Bit[24] is the T-bit and is loaded from bit[0] of the reset vector.

25.3.1.3.1 General-purpose registers

R0-R12 are 32-bit general-purpose registers for data operations.

25.3.1.3.2 Stack Pointer

The Stack Pointer (SP) is register R13. In Thread mode, bit[1] of the CONTROL register indicates the stack pointer to use:

- 0 = **Main Stack Pointer** (MSP). This is the reset value.
- 1 = **Process Stack Pointer** (PSP).

On reset, the processor loads the MSP with the value from address 0x00000000.

25.3.1.3.3 Link Register

The **Link Register** (LR) is register R14. It stores the return information for subroutines, function calls, and exceptions. On reset, the LR value is Unknown.

25.3.1.3.4 Program Counter

The **Program Counter** (PC) is register R15. It contains the current program address. On reset, the processor loads the PC with the value of the reset vector, which is at address 0x00000004. Bit[0] of the value is loaded into the EPSR T-bit at reset and must be 1.

25.3.1.3.5 Program Status Register

The **Program Status Register** (PSR) combines:

- **Application Program Status Register** (APSR)
- **Interrupt Program Status Register** (IPSR)
- **Execution Program Status Register** (EPSR).

These registers are mutually exclusive bitfields in the 32-bit PSR. The PSR bit assignments are:

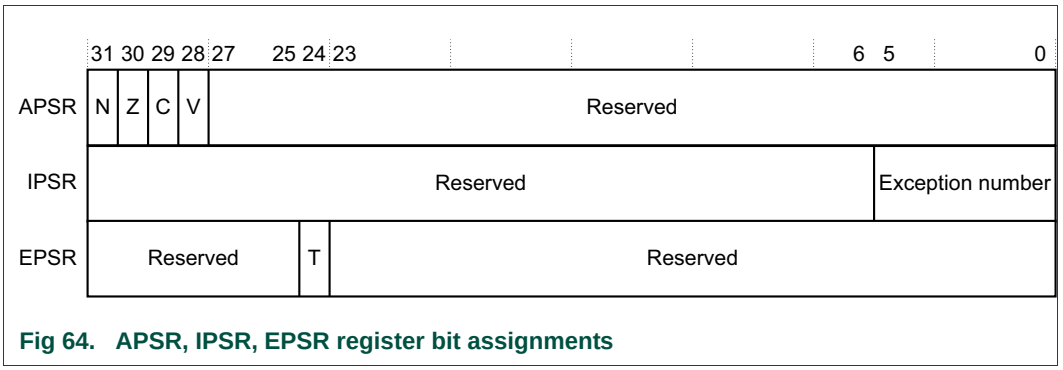


Fig 64. APSR, IPSR, EPSR register bit assignments

Access these registers individually or as a combination of any two or all three registers, using the register name as an argument to the MSR or MRS instructions. For example:

- read all of the registers using PSR with the MRS instruction
- write to the APSR using APSR with the MSR instruction.

The PSR combinations and attributes are:

Table 356. PSR register combinations

Register	Type	Combination
PSR	RW[1][2]	APSR, EPSR, and IPSR
IEPSR	RO	EPSR and IPSR
IAPSR	RW[1]	APSR and IPSR
EAPSR	RW[2]	APSR and EPSR

[1] The processor ignores writes to the IPSR bits.

[2] Reads of the EPSR bits return zero, and the processor ignores writes to the these bits

See the instruction descriptions Section 25–25.4.7.6 and Section 25–25.4.7.7 for more information about how to access the program status registers.

Application Program Status Register: The APSR contains the current state of the condition flags, from previous instruction executions. See the register summary in Table 25–355 for its attributes. The bit assignments are:

Table 357. APSR bit assignments

Bits	Name	Function
[31]	N	Negative flag
[30]	Z	Zero flag
[29]	C	Carry or borrow flag
[28]	V	Overflow flag
[27:0]	-	Reserved

See Section 25.4.4.1.4 for more information about the APSR negative, zero, carry or borrow, and overflow flags.

Interrupt Program Status Register: The IPSR contains the exception number of the current **Interrupt Service Routine** (ISR). See the register summary in Table 25–355 for its attributes. The bit assignments are:

Table 358. IPSR bit assignments

Bits	Name	Function
[31:6]	-	Reserved
[5:0]	Exception number	This is the number of the current exception: 0 = Thread mode 1 = Reserved 2 = NMI 3 = HardFault 4-10 = Reserved 11 = SVCall 12, 13 = Reserved 14 = PendSV 15 = SysTick 16 = IRQ0 . . . 47 = IRQ31 48-63 = Reserved. see Section 25–25.3.3.2 for more information.

Execution Program Status Register: The EPSR contains the Thumb state bit.

See the register summary in Table 25–355 for the EPSR attributes. The bit assignments are:

Table 359. EPSR bit assignments

Bits	Name	Function
[31:25]	-	Reserved
[24]	T	Thumb state bit
[23:0]	-	Reserved

Attempts by application software to read the EPSR directly using the `MRS` instruction always return zero. Attempts to write the EPSR using the `MSR` instruction are ignored. Fault handlers can examine the EPSR value in the stacked PSR to determine the cause of the fault. See Section 25–25.3.3.6. The following can clear the T bit to 0:

- instructions `BLX`, `BX` and `POP{PC}`
- restoration from the stacked xPSR value on an exception return
- bit[0] of the vector value on an exception entry.

Attempting to execute instructions when the T bit is 0 results in a HardFault or lockup. See Section 25–25.3.4.1 for more information.

Interruptible-restartable instructions: The interruptible-restartable instructions are `LDM` and `STM`. When an interrupt occurs during the execution of one of these instructions, the processor abandons execution of the instruction.

After servicing the interrupt, the processor restarts execution of the instruction from the beginning.

25.3.1.3.6 Exception mask register

The exception mask register disables the handling of exceptions by the processor. Disable exceptions where they might impact on timing critical tasks or code sequences requiring atomicity.

To disable or re-enable exceptions, use the `MSR` and `MRS` instructions, or the `CPS` instruction, to change the value of PRIMASK. See Section 25–25.4.7.6, Section 25–25.4.7.7, and Section 25–25.4.7.2 for more information.

Priority Mask Register: The PRIMASK register prevents activation of all exceptions with configurable priority. See the register summary in Table 25–355 for its attributes. The bit assignments are:

Table 360. PRIMASK register bit assignments

Bits	Name	Function
[31:1]	-	Reserved
[0]	PRIMASK	0 = no effect 1 = prevents the activation of all exceptions with configurable priority.

25.3.1.3.7 CONTROL register

The CONTROL register controls the stack used when the processor is in Thread mode. See the register summary in Table 25–355 for its attributes. The bit assignments are:

Table 361. CONTROL register bit assignments

Bits	Name	Function
[31:2]	-	Reserved
[1]	Active stack pointer	Defines the current stack: 0 = MSP is the current stack pointer 1 = PSP is the current stack pointer. In Handler mode this bit reads as zero and ignores writes.
[0]	-	Reserved.

Handler mode always uses the MSP, so the processor ignores explicit writes to the active stack pointer bit of the CONTROL register when in Handler mode. The exception entry and return mechanisms update the CONTROL register.

In an OS environment, it is recommended that threads running in Thread mode use the process stack and the kernel and exception handlers use the main stack.

By default, Thread mode uses the MSP. To switch the stack pointer used in Thread mode to the PSP, use the `MSR` instruction to set the Active stack pointer bit to 1, see Section 25–25.4.7.6.

Remark: When changing the stack pointer, software must use an `ISB` instruction immediately after the `MSR` instruction. This ensures that instructions after the ISB execute using the new stack pointer. See Section 25–25.4.7.5.

25.3.1.4 Exceptions and interrupts

The Cortex-M0 processor supports interrupts and system exceptions. The processor and the **Nested Vectored Interrupt Controller** (NVIC) prioritize and handle all exceptions. An interrupt or exception changes the normal flow of software control. The processor uses handler mode to handle all exceptions except for reset. See Section 25–25.3.3.6.1 and Section 25–25.3.3.6.2 for more information.

The NVIC registers control interrupt handling. See Section 25–25.5.2 for more information.

25.3.1.5 Data types

The processor:

- supports the following data types:
  - 32-bit words
  - 16-bit halfwords
  - 8-bit bytes
- manages all data memory accesses as little-endian. Instruction memory and **Private Peripheral Bus** (PPB) accesses are always little-endian. See Section 25–25.3.2.1 for more information.

25.3.1.6 The Cortex Microcontroller Software Interface Standard

ARM provides the **Cortex Microcontroller Software Interface Standard** (CMSIS) for programming Cortex-M0 microcontrollers. The CMSIS is an integrated part of the device driver library.

For a Cortex-M0 microcontroller system, CMSIS defines:

- a common way to:
  - access peripheral registers
  - define exception vectors
- the names of:
  - the registers of the core peripherals
  - the core exception vectors
- a device-independent interface for RTOS kernels.

The CMSIS includes address definitions and data structures for the core peripherals in the Cortex-M0 processor. It also includes optional interfaces for middleware components comprising a TCP/IP stack and a Flash file system.

The CMSIS simplifies software development by enabling the reuse of template code, and the combination of CMSIS-compliant software components from various middleware vendors. Software vendors can expand the CMSIS to include their peripheral definitions and access functions for those peripherals.

This document includes the register names defined by the CMSIS, and gives short descriptions of the CMSIS functions that address the processor core and the core peripherals.

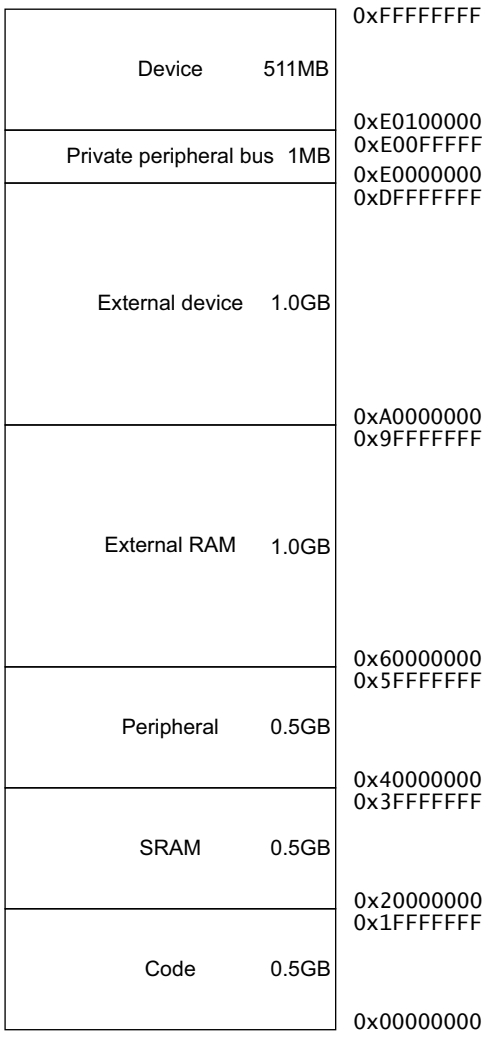
Remark: This document uses the register short names defined by the CMSIS. In a few cases these differ from the architectural short names that might be used in other documents.

The following sections give more information about the CMSIS:

- Section 25.3.5.3 “Power management programming hints”
- Section 25.4.2 “Intrinsic functions”
- Section 25.5.2.1 “Accessing the Cortex-M0 NVIC registers using CMSIS”
- Section 25.5.2.8.1 “NVIC programming hints”.

25.3.2 Memory model

This section describes the processor memory map and the behavior of memory accesses. The processor has a fixed memory map that provides up to 4GB of addressable memory. The memory map is:



For the LPC122x specific implementation of the memory map, see Figure 2.

Fig 65. Cortex-M0 memory map

The processor reserves regions of the **Private peripheral bus** (PPB) address range for core peripheral registers, see Section 25–25.2.

25.3.2.1 Memory regions, types and attributes

The memory map is split into regions. Each region has a defined memory type, and some regions have additional memory attributes. The memory type and attributes determine the behavior of accesses to the region.

The memory types are:

Normal — The processor can re-order transactions for efficiency, or perform speculative reads.

Device — The processor preserves transaction order relative to other transactions to Device or Strongly-ordered memory.

Strongly-ordered — The processor preserves transaction order relative to all other transactions.

The different ordering requirements for Device and Strongly-ordered memory mean that the memory system can buffer a write to Device memory, but must not buffer a write to Strongly-ordered memory.

The additional memory attributes include.

Execute Never (XN) — Means the processor prevents instruction accesses. A HardFault exception is generated on executing an instruction fetched from an XN region of memory.

25.3.2.2 Memory system ordering of memory accesses

For most memory accesses caused by explicit memory access instructions, the memory system does not guarantee that the order in which the accesses complete matches the program order of the instructions, providing any re-ordering does not affect the behavior of the instruction sequence. Normally, if correct program execution depends on two memory accesses completing in program order, software must insert a memory barrier instruction between the memory access instructions, see Section 25–25.3.2.4.

However, the memory system does guarantee some ordering of accesses to Device and Strongly-ordered memory. For two memory access instructions A1 and A2, if A1 occurs before A2 in program order, the ordering of the memory accesses caused by two instructions is:

A1 \ A2	Normal access	Device access		Strongly-ordered access
		Non-shareable	Shareable	
Normal access	-	-	-	-
Device access, non-shareable	-	<	-	<
Device access, shareable	-	-	<	<
Strongly-ordered access	-	<	<	<

Fig 66. Memory ordering restrictions

Where:

- — Means that the memory system does not guarantee the ordering of the accesses.

< — Means that accesses are observed in program order, that is, A1 is always observed before A2.

25.3.2.3 Behavior of memory accesses

The behavior of accesses to each region in the memory map is:

Table 362. Memory access behavior

Address range	Memory region	Memory type[1]	XN[1]	Description
0x00000000-0x1FFFFFFF	Code	Normal	-	Executable region for program code. You can also put data here.
0x20000000-0x3FFFFFFF	SRAM	Normal	-	Executable region for data. You can also put code here.
0x40000000-0x5FFFFFFF	Peripheral	Device	XN	External device memory.
0x60000000-0x9FFFFFFF	External RAM	Normal	-	Executable region for data.
0xA0000000-0xDFFFFFFF	External device	Device	XN	External device memory.
0xE0000000-0xE00FFFFF	Private Peripheral Bus	Strongly-ordered	XN	This region includes the NVIC, System timer, and System Control Block. Only word accesses can be used in this region.
0xE0100000-0xFFFFFFFF	Device	Device	XN	Vendor specific.

[1] See Section 25–25.3.2.1 for more information.

The Code, SRAM, and external RAM regions can hold programs.

25.3.2.4 Software ordering of memory accesses

The order of instructions in the program flow does not always guarantee the order of the corresponding memory transactions. This is because:

- the processor can reorder some memory accesses to improve efficiency, providing this does not affect the behavior of the instruction sequence
- memory or devices in the memory map might have different wait states
- some memory accesses are buffered or speculative.

Section 25–25.3.2.2 describes the cases where the memory system guarantees the order of memory accesses. Otherwise, if the order of memory accesses is critical, software must include memory barrier instructions to force that ordering. The processor provides the following memory barrier instructions:

DMB — The Data Memory Barrier (DMB) instruction ensures that outstanding memory transactions complete before subsequent memory transactions. See Section 25–25.4.7.3.

DSB — The Data Synchronization Barrier (DSB) instruction ensures that outstanding memory transactions complete before subsequent instructions execute. See Section 25–25.4.7.4.

ISB — The Instruction Synchronization Barrier (ISB) ensures that the effect of all completed memory transactions is recognizable by subsequent instructions. See Section 25–25.4.7.5.

The following are examples of using memory barrier instructions:

Vector table — If the program changes an entry in the vector table, and then enables the corresponding exception, use a `DMB` instruction between the operations. This ensures that if the exception is taken immediately after being enabled the processor uses the new exception vector.

Self-modifying code — If a program contains self-modifying code, use an `ISB` instruction immediately after the code modification in the program. This ensures subsequent instruction execution uses the updated program.

Memory map switching — If the system contains a memory map switching mechanism, use a `DSB` instruction after switching the memory map. This ensures subsequent instruction execution uses the updated memory map.

Memory accesses to Strongly-ordered memory, such as the System Control Block, do not require the use of `DMB` instructions.

The processor preserves transaction order relative to all other transactions.

25.3.2.5 Memory endianness

The processor views memory as a linear collection of bytes numbered in ascending order from zero. For example, bytes 0-3 hold the first stored word, and bytes 4-7 hold the second stored word. Section 25–25.3.2.5.1 describes how words of data are stored in memory.

25.3.2.5.1 Little-endian format

In little-endian format, the processor stores the **least significant byte** (lsbyte) of a word at the lowest-numbered byte, and the **most significant byte** (msbyte) at the highest-numbered byte. For example:

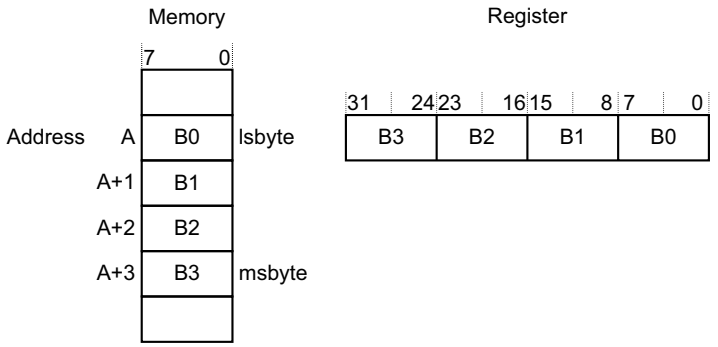


Fig 67. Little-endian format

25.3.3 Exception model

This section describes the exception model.

25.3.3.1 Exception states

Each exception is in one of the following states:

Inactive — The exception is not active and not pending.

Pending — The exception is waiting to be serviced by the processor.

An interrupt request from a peripheral or from software can change the state of the corresponding interrupt to pending.

Active — An exception that is being serviced by the processor but has not completed.

An exception handler can interrupt the execution of another exception handler. In this case both exceptions are in the active state.

Active and pending — The exception is being serviced by the processor and there is a pending exception from the same source.

25.3.3.2 Exception types

The exception types are:

Reset — Reset is invoked on power up or a warm reset. The exception model treats reset as a special form of exception. When reset is asserted, the operation of the processor stops, potentially at any point in an instruction. When reset is deasserted, execution restarts from the address provided by the reset entry in the vector table. Execution restarts in Thread mode.

NMI — A **NonMaskable Interrupt** (NMI) can be signalled by a peripheral or triggered by software. This is the highest priority exception other than reset. It is permanently enabled and has a fixed priority of −2. NMIs cannot be:

- masked or prevented from activation by any other exception
- preempted by any exception other than Reset.

HardFault — A HardFault is an exception that occurs because of an error during normal or exception processing. HardFaults have a fixed priority of -1, meaning they have higher priority than any exception with configurable priority.

SVCall — A **supervisor call** (SVC) is an exception that is triggered by the `SVC` instruction. In an OS environment, applications can use `SVC` instructions to access OS kernel functions and device drivers.

PendSV — PendSV is an interrupt-driven request for system-level service. In an OS environment, use PendSV for context switching when no other exception is active.

SysTick — A SysTick exception is an exception the system timer generates when it reaches zero. Software can also generate a SysTick exception. In an OS environment, the processor can use this exception as system tick.

Interrupt (IRQ) — An interrupt, or IRQ, is an exception signalled by a peripheral, or generated by a software request. All interrupts are asynchronous to instruction execution. In the system, peripherals use interrupts to communicate with the processor.

Table 363. Properties of different exception types

Exception number[1]	IRQ number[1]	Exception type	Priority	Vector address[2]
1	-	Reset	-3, the highest	`0x00000004`
2	-14	NMI	-2	`0x00000008`
3	-13	HardFault	-1	`0x0000000C`
4-10	-	Reserved	-	-
11	-5	SVCall	Configurable[3]	`0x0000002C`
12-13	-	Reserved	-	-

Table 363. Properties of different exception types

Exception number[1]	IRQ number[1]	Exception type	Priority	Vector address[2]
14	-2	PendSV	Configurable[3]	0x00000038
15	-1	SysTick	Configurable[3]	0x0000003C
16 and above	0 and above	Interrupt (IRQ)	Configurable[3]	0x00000040 and above[4]

[1] To simplify the software layer, the CMSIS only uses IRQ numbers and therefore uses negative values for exceptions other than interrupts. The IPSR returns the Exception number, see Table 25–358.

[2] See Section 25.3.3.4 for more information.

[3] See Section 25–25.5.2.6.

[4] Increasing in steps of 4.

For an asynchronous exception, other than reset, the processor can execute additional instructions between when the exception is triggered and when the processor enters the exception handler.

Privileged software can disable the exceptions that Table 25–363 shows as having configurable priority, see Section 25–25.5.2.3.

For more information about HardFaults, see Section 25–25.3.4.

25.3.3.3 Exception handlers

The processor handles exceptions using:

Interrupt Service Routines (ISRs) — Interrupts IRQ0 to IRQ31 are the exceptions handled by ISRs.

Fault handler — HardFault is the only exception handled by the fault handler.

System handlers — NMI, PendSV, SVCall SysTick, and HardFault are all system exceptions handled by system handlers.

25.3.3.4 Vector table

The vector table contains the reset value of the stack pointer, and the start addresses, also called exception vectors, for all exception handlers. Figure 25–68 shows the order of the exception vectors in the vector table. The least-significant bit of each vector must be 1, indicating that the exception handler is written in Thumb code.

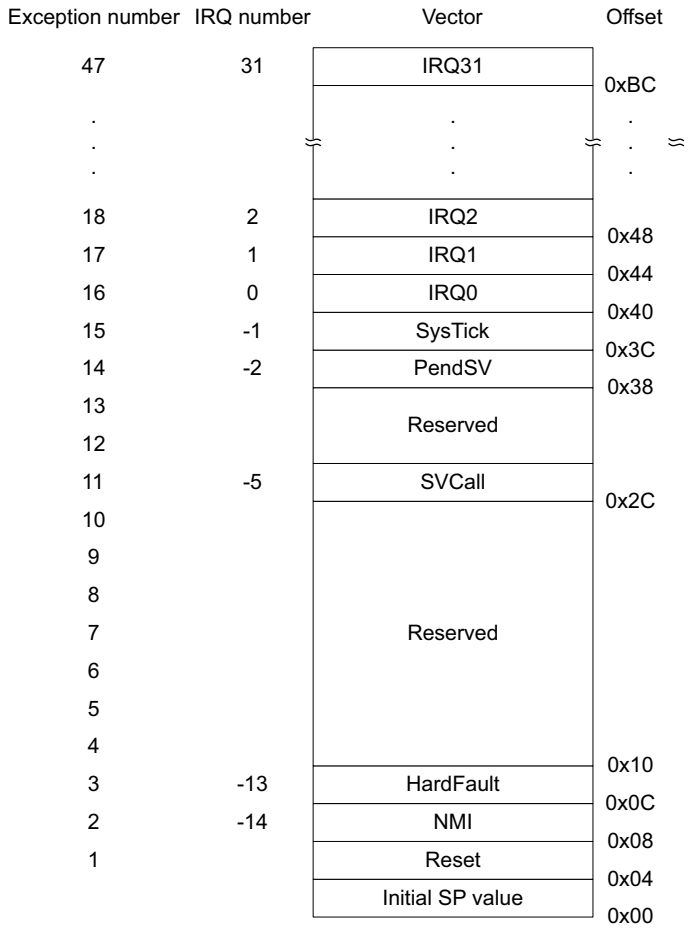


Fig 68. Vector table

The vector table is fixed at address `0x00000000`.

25.3.3.5 Exception priorities

As Table 25–363 shows, all exceptions have an associated priority, with:

- a lower priority value indicating a higher priority
- configurable priorities for all exceptions except Reset, HardFault, and NMI.

If software does not configure any priorities, then all exceptions with a configurable priority have a priority of 0. For information about configuring exception priorities see

- Section 25–25.5.3.7
- Section 25–25.5.2.6.

Configurable priority values are in the range 0-192, in steps of 64. The Reset, HardFault, and NMI exceptions, with fixed negative priority values, always have higher priority than any other exception.

Assigning a higher priority value to IRQ[0] and a lower priority value to IRQ[1] means that IRQ[1] has higher priority than IRQ[0]. If both IRQ[1] and IRQ[0] are asserted, IRQ[1] is processed before IRQ[0].

If multiple pending exceptions have the same priority, the pending exception with the lowest exception number takes precedence. For example, if both IRQ[0] and IRQ[1] are pending and have the same priority, then IRQ[0] is processed before IRQ[1].

When the processor is executing an exception handler, the exception handler is preempted if a higher priority exception occurs. If an exception occurs with the same priority as the exception being handled, the handler is not preempted, irrespective of the exception number. However, the status of the new interrupt changes to pending.

25.3.3.6 Exception entry and return

Descriptions of exception handling use the following terms:

Preemption — When the processor is executing an exception handler, an exception can preempt the exception handler if its priority is higher than the priority of the exception being handled.

When one exception preempts another, the exceptions are called nested exceptions. See Section 25–25.3.3.6.1 for more information.

Return — This occurs when the exception handler is completed, and:

- there is no pending exception with sufficient priority to be serviced
- the completed exception handler was not handling a late-arriving exception.

The processor pops the stack and restores the processor state to the state it had before the interrupt occurred. See Section 25–25.3.3.6.2 for more information.

Tail-chaining — This mechanism speeds up exception servicing. On completion of an exception handler, if there is a pending exception that meets the requirements for exception entry, the stack pop is skipped and control transfers to the new exception handler.

Late-arriving — This mechanism speeds up preemption. If a higher priority exception occurs during state saving for a previous exception, the processor switches to handle the higher priority exception and initiates the vector fetch for that exception. State saving is not affected by late arrival because the state saved would be the same for both exceptions. On return from the exception handler of the late-arriving exception, the normal tail-chaining rules apply.

25.3.3.6.1 Exception entry

Exception entry occurs when there is a pending exception with sufficient priority and either:

- the processor is in Thread mode
- the new exception is of higher priority than the exception being handled, in which case the new exception preempts the exception being handled.

When one exception preempts another, the exceptions are nested.

Sufficient priority means the exception has greater priority than any limit set by the mask register, see Section 25–25.3.1.3.6. An exception with less priority than this is pending but is not handled by the processor.

When the processor takes an exception, unless the exception is a tail-chained or a late-arriving exception, the processor pushes information onto the current stack. This operation is referred to as **stacking** and the structure of eight data words is referred to as a **stack frame**. The stack frame contains the following information:

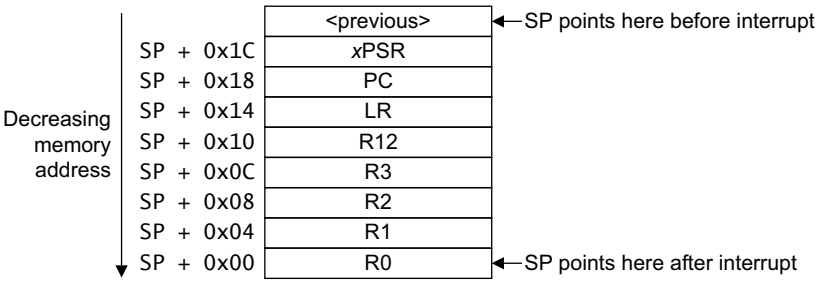


Fig 69. Exception entry stack contents

Immediately after stacking, the stack pointer indicates the lowest address in the stack frame. The stack frame is aligned to a double-word address.

The stack frame includes the return address. This is the address of the next instruction in the interrupted program. This value is restored to the PC at exception return so that the interrupted program resumes.

The processor performs a vector fetch that reads the exception handler start address from the vector table. When stacking is complete, the processor starts executing the exception handler. At the same time, the processor writes an EXC_RETURN value to the LR. This indicates which stack pointer corresponds to the stack frame and what operation mode the processor was in before the entry occurred.

If no higher priority exception occurs during exception entry, the processor starts executing the exception handler and automatically changes the status of the corresponding pending interrupt to active.

If another higher priority exception occurs during exception entry, the processor starts executing the exception handler for this exception and does not change the pending status of the earlier exception. This is the late arrival case.

25.3.3.6.2 Exception return

Exception return occurs when the processor is in Handler mode and execution of one of the following instructions attempts to set the PC to an EXC_RETURN value:

- a `POP` instruction that loads the PC
- a `BX` instruction using any register.

The processor saves an EXC_RETURN value to the LR on exception entry. The exception mechanism relies on this value to detect when the processor has completed an exception handler. Bits[31:4] of an EXC_RETURN value are `0xFFFFFFF`. When the processor loads a value matching this pattern to the PC it detects that the operation is a

not a normal branch operation and, instead, that the exception is complete. Therefore, it starts the exception return sequence. Bits[3:0] of the EXC_RETURN value indicate the required return stack and processor mode, as Table 25–364 shows.

Table 364. Exception return behavior

EXC_RETURN	Description
0xFFFFFFF1	Return to Handler mode. Exception return gets state from the main stack. Execution uses MSP after return.
0xFFFFFFF9	Return to Thread mode. Exception return gets state from MSP. Execution uses MSP after return.
0xFFFFFFFD	Return to Thread mode. Exception return gets state from PSP. Execution uses PSP after return.
All other values	Reserved.

25.3.4 Fault handling

Faults are a subset of exceptions, see Section 25–25.3.3. All faults result in the HardFault exception being taken or cause lockup if they occur in the NMI or HardFault handler. The faults are:

- execution of an SVC instruction at a priority equal or higher than SVCall
- execution of a BKPT instruction without a debugger attached
- a system-generated bus error on a load or store
- execution of an instruction from an XN memory address
- execution of an instruction from a location for which the system generates a bus fault
- a system-generated bus error on a vector fetch
- execution of an Undefined instruction
- execution of an instruction when not in Thumb-State as a result of the T-bit being previously cleared to 0
- an attempted load or store to an unaligned address.

Only Reset and NMI can preempt the fixed priority HardFault handler. A HardFault can preempt any exception other than Reset, NMI, or another hard fault.

25.3.4.1 Lockup

The processor enters a lockup state if a fault occurs when executing the NMI or HardFault handlers, or if the system generates a bus error when unstacking the PSR on an exception return using the MSP. When the processor is in lockup state it does not execute any instructions. The processor remains in lockup state until one of the following occurs:

- it is reset
- a debugger halts it
- an NMI occurs and the current lockup is in the HardFault handler.

If lockup state occurs in the NMI handler a subsequent NMI does not cause the processor to leave lockup state.

25.3.5 Power management

The Cortex-M0 processor sleep modes reduce power consumption:

- a sleep mode, that stops the processor clock
- a Deep-sleep mode, details <tbd>.

The SLEEPDEEP bit of the SCR selects which sleep mode is used, see Section 25–25.5.3.5.

This section describes the mechanisms for entering sleep mode, and the conditions for waking up from sleep mode.

25.3.5.1 Entering sleep mode

This section describes the mechanisms software can use to put the processor into sleep mode.

The system can generate spurious wake-up events, for example a debug operation wakes up the processor. Therefore software must be able to put the processor back into sleep mode after such an event. A program might have an idle loop to put the processor back in to sleep mode.

25.3.5.1.1 Wait for interrupt

The Wait For Interrupt instruction, `WFI`, causes immediate entry to sleep mode. When the processor executes a `WFI` instruction it stops executing instructions and enters sleep mode. See Section 25–25.4.7.12 for more information.

25.3.5.1.2 Wait for event

Remark: The WFE instruction is not implemented on the LPC11U1x.

The Wait For Event instruction, `WFE`, causes entry to sleep mode conditional on the value of a one-bit event register. When the processor executes a `WFE` instruction, it checks the value of the event register:

0 — The processor stops executing instructions and enters sleep mode

1 — The processor sets the register to zero and continues executing instructions without entering sleep mode.

See Section 25–25.4.7.11 for more information.

If the event register is 1, this indicates that the processor must not enter sleep mode on execution of a `WFE` instruction. Typically, this is because of the assertion of an external event, or because another processor in the system has executed a `SEV` instruction, see Section 25–25.4.7.9. Software cannot access this register directly.

25.3.5.1.3 Sleep-on-exit

If the SLEEPONEXIT bit of the SCR is set to 1, when the processor completes the execution of an exception handler and returns to Thread mode it immediately enters sleep mode. Use this mechanism in applications that only require the processor to run when an interrupt occurs.

25.3.5.2 Wake-up from sleep mode

The conditions for the processor to wake-up depend on the mechanism that caused it to enter sleep mode.

25.3.5.2.1 Wake-up from WFI or sleep-on-exit

Normally, the processor wakes up only when it detects an exception with sufficient priority to cause exception entry.

Some embedded systems might have to execute system restore tasks after the processor wakes up, and before it executes an interrupt handler. To achieve this set the PRIMASK bit to 1. If an interrupt arrives that is enabled and has a higher priority than current exception priority, the processor wakes up but does not execute the interrupt handler until the processor sets PRIMASK to zero. For more information about PRIMASK, see Section 25–25.3.1.3.6.

25.3.5.2.2 Wake-up from WFE

The processor wakes up if:

- it detects an exception with sufficient priority to cause exception entry
- in a multiprocessor system, another processor in the system executes a `SEV` instruction.

In addition, if the SEVONPEND bit in the SCR is set to 1, any new pending interrupt triggers an event and wakes up the processor, even if the interrupt is disabled or has insufficient priority to cause exception entry. For more information about the SCR see Section 25–25.5.3.5.

25.3.5.3 Power management programming hints

ISO/IEC C cannot directly generate the `WFI`, `WFE`, and `SEV` instructions. The CMSIS provides the following intrinsic functions for these instructions:

```
void __WFE(void) // Wait for Event

void __WFI(void) // Wait for Interrupt

void __SEV(void) // Send Event
```

25.4 Instruction set

25.4.1 Instruction set summary

The processor implements a version of the Thumb instruction set. Table 365 lists the supported instructions.

Remark: In Table 365

- angle brackets, <>, enclose alternative forms of the operand
- braces, {}, enclose optional operands and mnemonic parts
- the Operands column is not exhaustive.

For more information on the instructions and operands, see the instruction descriptions.

Table 365. Cortex-M0 instructions

Mnemonic	Operands	Brief description	Flags	Reference
ADCS	*{Rd,} Rn, Rm*	Add with Carry	N,Z,C,V	Section 25–25.4.5.1
ADD{S}	*{Rd,} Rn, <Rm|#imm>*	Add	N,Z,C,V	Section 25–25.4.5.1
ADR	*Rd, label*	PC-relative Address to Register	-	Section 25–25.4.4.1
ANDS	*{Rd,} Rn, Rm*	Bitwise AND	N,Z	Section 25–25.4.5.1
ASRS	*{Rd,} Rm, <Rs|#imm>*	Arithmetic Shift Right	N,Z,C	Section 25–25.4.5.3
B{cc}	*label*	Branch {conditionally}	-	Section 25–25.4.6.1
BICS	*{Rd,} Rn, Rm*	Bit Clear	N,Z	Section 25–25.4.5.2
BKPT	*#imm*	Breakpoint	-	Section 25–25.4.7.1
BL	*label*	Branch with Link	-	Section 25–25.4.6.1
BLX	*Rm*	Branch indirect with Link	-	Section 25–25.4.6.1
BX	*Rm*	Branch indirect	-	Section 25–25.4.6.1
CMN	*Rn, Rm*	Compare Negative	N,Z,C,V	Section 25–25.4.5.4
CMP	*Rn, <Rm|#imm>*	Compare	N,Z,C,V	Section 25–25.4.5.4
CPSID	*i*	Change Processor State, Disable Interrupts	-	Section 25–25.4.7.2
CPSIE	*i*	Change Processor State, Enable Interrupts	-	Section 25–25.4.7.2
DMB	-	Data Memory Barrier	-	Section 25–25.4.7.3
DSB	-	Data Synchronization Barrier	-	Section 25–25.4.7.4
EORS	*{Rd,} Rn, Rm*	Exclusive OR	N,Z	Section 25–25.4.5.2
ISB	-	Instruction Synchronization Barrier	-	Section 25–25.4.7.5
LDM	*Rn{!}, reglist*	Load Multiple registers, increment after	-	Section 25–25.4.4.5

Table 365. Cortex-M0 instructions

Mnemonic	Operands	Brief description	Flags	Reference
LDR	*Rt, label*	Load Register from PC-relative address	-	Section 25–25.4.4
LDR	*Rt, [Rn, <Rm|#imm>]*	Load Register with word	-	Section 25–25.4.4
LDRB	*Rt, [Rn, <Rm|#imm>]*	Load Register with byte	-	Section 25–25.4.4
LDRH	*Rt, [Rn, <Rm|#imm>]*	Load Register with halfword	-	Section 25–25.4.4
LDRSB	*Rt, [Rn, <Rm|#imm>]*	Load Register with signed byte	-	Section 25–25.4.4
LDRSH	*Rt, [Rn, <Rm|#imm>]*	Load Register with signed halfword	-	Section 25–25.4.4
LSLS	*{Rd,} Rn, <Rs|#imm>*	Logical Shift Left	N,Z,C	Section 25–25.4.5.3
U	*{Rd,} Rn, <Rs|#imm>*	Logical Shift Right	N,Z,C	Section 25–25.4.5.3
MOV{S}	*Rd, Rm*	Move	N,Z	Section 25–25.4.5.5
MRS	*Rd, spec_reg*	Move to general register from special register	-	Section 25–25.4.7.6
MSR	*spec_reg, Rm*	Move to special register from general register	N,Z,C,V	Section 25–25.4.7.7
MULS	*Rd, Rn, Rm*	Multiply, 32-bit result	N,Z	Section 25–25.4.5.6
MVNS	*Rd, Rm*	Bitwise NOT	N,Z	Section 25–25.4.5.5
NOP	-	No Operation	-	Section 25–25.4.7.8
ORRS	*{Rd,} Rn, Rm*	Logical OR	N,Z	Section 25–25.4.5.2
POP	*reglist*	Pop registers from stack	-	Section 25–25.4.4.6
PUSH	*reglist*	Push registers onto stack	-	Section 25–25.4.4.6
REV	*Rd, Rm*	Byte-Reverse word	-	Section 25–25.4.5.7
REV16	*Rd, Rm*	Byte-Reverse packed halfwords	-	Section 25–25.4.5.7
REVSH	*Rd, Rm*	Byte-Reverse signed halfword	-	Section 25–25.4.5.7
RORS	*{Rd,} Rn, Rs*	Rotate Right	N,Z,C	Section 25–25.4.5.3
RSBS	*{Rd,} Rn, #0*	Reverse Subtract	N,Z,C,V	Section 25–25.4.5.1
SBCS	*{Rd,} Rn, Rm*	Subtract with Carry	N,Z,C,V	Section 25–25.4.5.1
SEV	-	Send Event	-	Section 25–25.4.7.9
STM	*Rn!, reglist*	Store Multiple registers, increment after	-	Section 25–25.4.4.5

Table 365. Cortex-M0 instructions

Mnemonic	Operands	Brief description	Flags	Reference
STR	*Rt, [Rn, <Rm|#imm>]*	Store Register as word	-	Section 25–25.4.4
STRB	*Rt, [Rn, <Rm|#imm>]*	Store Register as byte	-	Section 25–25.4.4
STRH	*Rt, [Rn, <Rm|#imm>]*	Store Register as halfword	-	Section 25–25.4.4
SUB{S}	*{Rd,} Rn, <Rm|#imm>*	Subtract	N,Z,C,V	Section 25–25.4.5.1
SVC	*#imm*	Supervisor Call	-	Section 25–25.4.7.10
SXTB	*Rd, Rm*	Sign extend byte	-	Section 25–25.4.5.8
SXTH	*Rd, Rm*	Sign extend halfword	-	Section 25–25.4.5.8
TST	*Rn, Rm*	Logical AND based test	N,Z	Section 25–25.4.5.9
UXTB	*Rd, Rm*	Zero extend a byte	-	Section 25–25.4.5.8
UXTH	*Rd, Rm*	Zero extend a halfword	-	Section 25–25.4.5.8
WFE	-	Wait For Event	-	Section 25–25.4.7.11
WFI	-	Wait For Interrupt	-	Section 25–25.4.7.12

25.4.2 Intrinsic functions

ISO/IEC C code cannot directly access some Cortex-M0 instructions. This section describes intrinsic functions that can generate these instructions, provided by the CMSIS and that might be provided by a C compiler. If a C compiler does not support an appropriate intrinsic function, you might have to use inline assembler to access the relevant instruction.

The CMSIS provides the following intrinsic functions to generate instructions that ISO/IEC C code cannot directly access:

Table 366. CMSIS intrinsic functions to generate some Cortex-M0 instructions

Instruction	CMSIS intrinsic function
CPSIE i	void __enable_irq(void)
CPSID i	void __disable_irq(void)
ISB	void __ISB(void)
DSB	void __DSB(void)
DMB	void __DMB(void)
NOP	void __NOP(void)
REV	uint32_t __REV(uint32_t int value)
REV16	uint32_t __REV16(uint32_t int value)
REVSH	uint32_t __REVSH(uint32_t int value)

Table 366. CMSIS intrinsic functions to generate some Cortex-M0 instructions

Instruction	CMSIS intrinsic function
SEV	void __SEV(void)
WFE	void __WFE(void)
WFI	void __WFI(void)

The CMSIS also provides a number of functions for accessing the special registers using MRS and MSR instructions:

Table 367. CMSIS intrinsic functions to access the special registers

Special register	Access	CMSIS function
PRIMASK	Read	uint32_t __get_PRIMASK (void)
	Write	void __set_PRIMASK (uint32_t value)
CONTROL	Read	uint32_t __get_CONTROL (void)
	Write	void __set_CONTROL (uint32_t value)
MSP	Read	uint32_t __get_MSP (void)
	Write	void __set_MSP (uint32_t TopOfMainStack)
PSP	Read	uint32_t __get_PSP (void)
	Write	void __set_PSP (uint32_t TopOfProcStack)

25.4.3 About the instruction descriptions

The following sections give more information about using the instructions:

- Section 25.4.3.1 “Operands”
- Section 25.4.3.2 “Restrictions when using PC or SP”
- Section 25.4.3.3 “Shift Operations”
- Section 25.4.3.4 “Address alignment”
- Section 25.4.3.5 “PC-relative expressions”
- Section 25.4.3.6 “Conditional execution”.

25.4.3.1 Operands

An instruction operand can be an ARM register, a constant, or another instruction-specific parameter. Instructions act on the operands and often store the result in a destination register. When there is a destination register in the instruction, it is usually specified before the other operands.

25.4.3.2 Restrictions when using PC or SP

Many instructions are unable to use, or have restrictions on whether you can use, the **Program Counter** (PC) or **Stack Pointer** (SP) for the operands or destination register. See instruction descriptions for more information.

Remark: When you update the PC with a BX, BLX, or POP instruction, bit[0] of any address must be 1 for correct execution. This is because this bit indicates the destination instruction set, and the Cortex-M0 processor only supports Thumb instructions. When a BL or BLX instruction writes the value of bit[0] into the LR it is automatically assigned the value 1.

25.4.3.3 Shift Operations

Register shift operations move the bits in a register left or right by a specified number of bits, the **shift length**. Register shift can be performed directly by the instructions ASR, LSR, LSL, and ROR and the result is written to a destination register.The permitted shift lengths depend on the shift type and the instruction, see the individual instruction description. If the shift length is 0, no shift occurs. Register shift operations update the carry flag except when the specified shift length is 0. The following sub-sections describe the various shift operations and how they affect the carry flag. In these descriptions, *Rm* is the register containing the value to be shifted, and *n* is the shift length.

25.4.3.3.1 ASR

Arithmetic shift right by *n* bits moves the left-hand 32 -*n* bits of the register *Rm*, to the right by *n* places, into the right-hand 32 -*n* bits of the result, and it copies the original bit[31] of the register into the left-hand *n* bits of the result. See Figure 25–70.

You can use the ASR operation to divide the signed value in the register *Rm* by 2^n, with the result being rounded towards negative-infinity.

When the instruction is ASRS the carry flag is updated to the last bit shifted out, bit[*n*-1], of the register *Rm*.

Remark:

- If *n* is 32 or more, then all the bits in the result are set to the value of bit[31] of *Rm*.
- If *n* is 32 or more and the carry flag is updated, it is updated to the value of bit[31] of *Rm*.

Fig 70. ASR #3

25.4.3.3.2 LSR

Logical shift right by *n* bits moves the left-hand 32-*n* bits of the register *Rm*, to the right by *n* places, into the right-hand 32 -*n* bits of the result, and it sets the left-hand *n* bits of the result to 0. See Figure 71.

You can use the LSR operation to divide the value in the register *Rm* by 2^n, if the value is regarded as an unsigned integer.

When the instruction is LSRS, the carry flag is updated to the last bit shifted out, bit[*n*-1], of the register *Rm*.

Remark:

- If *n* is 32 or more, then all the bits in the result are cleared to 0.
- If *n* is 33 or more and the carry flag is updated, it is updated to 0.

Fig 71. LSR #3

25.4.3.3.3 LSL

Logical shift left by *n* bits moves the right-hand 32-*n* bits of the register *Rm*, to the left by *n* places, into the left-hand 32-*n* bits of the result, and it sets the right-hand *n* bits of the result to 0. See Figure 72.

You can use the LSL operation to multiply the value in the register *Rm* by 2^n, if the value is regarded as an unsigned integer or a two's complement signed integer. Overflow can occur without warning.

When the instruction is LSLS the carry flag is updated to the last bit shifted out, bit[32-*n*], of the register *Rm*. These instructions do not affect the carry flag when used with LSL #0.

Remark:

- If *n* is 32 or more, then all the bits in the result are cleared to 0.
- If *n* is 33 or more and the carry flag is updated, it is updated to 0.

Fig 72. LSL #3

25.4.3.3.4 ROR

Rotate right by *n* bits moves the left-hand 32-*n* bits of the register *Rm*, to the right by *n* places, into the right-hand 32-*n* bits of the result, and it moves the right-hand *n* bits of the register into the left-hand *n* bits of the result. See Figure 25–73.

When the instruction is RORS the carry flag is updated to the last bit rotation, bit[*n*-1], of the register *Rm*.

Remark:

- If *n* is 32, then the value of the result is same as the value in *Rm*, and if the carry flag is updated, it is updated to bit[31] of *Rm*.
- ROR

  with shift length, *n*, greater than 32 is the same as

  ROR

  with shift length *n*-32.

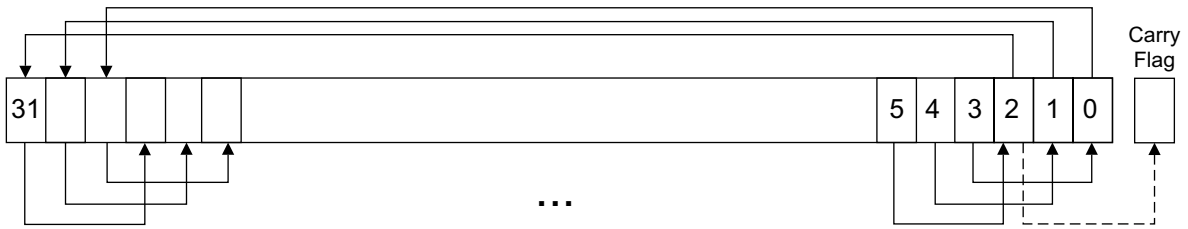


Fig 73. ROR #3

25.4.3.4 Address alignment

An aligned access is an operation where a word-aligned address is used for a word, or multiple word access, or where a halfword-aligned address is used for a halfword access. Byte accesses are always aligned.

There is no support for unaligned accesses on the Cortex-M0 processor. Any attempt to perform an unaligned memory access operation results in a HardFault exception.

25.4.3.5 PC-relative expressions

A PC-relative expression or **label** is a symbol that represents the address of an instruction or literal data. It is represented in the instruction as the PC value plus or minus a numeric offset. The assembler calculates the required offset from the label and the address of the current instruction. If the offset is too big, the assembler produces an error.

Remark:

- For most instructions, the value of the PC is the address of the current instruction plus 4 bytes.
- Your assembler might permit other syntaxes for PC-relative expressions, such as a label plus or minus a number, or an expression of the form [PC, #*imm*].

25.4.3.6 Conditional execution

Most data processing instructions update the condition flags in the **Application Program Status Register** (APSR) according to the result of the operation, see Section . Some instructions update all flags, and some only update a subset. If a flag is not updated, the original value is preserved. See the instruction descriptions for the flags they affect.

You can execute a conditional branch instruction, based on the condition flags set in another instruction, either:

- immediately after the instruction that updated the flags
- after any number of intervening instructions that have not updated the flags.

On the Cortex-M0 processor, conditional execution is available by using conditional branches.

This section describes:

- Section 25.4.3.6.1 “The condition flags”
- Section 25.4.3.6.2 “Condition code suffixes”.

25.4.3.6.1 The condition flags

The APSR contains the following condition flags:

N — Set to 1 when the result of the operation was negative, cleared to 0 otherwise.

Z — Set to 1 when the result of the operation was zero, cleared to 0 otherwise.

C — Set to 1 when the operation resulted in a carry, cleared to 0 otherwise.

V — Set to 1 when the operation caused overflow, cleared to 0 otherwise.

For more information about the APSR see Section 25–25.3.1.3.5.

A carry occurs:

- if the result of an addition is greater than or equal to 2^{32}
- if the result of a subtraction is positive or zero
- as the result of a shift or rotate instruction.

Overflow occurs when the sign of the result, in bit[31], does not match the sign of the result had the operation been performed at infinite precision, for example:

- if adding two negative values results in a positive value
- if adding two positive values results in a negative value
- if subtracting a positive value from a negative value generates a positive value
- if subtracting a negative value from a positive value generates a negative value.

The Compare operations are identical to subtracting, for CMP, or adding, for CMN, except that the result is discarded. See the instruction descriptions for more information.

25.4.3.6.2 Condition code suffixes

Conditional branch is shown in syntax descriptions as B{*cond*}. A branch instruction with a condition code is only taken if the condition code flags in the APSR meet the specified condition, otherwise the branch instruction is ignored. shows the condition codes to use.

Table 368 also shows the relationship between condition code suffixes and the N, Z, C, and V flags.

Table 368. Condition code suffixes

Suffix	Flags	Meaning
EQ	Z = 1	Equal, last flag setting result was zero
NE	Z = 0	Not equal, last flag setting result was non-zero
CS or HS	C = 1	Higher or same, unsigned
CC or LO	C = 0	Lower, unsigned
MI	N = 1	Negative

Table 368. Condition code suffixes

Suffix	Flags	Meaning
PL	N = 0	Positive or zero
VS	V = 1	Overflow
VC	V = 0	No overflow
HI	C = 1 and Z = 0	Higher, unsigned
LS	C = 0 or Z = 1	Lower or same, unsigned
GE	N = V	Greater than or equal, signed
LT	N != V	Less than, signed
GT	Z = 0 and N = V	Greater than, signed
LE	Z = 1 and N != V	Less than or equal, signed
AL	Can have any value	Always. This is the default when no suffix is specified.

25.4.4 Memory access instructions

Table 369 shows the memory access instructions:

Table 369. Access instructions

Mnemonic	Brief description	See
LDR{type}	Load Register using register offset	Section 25–25.4.4.3
LDR	Load Register from PC-relative address	Section 25–25.4.4.4
POP	Pop registers from stack	Section 25–25.4.4.6
PUSH	Push registers onto stack	Section 25–25.4.4.6
STM	Store Multiple registers	Section 25–25.4.4.5
STR{type}	Store Register using immediate offset	Section 25–25.4.4.2
STR{type}	Store Register using register offset	Section 25–25.4.4.3

25.4.4.1 ADR

Generates a PC-relative address.

25.4.4.1.1 Syntax

ADR *Rd*, *label*

where:

Rd is the destination register.

label is a PC-relative expression. See Section 25–25.4.3.5.

25.4.4.1.2 Operation

ADR generates an address by adding an immediate value to the PC, and writes the result to the destination register.

ADR facilitates the generation of position-independent code, because the address is PC-relative.

If you use ADR to generate a target address for a BX or BLX instruction, you must ensure that bit[0] of the address you generate is set to 1 for correct execution.

25.4.4.1.3 Restrictions

In this instruction *Rd* must specify R0-R7. The data-value addressed must be word aligned and within 1020 bytes of the current PC.

25.4.4.1.4 Condition flags

This instruction does not change the flags.

25.4.4.1.5 Examples

```
ADR    R1, TextMessage   ; Write address value of a location labelled as
                               ; TextMessage to R1

ADR    R3, [PC,#996]     ; Set R3 to value of PC + 996.
```

25.4.4.2 LDR and STR, immediate offset

Load and Store with immediate offset.

25.4.4.2.1 Syntax

LDR *Rt*, [<*Rn* | SP> {, #*imm*}]

LDR<B|H> *Rt*, [*Rn* {, #*imm*}]

STR *Rt*, [<*Rn* | SP>, {,#*imm*}]

STR<B|H> *Rt*, [*Rn* {,#*imm*}]

where:

Rt is the register to load or store.

Rn is the register on which the memory address is based.

imm is an offset from *Rn*. If *imm* is omitted, it is assumed to be zero.

25.4.4.2.2 Operation

LDR, LDRB and LDRH instructions load the register specified by *Rt* with either a word, byte or halfword data value from memory. Sizes less than word are zero extended to 32-bits before being written to the register specified by *Rt*.

STR, STRB and STRH instructions store the word, least-significant byte or lower halfword contained in the single register specified by *Rt* in to memory. The memory address to load from or store to is the sum of the value in the register specified by either *Rn* or SP and the immediate value *imm*.

25.4.4.2.3 Restrictions

In these instructions:

- *Rt* and *Rn* must only specify R0-R7.

- *imm* must be between:
  - 0 and 1020 and an integer multiple of four for LDR and STR using SP as the base register
  - 0 and 124 and an integer multiple of four for LDR and STR using R0-R7 as the base register
  - 0 and 62 and an integer multiple of two for LDRH and STRH
  - 0 and 31 for LDRB and STRB.
- The computed address must be divisible by the number of bytes in the transaction, see Section 25–25.4.3.4.

25.4.4.2.4 Condition flags

These instructions do not change the flags.

25.4.4.2.5 Examples

```
LDR     R4, [R7                ; Loads R4 from the address in R7.
STR     R2, [R0,#const-struc]  ; const-struc is an expression evaluating
                               ; to a constant in the range 0-1020.
```

25.4.4.3 LDR and STR, register offset

Load and Store with register offset.

25.4.4.3.1 Syntax

LDR *Rt*, [*Rn*, *Rm*]

LDR<B|H> *Rt*, [*Rn*, *Rm*]

LDR<SB|SH> *Rt*, [*Rn*, *Rm*]

STR *Rt*, [*Rn*, *Rm*]

STR<B|H> *Rt*, [*Rn*, *Rm*]

where:

Rt is the register to load or store.

Rn is the register on which the memory address is based.

Rm is a register containing a value to be used as the offset.

25.4.4.3.2 Operation

LDR, LDRB, U, LDRSB and LDRSH load the register specified by *Rt* with either a word, zero extended byte, zero extended halfword, sign extended byte or sign extended halfword value from memory.

STR, STRB and STRH store the word, least-significant byte or lower halfword contained in the single register specified by *Rt* into memory.

The memory address to load from or store to is the sum of the values in the registers specified by *Rn* and *Rm*.

25.4.4.3.3 Restrictions

In these instructions:

- *Rt*, *Rn*, and *Rm* must only specify R0-R7.
- the computed memory address must be divisible by the number of bytes in the load or store, see Section 25–25.4.3.4.

25.4.4.3.4 Condition flags

These instructions do not change the flags.

25.4.4.3.5 Examples

```
STR    R0, [R5, R1]         ; Store value of R0 into an address equal to
                                ; sum of R5 and R1

LDRSH  R1, [R2, R3]         ; Load a halfword from the memory address
                                ; specified by (R2 + R3), sign extend to 32-bits

                            ; and write to R1.
```

25.4.4.4 LDR, PC-relative

Load register (literal) from memory.

25.4.4.4.1 Syntax

LDR *Rt*, *label*

where:

Rt is the register to load.

label is a PC-relative expression. See Section 25–25.4.3.5.

25.4.4.4.2 Operation

Loads the register specified by *Rt* from the word in memory specified by *label*.

25.4.4.4.3 Restrictions

In these instructions, *label* must be within 1020 bytes of the current PC and word aligned.

25.4.4.4.4 Condition flags

These instructions do not change the flags.

25.4.4.4.5 Examples

```
LDR     R0, LookUpTable   ; Load R0 with a word of data from an address
                              ; labelled as LookUpTable.

LDR     R3, [PC, #100]    ; Load R3 with memory word at (PC + 100).
```

25.4.4.5 LDM and STM

Load and Store Multiple registers.

25.4.4.5.1 Syntax

LDM *Rn*{!}, reglist

STM *Rn*!, reglist

where:

Rn is the register on which the memory addresses are based.

! writeback suffix.

reglist is a list of one or more registers to be loaded or stored, enclosed in braces. It can contain register ranges. It must be comma separated if it contains more than one register or register range, see Section 25–25.4.4.5.5.

LDMIA and LDMFD are synonyms for LDM. LDMIA refers to the base register being Incremented After each access. LDMFD refers to its use for popping data from Full Descending stacks.

STMIA and STMEA are synonyms for STM. STMIA refers to the base register being Incremented After each access. STMEA refers to its use for pushing data onto Empty Ascending stacks.

25.4.4.5.2 Operation

LDM instructions load the registers in *reglist* with word values from memory addresses based on *Rn*.

STM instructions store the word values in the registers in *reglist* to memory addresses based on *Rn*.

The memory addresses used for the accesses are at 4-byte intervals ranging from the value in the register specified by *Rn* to the value in the register specified by *Rn* + 4 * (*n*-1), where *n* is the number of registers in *reglist*. The accesses happens in order of increasing register numbers, with the lowest numbered register using the lowest memory address and the highest number register using the highest memory address. If the writeback suffix is specified, the value in the register specified by *Rn* + 4 **n* is written back to the register specified by *Rn*.

25.4.4.5.3 Restrictions

In these instructions:

- *reglist* and *Rn* are limited to R0-R7.
- the writeback suffix must always be used unless the instruction is an LDM where reglist also contains *Rn*, in which case the writeback suffix must not be used.
- the value in the register specified by *Rn* must be word aligned. See Section 25–25.4.3.4 for more information.
- for STM, if *Rn* appears in *reglist*, then it must be the first register in the list.

25.4.4.5.4 Condition flags

These instructions do not change the flags.

25.4.4.5.5 Examples

```
LDM     R0,{R0,R3,R4}      ; LDMIA is a synonym for LDM
      STMIA   R1!,{R2-R4,R6}
```

25.4.4.5.6 Incorrect examples

```
STM     R5!,{R4,R5,R6} ; Value stored for R5 is unpredictable
      LDM     R2,{}          ; There must be at least one register in the list
```

25.4.4.6 PUSH and POP

Push registers onto, and pop registers off a full-descending stack.

25.4.4.6.1 Syntax

PUSH *reglist*

POP *reglist*

where:

reglist is a non-empty list of registers, enclosed in braces. It can contain register ranges. It must be comma separated if it contains more than one register or register range.

25.4.4.6.2 Operation

PUSH stores registers on the stack, with the lowest numbered register using the lowest memory address and the highest numbered register using the highest memory address.

POP loads registers from the stack, with the lowest numbered register using the lowest memory address and the highest numbered register using the highest memory address.

PUSH uses the value in the SP register minus four as the highest memory address,

POP uses the value in the SP register as the lowest memory address, implementing a full-descending stack. On completion,

PUSH updates the SP register to point to the location of the lowest store value,

POP updates the SP register to point to the location above the highest location loaded.

If a POP instruction includes PC in its *reglist*, a branch to this location is performed when the POP instruction has completed. Bit[0] of the value read for the PC is used to update the APSR T-bit. This bit must be 1 to ensure correct operation.

25.4.4.6.3 Restrictions

In these instructions:

- *reglist* must use only R0-R7.
- The exception is LR for a PUSH and PC for a POP.

25.4.4.6.4 Condition flags

These instructions do not change the flags.

25.4.4.6.5 Examples

```
PUSH    {R0,R4-R7}      ; Push R0,R4,R5,R6,R7 onto the stack
        PUSH    {R2,LR}         ; Push R2 and the link-register onto the stack
        POP     {R0,R6,PC}      ; Pop r0,r6 and PC from the stack, then branch to
                                ; the new PC.
```

25.4.5 General data processing instructions

Table 370 shows the data processing instructions:

Table 370. Data processing instructions

Mnemonic	Brief description	See
ADCS	Add with Carry	Section 25–25.4.5.1
ADD{S}	Add	Section 25–25.4.5.1
ANDS	Logical AND	Section 25–25.4.5.2
ASRS	Arithmetic Shift Right	Section 25–25.4.5.3
BICS	Bit Clear	Section 25–25.4.5.2
CMN	Compare Negative	Section 25–25.4.5.4
CMP	Compare	Section 25–25.4.5.4
EORS	Exclusive OR	Section 25–25.4.5.2
LSLS	Logical Shift Left	Section 25–25.4.5.3
LSRS	Logical Shift Right	Section 25–25.4.5.3
MOV{S}	Move	Section 25–25.4.5.5
MULS	Multiply	Section 25–25.4.5.6
MVNS	Move NOT	Section 25–25.4.5.5
ORRS	Logical OR	Section 25–25.4.5.2
REV	Reverse byte order in a word	Section 25–25.4.5.7
REV16	Reverse byte order in each halfword	Section 25–25.4.5.7
REVSH	Reverse byte order in bottom halfword and sign extend	Section 25–25.4.5.7
RORS	Rotate Right	Section 25–25.4.5.3
RSBS	Reverse Subtract	Section 25–25.4.5.1
SBCS	Subtract with Carry	Section 25–25.4.5.1
SUBS	Subtract	Section 25–25.4.5.1
SXTB	Sign extend a byte	Section 25–25.4.5.8
SXTH	Sign extend a halfword	Section 25–25.4.5.8
UXTB	Zero extend a byte	Section 25–25.4.5.8
UXTH	Zero extend a halfword	Section 25–25.4.5.8
TST	Test	Section 25–25.4.5.9

25.4.5.1 ADC, ADD, RSB, SBC, and SUB

Add with carry, Add, Reverse Subtract, Subtract with carry, and Subtract.

25.4.5.1.1 Syntax

ADCS {*Rd*,} *Rn*, *Rm*

ADD{S} {*Rd*,} *Rn*, <*Rm*|*#imm*>

RSBS {*Rd*,} *Rn*, *Rm*, *#0*

SBCS {*Rd*,} *Rn*, *Rm*

SUB{S} {*Rd*,} *Rn*,

<*Rm*|*#imm*>

Where:

S causes an ADD or SUB instruction to update flags

Rd specifies the result register

Rn specifies the first source register

Rm specifies the second source register

imm specifies a constant immediate value.

When the optional *Rd* register specifier is omitted, it is assumed to take the same value as *Rn*, for example ADDS R1,R2 is identical to ADDS R1,R1,R2.

25.4.5.1.2 Operation

The ADCS instruction adds the value in *Rn* to the value in *Rm*, adding a further one if the carry flag is set, places the result in the register specified by *Rd* and updates the N, Z, C, and V flags.

The ADD instruction adds the value in *Rn* to the value in *Rm* or an immediate value specified by *imm* and places the result in the register specified by *Rd*.

The ADDS instruction performs the same operation as ADD and also updates the N, Z, C and V flags.

The RSBS instruction subtracts the value in *Rn* from zero, producing the arithmetic negative of the value, and places the result in the register specified by Rd and updates the N, Z, C and V flags.

The SBCS instruction subtracts the value of *Rm* from the value in *Rn*, deducts a further one if the carry flag is set. It places the result in the register specified by Rd and updates the N, Z, C and V flags.

The SUB instruction subtracts the value in *Rm* or the immediate specified by *imm*. It places the result in the register specified by *Rd*.

The SUBS instruction performs the same operation as SUB and also updates the N, Z, C and V flags.

Use ADC and SBC to synthesize multiword arithmetic, see Section 25.4.5.1.4.

See also Section 25–25.4.4.1.

25.4.5.1.3 Restrictions

Table 371 lists the legal combinations of register specifiers and immediate values that can be used with each instruction.

Table 371. ADC, ADD, RSB, SBC and SUB operand restrictions

Instruction	Rd	Rn	Rm	imm	Restrictions
ADCS	R0-R7	R0-R7	R0-R7	-	*Rd* and *Rn* must specify the same register.
ADD	R0-R15	R0-R15	R0-PC	-	*Rd* and *Rn* must specify the same register. *Rn* and *Rm* must not both specify PC.
	R0-R7	SP or PC	-	0-1020	Immediate value must be an integer multiple of four.
	SP	SP	-	0-508	Immediate value must be an integer multiple of four.
ADDS	R0-R7	R0-R7	-	0-7	-
	R0-R7	R0-R7	-	0-255	*Rd* and *Rn* must specify the same register.
	R0-R7	R0-R7	R0-R7	-	-
RSBS	R0-R7	R0-R7	-	-	-
SBCS	R0-R7	R0-R7	R0-R7	-	*Rd* and *Rn* must specify the same register.
SUB	SP	SP	-	0-508	Immediate value must be an integer multiple of four.
SUBS	R0-R7	R0-R7	-	0-7	-
	R0-R7	R0-R7	-	0-255	Rd and *Rn* must specify the same register.
	R0-R7	R0-R7	R0-R7	-	-

25.4.5.1.4 Examples

The following shows two instructions that add a 64-bit integer contained in R0 and R1 to another 64-bit integer contained in R2 and R3, and place the result in R0 and R1.

64-bit addition:

```
ADDS    R0, R0, R2    ; add the least significant words
ADCS    R1, R1, R3    ; add the most significant words with carry
```

Multiword values do not have to use consecutive registers. The following shows instructions that subtract a 96-bit integer contained in R1, R2, and R3 from another contained in R4, R5, and R6. The example stores the result in R4, R5, and R6.

96-bit subtraction:

```
SUBS    R4, R4, R1    ; subtract the least significant words
SBCS    R5, R5, R2    ; subtract the middle words with carry
SBCS    R6, R6, R3    ; subtract the most significant words with carry
```

The following shows the RSBS instruction used to perform a 1's complement of a single register.

Arithmetic negation:

```
RSBS    R7, R7, #0    ; subtract R7 from zero
```

25.4.5.2 AND, ORR, EOR, and BIC

Logical AND, OR, Exclusive OR, and Bit Clear.

25.4.5.2.1 Syntax

ANDS {*Rd,*} *Rn*, *Rm*

ORRS {*Rd,*} *Rn*, *Rm*

EORS {*Rd,*} *Rn*, *Rm*

BICS {*Rd,*} *Rn*, *Rm*

where:

Rd is the destination register.

Rn is the register holding the first operand and is the same as the destination register.

Rm second register.

25.4.5.2.2 Operation

The AND, EOR, and ORR instructions perform bitwise AND, exclusive OR, and inclusive OR operations on the values in *Rn* and *Rm*.

The BIC instruction performs an AND operation on the bits in *Rn* with the logical negation of the corresponding bits in the value of *Rm*.

The condition code flags are updated on the result of the operation, see Section 25.4.3.6.1.

25.4.5.2.3 Restrictions

In these instructions, *Rd*, *Rn*, and *Rm* must only specify R0-R7.

25.4.5.2.4 Condition flags

These instructions:

- update the N and Z flags according to the result
- do not affect the C or V flag.

25.4.5.2.5 Examples

```
ANDS    R2, R2, R1
     ORRS    R2, R2, R5
     ANDS    R5, R5, R8
     EORS    R7, R7, R6
     BICS    R0, R0, R1
```

25.4.5.3 ASR, LSL, LSR, and ROR

Arithmetic Shift Right, Logical Shift Left, Logical Shift Right, and Rotate Right.

25.4.5.3.1 Syntax

ASRS {Rd,} *Rm*, *Rs*

ASRS {Rd,} *Rm*, *#imm*

LSLS {Rd,} *Rm*, *Rs*

LSLS {Rd,} *Rm*, *#imm*

LSRS {Rd,} *Rm*, *Rs*

LSRS {Rd,} *Rm*, *#imm*

RORS {Rd,} *Rm*, *Rs*

where:

Rd is the destination register. If *Rd* is omitted, it is assumed to take the same value as *Rm*.

Rm is the register holding the value to be shifted.

Rs is the register holding the shift length to apply to the value in *Rm*.

imm is the shift length.

The range of shift length depends on the instruction:

ASR — shift length from 1 to 32

LSL — shift length from 0 to 31

LSR — shift length from 1 to 32.

Remark: MOVS Rd, *Rm* is a pseudonym for LSLS Rd, *Rm*, #0.

25.4.5.3.2 Operation

ASR, LSL, LSR, and ROR perform an arithmetic-shift-left, logical-shift-left, logical-shift-right or a right-rotation of the bits in the register *Rm* by the number of places specified by the immediate *imm* or the value in the least-significant byte of the register specified by *Rs*.

For details on what result is generated by the different instructions, see Section 25–25.4.3.3.

25.4.5.3.3 Restrictions

In these instructions, *Rd*, *Rm*, and *Rs* must only specify R0-R7. For non-immediate instructions, *Rd* and *Rm* must specify the same register.

25.4.5.3.4 Condition flags

These instructions update the N and Z flags according to the result.

The C flag is updated to the last bit shifted out, except when the shift length is 0, see Section 25–25.4.3.3. The V flag is left unmodified.

25.4.5.3.5 Examples

```
ASRS    R7, R5, #9  ; Arithmetic shift right by 9 bits
     LSLS    R1, R2, #3  ; Logical shift left by 3 bits with flag update
     LSRS    R4, R5, #6  ; Logical shift right by 6 bits
     RORS    R4, R4, R6  ; Rotate right by the value in the bottom byte of R6.
```

25.4.5.4 CMP and CMN

Compare and Compare Negative.

25.4.5.4.1 Syntax

CMN *Rn*, *Rm*

CMP *Rn*, *#imm*

CMP *Rn*, *Rm*

where:

Rn is the register holding the first operand.

Rm is the register to compare with.

imm is the immediate value to compare with.

25.4.5.4.2 Operation

These instructions compare the value in a register with either the value in another register or an immediate value. They update the condition flags on the result, but do not write the result to a register.

The CMP instruction subtracts either the value in the register specified by *Rm*, or the immediate *imm* from the value in *Rn* and updates the flags. This is the same as a SUBS instruction, except that the result is discarded.

The CMN instruction adds the value of *Rm* to the value in *Rn* and updates the flags. This is the same as an ADDS instruction, except that the result is discarded.

25.4.5.4.3 Restrictions

For the:

- CMN

  instruction *Rn*, and *Rm* must only specify R0-R7.
- CMP instruction:
  - *Rn* and *Rm* can specify R0-R14
  - immediate must be in the range 0-255.

25.4.5.4.4 Condition flags

These instructions update the N, Z, C and V flags according to the result.

25.4.5.4.5 Examples

```
CMP     R2, R9
     CMN     R0, R2
```

25.4.5.5 MOV and MVN

Move and Move NOT.

25.4.5.5.1 Syntax

MOV{S} *Rd*, *Rm*

MOVS *Rd*, #*imm*

MVNS *Rd*, *Rm*

where:

S is an optional suffix. If *S* is specified, the condition code flags are updated on the result of the operation, see Section 25–25.4.3.6.

Rd is the destination register.

Rm is a register.

imm is any value in the range 0-255.

25.4.5.5.2 Operation

The MOV instruction copies the value of *Rm* into *Rd*.

The MOVS instruction performs the same operation as the MOV instruction, but also updates the N and Z flags.

The MVNS instruction takes the value of *Rm*, performs a bitwise logical negate operation on the value, and places the result into *Rd*.

25.4.5.5.3 Restrictions

In these instructions, *Rd*, and *Rm* must only specify R0-R7.

When *Rd* is the PC in a MOV instruction:

- Bit[0] of the result is discarded.
- A branch occurs to the address created by forcing bit[0] of the result to 0. The T-bit remains unmodified.

Remark: Though it is possible to use MOV as a branch instruction, ARM strongly recommends the use of a BX or BLX instruction to branch for software portability.

25.4.5.5.4 Condition flags

If *S* is specified, these instructions:

- update the N and Z flags according to the result
- do not affect the C or V flags.

25.4.5.5.5 Example

```
MOVS  R0, #0x000B     ; Write value of 0x000B to R0, flags get updated
      MOVS  R1, #0x0        ; Write value of zero to R1, flags are updated
      MOV   R10, R12        ; Write value in R12 to R10, flags are not updated
      MOVS  R3, #23         ; Write value of 23 to R3
      MOV   R8, SP          ; Write value of stack pointer to R8
      MVNS  R2, R0          ; Write inverse of R0 to the R2 and update flags
```

25.4.5.6 MULS

Multiply using 32-bit operands, and producing a 32-bit result.

25.4.5.6.1 Syntax

MULS Rd, *Rn*, *Rm*

where:

Rd is the destination register.

Rn, Rm are registers holding the values to be multiplied.

25.4.5.6.2 Operation

The MUL instruction multiplies the values in the registers specified by *Rn* and *Rm*, and places the least significant 32 bits of the result in *Rd*. The condition code flags are updated on the result of the operation, see Section 25–25.4.3.6.

The results of this instruction does not depend on whether the operands are signed or unsigned.

25.4.5.6.3 Restrictions

In this instruction:

- *Rd*, *Rn*, and *Rm* must only specify R0-R7
- *Rd* must be the same as *Rm*.

25.4.5.6.4 Condition flags

This instruction:

- updates the N and Z flags according to the result
- does not affect the C or V flags.

25.4.5.6.5 Examples

```
MULS    R0, R2, R0      ; Multiply with flag update, R0 = R0 x R2
```

25.4.5.7 REV, REV16, and REVSH

Reverse bytes.

25.4.5.7.1 Syntax

REV Rd, *Rn*

REV16 Rd, *Rn*

REVSH Rd, *Rn*

where:

Rd is the destination register.

Rn is the source register.

25.4.5.7.2 Operation

Use these instructions to change endianness of data:

REV — converts 32-bit big-endian data into little-endian data or 32-bit little-endian data into big-endian data.

REV16 — converts two packed 16-bit big-endian data into little-endian data or two packed 16-bit little-endian data into big-endian data.

REVSH — converts 16-bit signed big-endian data into 32-bit signed little-endian data or 16-bit signed little-endian data into 32-bit signed big-endian data.

25.4.5.7.3 Restrictions

In these instructions, *Rd*, and *Rn* must only specify R0-R7.

25.4.5.7.4 Condition flags

These instructions do not change the flags.

25.4.5.7.5 Examples

```
REV    R3, R7  ; Reverse byte order of value in R7 and write it to R3
     REV16  R0, R0  ; Reverse byte order of each 16-bit halfword in R0
     REVSH  R0, R5  ; Reverse signed halfword
```

25.4.5.8 SXT and UXT

Sign extend and Zero extend.

25.4.5.8.1 Syntax

SXTB Rd, *Rm*

SXTH Rd, *Rm*

UXTB Rd, *Rm*

UXTH Rd, *Rm*

where:

Rd is the destination register.

Rm is the register holding the value to be extended.

25.4.5.8.2 Operation

These instructions extract bits from the resulting value:

- SXTB extracts bits[7:0] and sign extends to 32 bits
- UXTB extracts bits[7:0] and zero extends to 32 bits
- SXTH extracts bits[15:0] and sign extends to 32 bits
- UXTH extracts bits[15:0] and zero extends to 32 bits.

25.4.5.8.3 Restrictions

In these instructions, *Rd* and *Rm* must only specify R0-R7.

25.4.5.8.4 Condition flags

These instructions do not affect the flags.

25.4.5.8.5 Examples

```
SXTH  R4, R6          ; Obtain the lower halfword of the
                           ; value in R6 and then sign extend to
                           ; 32 bits and write the result to R4.
     UXTB  R3, R1          ; Extract lowest byte of the value in R10 and zero
                           ; extend it, and write the result to R3
```

25.4.5.9 TST

Test bits.

25.4.5.9.1 Syntax

TST *Rn*, *Rm*

where:

Rn is the register holding the first operand.

Rm the register to test against.

25.4.5.9.2 Operation

This instruction tests the value in a register against another register. It updates the condition flags based on the result, but does not write the result to a register.

The TST instruction performs a bitwise AND operation on the value in *Rn* and the value in *Rm*. This is the same as the ANDS instruction, except that it discards the result.

To test whether a bit of *Rn* is 0 or 1, use the TST instruction with a register that has that bit set to 1 and all other bits cleared to 0.

25.4.5.9.3 Restrictions

In these instructions, *Rn* and *Rm* must only specify R0-R7.

25.4.5.9.4 Condition flags

This instruction:

- updates the N and Z flags according to the result
- does not affect the C or V flags.

25.4.5.9.5 Examples

```
TST     R0, R1  ; Perform bitwise AND of R0 value and R1 value,
                ; condition code flags are updated but result is discarded
```

25.4.6 Branch and control instructions

Table 372 shows the branch and control instructions:

Table 372. Branch and control instructions

Mnemonic	Brief description	See
B{cc}	Branch {conditionally}	Section 25–25.4.6.1
BL	Branch with Link	Section 25–25.4.6.1
BLX	Branch indirect with Link	Section 25–25.4.6.1
BX	Branch indirect	Section 25–25.4.6.1

25.4.6.1 B, BL, BX, and BLX

Branch instructions.

25.4.6.1.1 Syntax

B{*cond*} *label*

BL *label*

BX *Rm*

BLX *Rm*

where:

cond is an optional condition code, see Section 25–25.4.3.6.

label is a PC-relative expression. See Section 25–25.4.3.5.

Rm is a register providing the address to branch to.

25.4.6.1.2 Operation

All these instructions cause a branch to the address indicated by *label* or contained in the register specified by *Rm*. In addition:

- The BL and BLX instructions write the address of the next instruction to LR, the link register R14.
- The BX and BLX instructions result in a HardFault exception if bit[0] of *Rm* is 0.

BL and BLX instructions also set bit[0] of the LR to 1. This ensures that the value is suitable for use by a subsequent POP {PC} or BX instruction to perform a successful return branch.

Table 373 shows the ranges for the various branch instructions.

Table 373. Branch ranges

Instruction	Branch range
B *label*	–2 KB to +2 KB
B*cond label*	–256 bytes to +254 bytes
BL *label*	–16 MB to +16 MB
BX *Rm*	Any value in register
BLX *Rm*	Any value in register

25.4.6.1.3 Restrictions

In these instructions:

- Do not use SP or PC in the BX or BLX instruction.
- For BX and BLX, bit[0] of *Rm* must be 1 for correct execution. Bit[0] is used to update the EPSR T-bit and is discarded from the target address.

Remark: B*cond* is the only conditional instruction on the Cortex-M0 processor.

25.4.6.1.4 Condition flags

These instructions do not change the flags.

25.4.6.1.5 Examples

```
B      loopA  ; Branch to loopA
      BL     funC   ; Branch with link (Call) to function funC, return address
                    ; stored in LR
```

```
        BX      LR      ; Return from function call
        BLX     R0      ; Branch with link and exchange (Call) to a address stored
                        ; in R0

    BEQ     labelD ; Conditionally branch to labelD if last flag setting

                    ; instruction set the Z flag, else do not branch.
```

25.4.7 Miscellaneous instructions

Table 374 shows the remaining Cortex-M0 instructions:

Table 374. Miscellaneous instructions

Mnemonic	Brief description	See
BKPT	Breakpoint	Section 25–25.4.7.1
CPSID	Change Processor State, Disable Interrupts	Section 25–25.4.7.2
CPSIE	Change Processor State, Enable Interrupts	Section 25–25.4.7.2
DMB	Data Memory Barrier	Section 25–25.4.7.3
DSB	Data Synchronization Barrier	Section 25–25.4.7.4
ISB	Instruction Synchronization Barrier	Section 25–25.4.7.5
MRS	Move from special register to register	Section 25–25.4.7.6
MSR	Move from register to special register	Section 25–25.4.7.7
NOP	No Operation	Section 25–25.4.7.8
SEV	Send Event	Section 25–25.4.7.9
SVC	Supervisor Call	Section 25–25.4.7.10
WFE	Wait For Event	Section 25–25.4.7.11
WFI	Wait For Interrupt	Section 25–25.4.7.12

25.4.7.1 BKPT

Breakpoint.

25.4.7.1.1 Syntax

BKPT *#imm*

where:

imm is an integer in the range 0-255.

25.4.7.1.2 Operation

The BKPT instruction causes the processor to enter Debug state. Debug tools can use this to investigate system state when the instruction at a particular address is reached. *imm* is ignored by the processor. If required, a debugger can use it to store additional information about the breakpoint.

The processor might also produce a HardFault or go in to lockup if a debugger is not attached when a BKPT instruction is executed. See Section 25–25.3.4.1 for more information.

25.4.7.1.3 Restrictions

There are no restrictions.

25.4.7.1.4 Condition flags

This instruction does not change the flags.

25.4.7.1.5 Examples

```
BKPT #0     ; Breakpoint with immediate value set to 0x0.
```

25.4.7.2 CPS

Change Processor State.

25.4.7.2.1 Syntax

CPSID i

CPSIE i

25.4.7.2.2 Operation

CPS changes the PRIMASK special register values. CPSID causes interrupts to be disabled by setting PRIMASK. CPSIE cause interrupts to be enabled by clearing PRIMASK.See Section 25–25.3.1.3.6 for more information about these registers.

25.4.7.2.3 Restrictions

There are no restrictions.

25.4.7.2.4 Condition flags

This instruction does not change the condition flags.

25.4.7.2.5 Examples

```
CPSID i ; Disable all interrupts except NMI (set PRIMASK)

CPSIE i ; Enable interrupts (clear PRIMASK)
```

25.4.7.3 DMB

Data Memory Barrier.

25.4.7.3.1 Syntax

DMB

25.4.7.3.2 Operation

DMB acts as a data memory barrier. It ensures that all explicit memory accesses that appear in program order before the DMB instruction are observed before any explicit memory accesses that appear in program order after the DMB instruction. DMB does not affect the ordering of instructions that do not access memory.

25.4.7.3.3 Restrictions

There are no restrictions.

25.4.7.3.4 Condition flags

This instruction does not change the flags.

25.4.7.3.5 Examples

```
DMB  ; Data Memory Barrier
```

25.4.7.4 DSB

Data Synchronization Barrier.

25.4.7.4.1 Syntax

DSB

25.4.7.4.2 Operation

DSB acts as a special data synchronization memory barrier. Instructions that come after the DSB, in program order, do not execute until the DSB instruction completes. The DSB instruction completes when all explicit memory accesses before it complete.

25.4.7.4.3 Restrictions

There are no restrictions.

25.4.7.4.4 Condition flags

This instruction does not change the flags.

25.4.7.4.5 Examples

```
DSB ; Data Synchronisation Barrier
```

25.4.7.5 ISB

Instruction Synchronization Barrier.

25.4.7.5.1 Syntax

ISB

25.4.7.5.2 Operation

ISB acts as an instruction synchronization barrier. It flushes the pipeline of the processor, so that all instructions following the ISB are fetched from cache or memory again, after the ISB instruction has been completed.

25.4.7.5.3 Restrictions

There are no restrictions.

25.4.7.5.4 Condition flags

This instruction does not change the flags.

25.4.7.5.5 Examples

```
ISB  ; Instruction Synchronisation Barrier
```

25.4.7.6 MRS

Move the contents of a special register to a general-purpose register.

25.4.7.6.1 Syntax

MRS *Rd*, *spec_reg*

where:

Rd is the general-purpose destination register.

spec_reg is one of the special-purpose registers: APSR, IPSR, EPSR, IEPSR, IAPSR, EAPSR, PSR, MSP, PSP, PRIMASK, or CONTROL.

25.4.7.6.2 Operation

MRS stores the contents of a special-purpose register to a general-purpose register. The MRS instruction can be combined with the MR instruction to produce read-modify-write sequences, which are suitable for modifying a specific flag in the PSR.

See Section 25–25.4.7.7.

25.4.7.6.3 Restrictions

In this instruction, *Rd* must not be SP or PC.

25.4.7.6.4 Condition flags

This instruction does not change the flags.

25.4.7.6.5 Examples

```
MRS  R0, PRIMASK ; Read PRIMASK value and write it to R0
```

25.4.7.7 MSR

Move the contents of a general-purpose register into the specified special register.

25.4.7.7.1 Syntax

MSR *spec_reg*, *Rn*

where:

Rn is the general-purpose source register.

spec_reg is the special-purpose destination register: APSR, IPSR, EPSR, IEPSR, IAPSR, EAPSR, PSR, MSP, PSP, PRIMASK, or CONTROL.

25.4.7.7.2 Operation

MSR updates one of the special registers with the value from the register specified by *Rn*.

See Section 25–25.4.7.6.

25.4.7.7.3 Restrictions

In this instruction, *Rn* must not be SP and must not be PC.

25.4.7.7.4 Condition flags

This instruction updates the flags explicitly based on the value in *Rn*.

25.4.7.7.5 Examples

```
MSR  CONTROL, R1 ; Read R1 value and write it to the CONTROL register
```

25.4.7.8 NOP

No Operation.

25.4.7.8.1 Syntax

NOP

25.4.7.8.2 Operation

NOP performs no operation and is not guaranteed to be time consuming. The processor might remove it from the pipeline before it reaches the execution stage.

Use NOP for padding, for example to place the subsequent instructions on a 64-bit boundary.

25.4.7.8.3 Restrictions

There are no restrictions.

25.4.7.8.4 Condition flags

This instruction does not change the flags.

25.4.7.8.5 Examples

```
NOP  ; No operation
```

25.4.7.9 SEV

Send Event.

25.4.7.9.1 Syntax

SEV

25.4.7.9.2 Operation

SEV causes an event to be signaled to all processors within a multiprocessor system. It also sets the local event register, see Section 25–25.3.5.

See also Section 25–25.4.7.11.

25.4.7.9.3 Restrictions

There are no restrictions.

25.4.7.9.4 Condition flags

This instruction does not change the flags.

25.4.7.9.5 Examples

```
SEV ; Send Event
```

25.4.7.10 SVC

Supervisor Call.

25.4.7.10.1 Syntax

SVC #*imm*

where:

imm is an integer in the range 0-255.

25.4.7.10.2 Operation

The SVC instruction causes the SVC exception.

imm is ignored by the processor. If required, it can be retrieved by the exception handler to determine what service is being requested.

25.4.7.10.3 Restrictions

There are no restrictions.

25.4.7.10.4 Condition flags

This instruction does not change the flags.

25.4.7.10.5 Examples

```
SVC  #0x32 ; Supervisor Call (SVC handler can extract the immediate value
               ; by locating it via the stacked PC)
```

25.4.7.11 WFE

Wait For Event.

Remark: The WFE instruction is not implemented on the LPC11U1x.

25.4.7.11.1 Syntax

WFE

25.4.7.11.2 Operation

If the event register is 0, WFE suspends execution until one of the following events occurs:

- an exception, unless masked by the exception mask registers or the current priority level
- an exception enters the Pending state, if SEVONPEND in the System Control Register is set
- a Debug Entry request, if debug is enabled
- an event signaled by a peripheral or another processor in a multiprocessor system using the SEV instruction.

If the event register is 1, WFE clears it to 0 and completes immediately.

For more information see Section 25–25.3.5.

Remark: WFE is intended for power saving only. When writing software assume that WFE might behave as NOP.

25.4.7.11.3 Restrictions

There are no restrictions.

25.4.7.11.4 Condition flags

This instruction does not change the flags.

25.4.7.11.5 Examples

```
WFE  ; Wait for event
```

25.4.7.12 WFI

Wait for Interrupt.

25.4.7.12.1 Syntax

WFI

25.4.7.12.2 Operation

WFI

suspends execution until one of the following events occurs:

- an exception
- an interrupt becomes pending which would preempt if PRIMASK was clear
- a Debug Entry request, regardless of whether debug is enabled.

Remark: WFI is intended for power saving only. When writing software assume that WFI might behave as a NOP operation.

25.4.7.12.3 Restrictions

There are no restrictions.

25.4.7.12.4 Condition flags

This instruction does not change the flags.

25.4.7.12.5 Examples

```
WFI ; Wait for interrupt
```

25.5 Peripherals

25.5.1 About the ARM Cortex-M0

The address map of the **Private peripheral bus** (PPB) is:

Table 375. Core peripheral register regions

Address	Core peripheral	Description
0xE000E008-0xE000E00F	System Control Block	Table 25–384
0xE000E010-0xE000E01F	System timer	Table 25–393
0xE000E100-0xE000E4EF	Nested Vectored Interrupt Controller	Table 25–376
0xE000ED00-0xE000ED3F	System Control Block	Table 25–384
0xE000EF00-0xE000EF03	Nested Vectored Interrupt Controller	Table 25–376

In register descriptions, the register **type** is described as follows:

RW — Read and write.

RO — Read-only.

WO — Write-only.

25.5.2 Nested Vectored Interrupt Controller

This section describes the **Nested Vectored Interrupt Controller** (NVIC) and the registers it uses. The NVIC supports:

- 32 interrupts.
- A programmable priority level of 0-3 for each interrupt. A higher level corresponds to a lower priority, so level 0 is the highest interrupt priority.
- Level and pulse detection of interrupt signals.
- Interrupt tail-chaining.
- An external **Non-maskable interrupt** (NMI).

The processor automatically stacks its state on exception entry and unstacks this state on exception exit, with no instruction overhead. This provides low latency exception handling. The hardware implementation of the NVIC registers is:

Table 376. NVIC register summary

Address	Name	Type	Reset value	Description
0xE000E100	ISER	RW	0x00000000	Section 25–25.5.2.2
0xE000E180	ICER	RW	0x00000000	Section 25–25.5.2.3
0xE000E200	ISPR	RW	0x00000000	Section 25–25.5.2.4
0xE000E280	ICPR	RW	0x00000000	Section 25–25.5.2.5
0xE000E400-0xE000E41C	IPR0-7	RW	0x00000000	Section 25–25.5.2.6

25.5.2.1 Accessing the Cortex-M0 NVIC registers using CMSIS

CMSIS functions enable software portability between different Cortex-M profile processors.

To access the NVIC registers when using CMSIS, use the following functions:

Table 377. CMISIS access NVIC functions

CMSIS function	Description
void NVIC_EnableIRQ(IRQn_Type IRQn)[1]	Enables an interrupt or exception.
void NVIC_DisableIRQ(IRQn_Type IRQn)[1]	Disables an interrupt or exception.
void NVIC_SetPendingIRQ(IRQn_Type IRQn)[1]	Sets the pending status of interrupt or exception to 1.
void NVIC_ClearPendingIRQ(IRQn_Type IRQn)[1]	Clears the pending status of interrupt or exception to 0.
uint32_t NVIC_GetPendingIRQ(IRQn_Type IRQn)[1]	Reads the pending status of interrupt or exception. This function returns non-zero value if the pending status is set to 1.
void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)[1]	Sets the priority of an interrupt or exception with configurable priority level to 1.
uint32_t NVIC_GetPriority(IRQn_Type IRQn)[1]	Reads the priority of an interrupt or exception with configurable priority level. This function returns the current priority level.

[1] The input parameter IRQn is the IRQ number, see Table 363 for more information.

25.5.2.2 Interrupt Set-enable Register

The ISER enables interrupts, and shows which interrupts are enabled. See the register summary in Table 376 for the register attributes.

The bit assignments are:

Table 378. ISER bit assignments

Bits	Name	Function
[31:0]	SETENA	Interrupt set-enable bits. Write: 0 = no effect 1 = enable interrupt. Read: 0 = interrupt disabled 1 = interrupt enabled.

If a pending interrupt is enabled, the NVIC activates the interrupt based on its priority. If an interrupt is not enabled, asserting its interrupt signal changes the interrupt state to pending, but the NVIC never activates the interrupt, regardless of its priority.

25.5.2.3 Interrupt Clear-enable Register

The ICER disables interrupts, and show which interrupts are enabled. See the register summary in Table 25–376 for the register attributes.

The bit assignments are:

Table 379. ICER bit assignments

Bits	Name	Function
[31:0]	CLRENA	Interrupt clear-enable bits. Write: 0 = no effect 1 = disable interrupt. Read: 0 = interrupt disabled 1 = interrupt enabled.

25.5.2.4 Interrupt Set-pending Register

The ISPR forces interrupts into the pending state, and shows which interrupts are pending. See the register summary in Table 25–376 for the register attributes.

The bit assignments are:

Table 380. ISPR bit assignments

Bits	Name	Function
[31:0]	SETPEND	Interrupt set-pending bits. Write: 0 = no effect 1 = changes interrupt state to pending. Read: 0 = interrupt is not pending 1 = interrupt is pending.

Remark: Writing 1 to the ISPR bit corresponding to:

- an interrupt that is pending has no effect
- a disabled interrupt sets the state of that interrupt to pending.

25.5.2.5 Interrupt Clear-pending Register

The ICPR removes the pending state from interrupts, and shows which interrupts are pending. See the register summary in Table 25–376 for the register attributes.

The bit assignments are:

Table 381. ICPR bit assignments

Bits	Name	Function
[31:0]	CLRPEND	Interrupt clear-pending bits. Write: 0 = no effect 1 = removes pending state an interrupt. Read: 0 = interrupt is not pending 1 = interrupt is pending.

Remark: Writing 1 to an ICPR bit does not affect the active state of the corresponding interrupt.

25.5.2.6 Interrupt Priority Registers

The IPR0-IPR7 registers provide an 2-bit priority field for each interrupt. These registers are only word-accessible. See the register summary in Table 25–376 for their attributes. Each register holds four priority fields as shown:

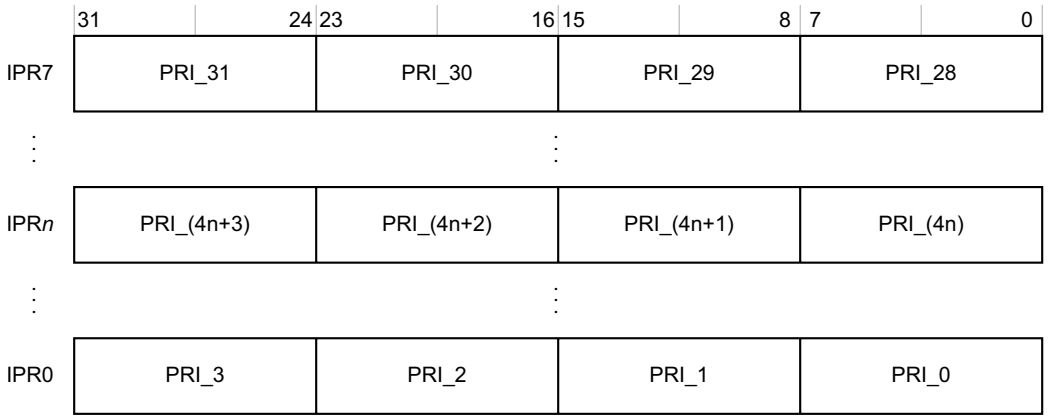


Fig 74. IPR register

Table 382. IPR bit assignments

Bits	Name	Function
[31:24]	Priority, byte offset 3	Each priority field holds a priority value, 0-3. The lower the value, the greater the priority of the corresponding interrupt. The processor implements only bits[7:6] of each field, bits [5:0] read as zero and ignore writes.
[23:16]	Priority, byte offset 2	
[15:8]	Priority, byte offset 1	
[7:0]	Priority, byte offset 0	

See Section 25–25.5.2.1 for more information about the access to the interrupt priority array, which provides the software view of the interrupt priorities.

Find the IPR number and byte offset for interrupt **M** as follows:

- the corresponding IPR number, **N**, is given by **N** = **N** DIV 4
- the byte offset of the required Priority field in this register is **M** MOD 4, where:
  - byte offset 0 refers to register bits[7:0]
  - byte offset 1 refers to register bits[15:8]
  - byte offset 2 refers to register bits[23:16]
  - byte offset 3 refers to register bits[31:24].

25.5.2.7 Level-sensitive and pulse interrupts

The processor supports both level-sensitive and pulse interrupts. Pulse interrupts are also described as edge-triggered interrupts.

A level-sensitive interrupt is held asserted until the peripheral deasserts the interrupt signal. Typically this happens because the ISR accesses the peripheral, causing it to clear the interrupt request. A pulse interrupt is an interrupt signal sampled synchronously on the

rising edge of the processor clock. To ensure the NVIC detects the interrupt, the peripheral must assert the interrupt signal for at least one clock cycle, during which the NVIC detects the pulse and latches the interrupt.

When the processor enters the ISR, it automatically removes the pending state from the interrupt, see Section 25.5.2.7.1. For a level-sensitive interrupt, if the signal is not deasserted before the processor returns from the ISR, the interrupt becomes pending again, and the processor must execute its ISR again. This means that the peripheral can hold the interrupt signal asserted until it no longer needs servicing.

25.5.2.7.1 Hardware and software control of interrupts

The Cortex-M0 latches all interrupts. A peripheral interrupt becomes pending for one of the following reasons:

- the NVIC detects that the interrupt signal is active and the corresponding interrupt is not active
- the NVIC detects a rising edge on the interrupt signal
- software writes to the corresponding interrupt set-pending register bit, see Section 25–25.5.2.4.

A pending interrupt remains pending until one of the following:

- The processor enters the ISR for the interrupt. This changes the state of the interrupt from pending to active. Then:
  - For a level-sensitive interrupt, when the processor returns from the ISR, the NVIC samples the interrupt signal. If the signal is asserted, the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR. Otherwise, the state of the interrupt changes to inactive.
  - For a pulse interrupt, the NVIC continues to monitor the interrupt signal, and if this is pulsed the state of the interrupt changes to pending and active. In this case, when the processor returns from the ISR the state of the interrupt changes to pending, which might cause the processor to immediately re-enter the ISR.

    If the interrupt signal is not pulsed while the processor is in the ISR, when the processor returns from the ISR the state of the interrupt changes to inactive.
- Software writes to the corresponding interrupt clear-pending register bit.

  For a level-sensitive interrupt, if the interrupt signal is still asserted, the state of the interrupt does not change. Otherwise, the state of the interrupt changes to inactive.

  For a pulse interrupt, state of the interrupt changes to:
  - inactive, if the state was pending
  - active, if the state was active and pending.

25.5.2.8 NVIC usage hints and tips

Ensure software uses correctly aligned register accesses. The processor does not support unaligned accesses to NVIC registers.

An interrupt can enter pending state even if it is disabled. Disabling an interrupt only prevents the processor from taking that interrupt.

25.5.2.8.1 NVIC programming hints

Software uses the `CPSIE i` and instructions to enable and disable interrupts. The CMSIS provides the following intrinsic functions for these instructions:

```
void __disable_irq(void) // Disable Interrupts

void __enable_irq(void) // Enable Interrupts
```

In addition, the CMSIS provides a number of functions for NVIC control, including:

Table 383. CMSIS functions for NVIC control

CMSIS interrupt control function	Description
`void NVIC_EnableIRQ(IRQn_t IRQn)`	Enable IRQn
`void NVIC_DisableIRQ(IRQn_t IRQn)`	Disable IRQn
`uint32_t NVIC_GetPendingIRQ (IRQn_t IRQn)`	Return true (1) if IRQn is pending
`void NVIC_SetPendingIRQ (IRQn_t IRQn)`	Set IRQn pending
`void NVIC_ClearPendingIRQ (IRQn_t IRQn)`	Clear IRQn pending status
`void NVIC_SetPriority (IRQn_t IRQn, uint32_t priority)`	Set priority for IRQn
`uint32_t NVIC_GetPriority (IRQn_t IRQn)`	Read priority of IRQn
`void NVIC_SystemReset (void)`	Reset the system

The input parameter IRQn is the IRQ number, see Table 25–363 for more information. For more information about these functions, see the CMSIS documentation.

25.5.3 System Control Block

The **System Control Block** (SCB) provides system implementation information, and system control. This includes configuration, control, and reporting of the system exceptions. The SCB registers are:

Table 384. Summary of the SCB registers

Address	Name	Type	Reset value	Description
`0xE000ED00`	CPUID	RO	`0x410CC200`	Section 25.5.3.2
`0xE000ED04`	ICSR	RW[1]	`0x00000000`	Section 25–25.5.3.3
`0xE000ED0C`	AIRCR	RW[1]	`0xFA050000`	Section 25–25.5.3.4
`0xE000ED10`	SCR	RW	`0x00000000`	Section 25–25.5.3.5
`0xE000ED14`	CCR	RO	`0x00000204`	Section 25–25.5.3.6
`0xE000ED1C`	SHPR2	RW	`0x00000000`	Section 25–25.5.3.7.1
`0xE000ED20`	SHPR3	RW	`0x00000000`	Section 25–25.5.3.7.2

[1] See the register description for more information.

25.5.3.1 The CMSIS mapping of the Cortex-M0 SCB registers

To improve software efficiency, the CMSIS simplifies the SCB register presentation. In the CMSIS, the array `SHP[1]` corresponds to the registers SHPR2-SHPR3.

25.5.3.2 CPUID Register

The CPUID register contains the processor part number, version, and implementation information. See the register summary in for its attributes. The bit assignments are:

Table 385. CPUID register bit assignments

Bits	Name	Function
[31:24]	Implementer	Implementer code: 0x41 = ARM
[23:20]	Variant	Variant number, the r value in the r**n**p**n** product revision identifier: 0x0 = Revision 0
[19:16]	Constant	Constant that defines the architecture of the processor:, reads as 0xC = ARMv6-M architecture
[15:4]	Partno	Part number of the processor: 0xC20 = Cortex-M0
[3:0]	Revision	Revision number, the p value in the r**n**p**n** product revision identifier: 0x0 = Patch 0

25.5.3.3 Interrupt Control and State Register

The ICSR:

- provides:
  - a set-pending bit for the **Non-Maskable Interrupt** (NMI) exception
  - set-pending and clear-pending bits for the PendSV and SysTick exceptions
- indicates:
  - the exception number of the exception being processed
  - whether there are preempted active exceptions
  - the exception number of the highest priority pending exception
  - whether any interrupts are pending.

See the register summary in Table 25–384 for the ICSR attributes. The bit assignments are:

Table 386. ICSR bit assignments

Bits	Name	Type	Function
[31]	NMIPENDSET	RW	NMI set-pending bit. Write: 0 = no effect 1 = changes NMI exception state to pending. Read: 0 = NMI exception is not pending 1 = NMI exception is pending. Because NMI is the highest-priority exception, normally the processor enters the NMI exception handler as soon as it detects a write of 1 to this bit. Entering the handler then clears this bit to 0. This means a read of this bit by the NMI exception handler returns 1 only if the NMI signal is reasserted while the processor is executing that handler.
[30:29]	-	-	Reserved.
[28]	PENDSVSET	RW	PendSV set-pending bit. Write: 0 = no effect 1 = changes PendSV exception state to pending. Read: 0 = PendSV exception is not pending 1 = PendSV exception is pending. Writing 1 to this bit is the only way to set the PendSV exception state to pending.
[27]	PENDSVCLR	WO	PendSV clear-pending bit. Write: 0 = no effect 1 = removes the pending state from the PendSV exception.
[26]	PENDSTSET	RW	SysTick exception set-pending bit. Write: 0 = no effect 1 = changes SysTick exception state to pending. Read: 0 = SysTick exception is not pending 1 = SysTick exception is pending.
[25]	PENDSTCLR	WO	SysTick exception clear-pending bit. Write: 0 = no effect 1 = removes the pending state from the SysTick exception. This bit is WO. On a register read its value is Unknown.
[24:23]	-	-	Reserved.

Table 386. ICSR bit assignments

Bits	Name	Type	Function
[22]	ISRPENDING	RO	Interrupt pending flag, excluding NMI and Faults: 0 = interrupt not pending 1 = interrupt pending.
[21:18]	-	-	Reserved.
[17:12]	VECTPENDING	RO	Indicates the exception number of the highest priority pending enabled exception: 0 = no pending exceptions Nonzero = the exception number of the highest priority pending enabled exception.
[11:6]	-	-	Reserved.
[5:0]	VECTACTIVE[1]	RO	Contains the active exception number: 0 = Thread mode Nonzero = The exception number[1] of the currently active exception. **Remark:** Subtract 16 from this value to obtain the CMSIS IRQ number that identifies the corresponding bit in the Interrupt Clear-Enable, Set-Enable, Clear-Pending, Set-pending, and Priority Register, see Table 25–358.

[1] This is the same value as IPSR bits[5:0], see Table 25–358.

When you write to the ICSR, the effect is Unpredictable if you:

- write 1 to the PENDSVSET bit and write 1 to the PENDSVCLR bit
- write 1 to the PENDSTSET bit and write 1 to the PENDSTCLR bit.

25.5.3.4 Application Interrupt and Reset Control Register

The AIRCR provides endian status for data accesses and reset control of the system. See the register summary in Table 25–384 and Table 25–387 for its attributes.

To write to this register, you must write `0x05FA` to the VECTKEY field, otherwise the processor ignores the write.

The bit assignments are:

Table 387. AIRCR bit assignments

Bits	Name	Type	Function
[31:16]	Read: Reserved Write: VECTKEY	RW	Register key: Reads as Unknown On writes, write `0x05FA` to VECTKEY, otherwise the write is ignored.
[15]	ENDIANESS	RO	Data endianness implemented: 0 = Little-endian 1 = Big-endian.
[14:3]	-	-	Reserved

Table 387. AIRCR bit assignments

Bits	Name	Type	Function
[2]	SYSRESETREQ	WO	System reset request: 0 = no effect 1 = requests a system level reset. This bit reads as 0.
[1]	VECTCLRACTIVE	WO	Reserved for debug use. This bit reads as 0. When writing to the register you must write 0 to this bit, otherwise behavior is Unpredictable.
[0]	-	-	Reserved.

25.5.3.5 System Control Register

The SCR controls features of entry to and exit from low power state. See the register summary in Table 25–384 for its attributes. The bit assignments are:

Table 388. SCR bit assignments

Bits	Name	Function
[31:5]	-	Reserved.
[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.
[3]	-	Reserved.
[2]	SLEEPDEEP	Controls whether the processor uses sleep or deep sleep as its low power mode: 0 = sleep 1 = deep sleep.
[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]	-	Reserved.

25.5.3.6 Configuration and Control Register

The CCR is a read-only register and indicates some aspects of the behavior of the Cortex-M0 processor. See the register summary in Table 25–384 for the CCR attributes.

The bit assignments are:

Table 389. CCR bit assignments

Bits	Name	Function
[31:10]	-	Reserved.
[9]	STKALIGN	Always reads as one, indicates 8-byte stack alignment on exception entry. On exception entry, the processor uses bit[9] of the stacked PSR to indicate the stack alignment. On return from the exception it uses this stacked bit to restore the correct stack alignment.
[8:4]	-	Reserved.
[3]	UNALIGN_TRP	Always reads as one, indicates that all unaligned accesses generate a HardFault.
[2:0]	-	Reserved.

25.5.3.7 System Handler Priority Registers

The SHPR2-SHPR3 registers set the priority level, 0 to 3, of the exception handlers that have configurable priority.

SHPR2-SHPR3 are word accessible. See the register summary in Table 25–384 for their attributes.

To access to the system exception priority level using CMSIS, use the following CMSIS functions:

- `uint32_t NVIC_GetPriority(IRQn_Type IRQn)`
- `void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)`

The input parameter `IRQn` is the IRQ number, see Table 25–363 for more information.

The system fault handlers, and the priority field and register for each handler are:

Table 390. System fault handler priority fields

Handler	Field	Register description
SVCall	PRI_11	Section 25–25.5.3.7.1
PendSV	PRI_14	Section 25–25.5.3.7.2
SysTick	PRI_15	

Each PRI_N field is 8 bits wide, but the processor implements only bits[7:6] of each field, and bits[5:0] read as zero and ignore writes.

25.5.3.7.1 System Handler Priority Register 2

The bit assignments are:

Table 391. SHPR2 register bit assignments

Bits	Name	Function
[31:24]	PRI_11	Priority of system handler 11, SVCall
[23:0]	-	Reserved

25.5.3.7.2 System Handler Priority Register 3

The bit assignments are:

Table 392. SHPR3 register bit assignments

Bits	Name	Function
[31:24]	PRI_15	Priority of system handler 15, SysTick exception
[23:16]	PRI_14	Priority of system handler 14, PendSV
[15:0]	-	Reserved

25.5.3.8 SCB usage hints and tips

Ensure software uses aligned 32-bit word size transactions to access all the SCB registers.

25.5.4 System timer, SysTick

When enabled, the timer counts down from the reload value to zero, reloads (wraps to) the value in the SYST_RVR on the next clock cycle, then decrements on subsequent clock cycles. Writing a value of zero to the SYST_RVR disables the counter on the next wrap. When the counter transitions to zero, the COUNTFLAG status bit is set to 1. Reading SYST_CSR clears the COUNTFLAG bit to 0.

Writing to the SYST_CVR clears the register and the COUNTFLAG status bit to 0. The write does not trigger the SysTick exception logic. Reading the register returns its value at the time it is accessed.

Remark: When the processor is halted for debugging the counter does not decrement.

The system timer registers are:

Table 393. System timer registers summary

Address	Name	Type	Reset value	Description
0xE000E010	SYST_CSR	RW	0x00000000	Section 25.5.4.1
0xE000E014	SYST_RVR	RW	Unknown	Section 25–25.5.4.2
0xE000E018	SYST_CVR	RW	Unknown	Section 25–25.5.4.3
0xE000E01C	SYST_CALIB	RO	0xC0000000 [1]	Section 25–25.5.4.4

[1] SysTick calibration value.

25.5.4.1 SysTick Control and Status Register

The SYST_CSR enables the SysTick features. See the register summary in for its attributes. The bit assignments are:

Table 394. SYST_CSR bit assignments

Bits	Name	Function
[31:17]	-	Reserved.
[16]	COUNTFLAG	Returns 1 if timer counted to 0 since the last read of this register.
[15:3]	-	Reserved.

Table 394. SYST_CSR bit assignments

Bits	Name	Function
[2]	CLKSOURCE	Selects the SysTick timer clock source: 0 = external reference clock 1 = processor clock.
[1]	TICKINT	Enables SysTick exception request: 0 = counting down to zero does not assert the SysTick exception request 1 = counting down to zero to asserts the SysTick exception request.
[0]	ENABLE	Enables the counter: 0 = counter disabled 1 = counter enabled.

25.5.4.2 SysTick Reload Value Register

The SYST_RVR specifies the start value to load into the SYST_CVR. See the register summary in Table 25–393 for its attributes. The bit assignments are:

Table 395. SYST_RVR bit assignments

Bits	Name	Function
[31:24]	-	Reserved.
[23:0]	RELOAD	Value to load into the SYST_CVR when the counter is enabled and when it reaches 0, see Section 25.5.4.2.1.

25.5.4.2.1 Calculating the RELOAD value

The RELOAD value can be any value in the range `0x00000001-0x00FFFFFF`. You can program a value of 0, but this has no effect because the SysTick exception request and COUNTFLAG are activated when counting from 1 to 0.

To generate a multi-shot timer with a period of N processor clock cycles, use a RELOAD value of N-1. For example, if the SysTick interrupt is required every 100 clock pulses, set RELOAD to 99.

25.5.4.3 SysTick Current Value Register

The SYST_CVR contains the current value of the SysTick counter. See the register summary in Table 25–393 for its attributes. The bit assignments are:

Table 396. SYST_CVR bit assignments

Bits	Name	Function
[31:24]	-	Reserved.
[23:0]	CURRENT	Reads return the current value of the SysTick counter. A write of any value clears the field to 0, and also clears the SYST_CSR.COUNTFLAG bit to 0.

25.5.4.4 SysTick Calibration Value Register

The SYST_CALIB register indicates the SysTick calibration properties. See the register summary in Table 25–393 for its attributes. The bit assignments are:

Table 397. SYST_CALIB register bit assignments

Bits	Name	Function
[31]	NOREF	Reads as one. Indicates that no separate reference clock is provided.
[30]	SKEW	Reads as one. Calibration value for the 10ms inexact timing is not known because TENMS is not known. This can affect the suitability of SysTick as a software real time clock.
[29:24]	-	Reserved.
[23:0]	TENMS	Reads as zero. Indicates calibration value is not known.

If calibration information is not known, calculate the calibration value required from the frequency of the processor clock or external clock.

25.5.4.5 SysTick usage hints and tips

The interrupt controller clock updates the SysTick counter.

Ensure software uses word accesses to access the SysTick registers.

If the SysTick counter reload and current value are undefined at reset, the correct initialization sequence for the SysTick counter is:

1. Program reload value.
2. Clear current value.
3. Program Control and Status register.

26.1 Abbreviations

Table 398. Abbreviations

Acronym	Description
ADC	Analog-to-Digital-Converter
AHB	Advanced High-performance Bus
APB	Advanced Peripheral Bus
BOD	BrownOut Detection
CRC	Cyclic Redundancy Check
CITT	Comité Consultatif International Téléphonique et Télégraphique
DMA	Direct Memory Access
FIFO	First-In-First-Out
GPIO	General Purpose Input/Output
I/O	Input/Output
IRC	Internal Resistor-Capacitor
PLL	Phase-Locked Loop
SPI	Serial Peripheral Interface
SSI	Serial Synchronous Interface
TTL	Transistor-Transistor Logic
UART	Universal Asynchronous Receiver/Transmitter

26.2 References

[1] **PCF8576D** — PCF8576D data sheet

[2] **ARM DUI 0497A —** Cortex-M0 Devices Generic User Guide

[3] **ARM DDI 0432C —** Cortex-M0 Revision r0p0 Technical Reference Manual

[4] **LPC122X —** http://www.nxp.com/documents/data_sheet/LPC122X.pdf

[5] **ES_LPC122X —** http://www.nxp.com/documents/errata_sheet/ES_LPC122X.pdf

[6] **ARM Primecell uDMA Controller Technical Reference Manual —** ARM DDI0417A

26.3 Legal information

26.3.1 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.

26.3.2 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.

26.3.3 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.

26.4 Tables

26.5 Figures

26.6 Contents

Chapter 1: LPC122x Introductory information

Chapter 2: LPC122x Memory map

Chapter 3: LPC122x Nested Vectored Interrupt Controller (NVIC)

Chapter 4: LPC122x System control (SYSCON)

Chapter 5: LPC122x Power Monitor Unit (PMU)

Chapter 6: LPC122x I/O configuration (IOCONFIG)

Chapter 7: LPC122x Pin configuration

Chapter 8: LPC122x General Purpose I/O (GPIO)

Chapter 9: LPC122x UART0 with modem control

Chapter 10: LPC122x UART1

Chapter 11: LPC122x I2C-bus controller

Chapter 12: LPC122x SSP controller

Chapter 13: LPC122x 16-bit Counter/timer 0/1 (CT16B0/1)

Chapter 14: LPC122x 32-bit Counter/timer 0/1 (CT32B0/1)

Chapter 15: LPC122x System Tick (SysTick) timer

Chapter 16: LPC122x Real-Time Clock (RTC)

Chapter 17: LPC122x Windowed Watchdog Timer (WWDT)

Chapter 18: LPC122x Comparator

Chapter 19: LPC122x ADC

Chapter 20: LPC122x Flash ISP/IAP

Chapter 21: LPC122x General purpose micro DMA controller

Chapter 22: LPC122x CRC engine

Chapter 23: LPC122x Integer division routines

Chapter 24: LPC122x Serial Wire Debug (SWD)

Chapter 25: LPC122x Appendix ARM Cortex-M0

Chapter 26: Supplementary information

For more information, please visit: http://www.nxp.com
For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: 16 May 2017
Document identifier: UM10441